RUHR-UNIVERSITY BOCHUM

FACULTY OF COMPUTER SCIENCE

RUB

Module Guide

For the Master Study Program

Computer Science

DATE 11.10.2023

https://informatik.rub.de/studium/studiengaenge/inf/minf/

RUB Faculty of Computer Science

www.informatik.rub.de

Where can I find help during my studies?

Important sources of information about studying computer science:

Faculty website:
https://informatik.rub.de/en/

Course website:
https://informatik.rub.de/studium/studiengaenge/inf/

Student advisory service of the Faculty of Computer Science:
https://informatik.rub.de/studium/studienberatung/

Examinations Office of the Faculty of Computer Science: https://informatik.rub.de/studium/pruefungsamt/

Student Council for Computer Science:
https://www.ruhr-uni-bochum.de/fsr-informatik/

If you have any subject-related questions, you can contact the lecturers during their office hours (see individual websites).

Other important contact addresses on campus are:

Central Student Advisory Service:
https://studium.ruhr-uni-bochum.de/en/zentrale-studienberatung
Offers help and coaching for individual problems (also psychological support).

Student finance advisory services:
https://studium.ruhr-uni-bochum.de/de/studienfinanzierung

RUB scholarship advisory service:
https://studium.ruhr-uni-bochum.de/en/scholarship-advice
Advice on scholarships for prospective and current students.

Counselling centre for the inclusion of disabled and chronically ill people:

https://www.akafoe.de/inklusion/

International Office: https://international.ruhr-uni-bochum.de/en/going-abroad-bochum-world

Advice on studying abroad

Dormitory places: https://www.akafoe.de/wohnen/

Study programme objectives:

In the Master's degree programme, competences in the field of computer science are taught according to DQR level 7, which are required for the processing of new complex tasks and problems as well as for the independent control of processes. Graduates have comprehensive, detailed and specialised knowledge at the latest level of knowledge in various areas of computer science. They use specialised technical and conceptual skills to solve problems and can also develop new procedures taking into account given framework conditions. You acquire missing knowledge independently. In group projects, you contribute responsibly to the solution of complex tasks and represent the results to others. Furthermore, graduates of the Master's programme are able to conduct subject-specific discussions in English. Dealing with English specialist literature is a matter of course. Depending on the choice of free elective modules, further interdisciplinary competences can be acquired.

Modularisation concept:

The degree programme is structured in modules. A module is a teaching and learning unit that is self-contained in terms of content and time and extends over one to a maximum of two semesters. The content of the study programme is structured through modularisation.

Each module is assigned a certain number of credit points, which reflect the workload required to successfully complete the module. One credit point refers to the workload of approximately 30 hours for an average student, whereby attendance times are also taken into account. The credit points (CP) to be awarded upon successful completion of the module correspond to the European Credit Transfer System (1 LP = 1 CP = 1 ECTS).

The workload to be completed for the entire degree programme is 120 credit points with an average of 30 credit points per semester.

Forms of examination:

Examinations may take the form of a written examination (also in electronic form), an oral examination, a seminar paper, a presentation, a term paper, a project paper, a practical examination or a colloquium presentation. Combinations of different examination forms are also possible.

Study Plan
Master Computer Science
Ruhr-University Bochum

Nr.	Module	Size (CP)	Recommended semester	Rating
Compulsory elective area				
1	Compulsory elective modules**	a*	1-3	graded
2	Specialisation modules***	b*	1-3	graded
3	Specialised practical course	c*	1-3	ungraded
4	Specialised seminar	3	1-3	graded
5	Project	10	2-3	graded
Free Elective area				
6	Free elective modules****	20	1-3	ungraded
Final Thesis				
7	Master thesis and colloquium	27+3	4	graded
	Total:	120		

* $a \geq 15$, $b \geq 35$, $3 \leq c \geq 5$; $a+b+c \geq 57$

** Here, modules from the compulsory elective catalogue "Basic/Foundation" must be taken. The modules that can be selected are listed in the current module handbook.

*** Here, modules from the specialisation area are to be taken, which are thematically assigned to different focus areas. The modules that can be selected are listed in the current module handbook.

**** Here, (almost) all courses of the RUB course catalogue, as well as courses within the framework of the University Alliance Ruhr can be selected, taking into account §4 (2).

MODULE GUIDE

Overview of the modules

Computer Science - Master (PO 2023)

Basic/Foundation Modules

Advanced Algorithms

Cryptography

Mathematics for Modeling and Data Analysis

Quantum Information and Computation

Software Languages

Theory of machine learning

Design, Implementation and Analysis of Computer Systems

Deterministic Network Calculus

Energy-Aware Computing Systems

High-Performance Computing on Clusters

High-Performance Computing on Multi- and Manycore Processors

Operating-System Concepts and Implementations

Real Time Networks and Systems

Algorithms, Complexity, Data

Geometric Algorithms

Computational complexity theory

Quantum Information and Computation

Computer Security

Blockchain Security and Privacy

Cryptography on hardware-based platforms

Cryptographic protocols

Microarchitectural Attacks and Defenses

Quantum Cryptography

Quantum Information and Computation

Software Engineering and Programming Languages

Autonomous Vehicles and Artificial Intelligence

Autonomous Vehicles and Artificial Intelligence Lab

Foundations of Programming Languages, Verification, and Security

High-Performance Computing on Clusters

High-Performance Computing on Multi- and Manycore Processors

Software Languages

Artificial Intelligence

Autonomous Robotics: Action, Perception, Cognition

Autonomous Vehicles and Artificial Intelligence

Autonomous Vehicles and Artificial Intelligence Lab

Computational Neuroscience: Neural Dynamics

Computational Neuroscience: Single-Neuron Models

Computational Neuroscience: Vision and Memory

Deep Learning

Machine Learning: Evolutionary Algorithms

Machine Learning: Supervised Methods

Machine Learning: Unsupervised Methods (no offer in WS 23/24)

Statistical learning and data mining

Theory of machine learning

Practical Modules

Practical Labs

Project

Seminars

Free Elective Moduls

Free Elective Moduls

Masterthesis

Master's thesis and colloquium

Module title: Advanced Algorithms

Modul code	Credit points	Workload	Term	Frequency	Duration
	9 CP	270 h	see Examination regulations	Wintersemester	1 semester

Courses	Contact time	Self-study	Group size
Advanced Algorithms (212029)	90 h	180 h	40 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Maike Buchin
Lecturers: Prof. Maike Buchin

Module application
M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. Mathematik

M.Sc. IT-Sicherheit / Informationstechnik

Precognitions
Recommended: Basic knowledge of algorithm design and analysis as known from the Bachelor's programme is expected.

Learning goals
- Advanced design methods for algorithms
- Advanced analysis methods for algorithms
- Knowledge of further data structures and methods for designing data structures
- Application of learned methods to new problems

Contents
In der Vorlesung betrachten wir fortgeschrittene Themen der Algorithmik. Nach einer kurzen Wiederholung bekannter Inhalte betrachten wir vor allem Graphalgorithmen, Approximationsalgorithmen und FPT-Algorithmen sowie exakte Algorithmen für NP-schwere Probleme. Ebenfalls betrachten wir einige neue und bekannte Datenstrukturen und deren Analyse. Die betrachteten Probleme dabei sind sowohl kombinatorisch, graphentheoretisch also auch geometrisch.

Teaching methods
Lecture (as slide and blackboard lecture) and exercises in which the presented contents are deepened.

Examination forms
Oral (20-30 minutes) or written final module examination (120 minutes) (to be announced at the beginning of the semester).

Requirements for the award of credits
Passed final module examination and successful participation in exercises.

Significance of the grade for the final grade (for a total of 120 ECTS)
9/97: M.Sc. Computer Science

9/105: M.Sc. Angewandte Informatik

9/91: M.Sc. IT-Sicherheit / Informationstechnik [PO 22]

9/84: M.Sc. IT-Sicherheit / Informationstechnik [PO 20]

Module title: Cryptography

Modul code	Credit points	Workload	Term	Frequency	Duration
	8 CP	240 h	see examination regulations	Wintersemester	1 semester

Courses	Contact time	Self-study	Group size
Kryptographie (212017)	90 h	150 h	100 participants

Teaching language	Requirements for participation
German	None

Module coordinators and Lecturers
Module coordinators: Jun.-Prof. Nils Fleischhacker
Lecturers: Jun.-Prof. Nils Fleischhacker

Module application
B.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. IT-Sicherheit/ Netze und Syteme

M.Sc. Computer Science

M.Sc. Angewandte Informatik

Precognitions
Contents of the lectures Introduction to Cryptography 1 and 2.

Learning goals
The students have an understanding of the essential mathematical methods and procedures on which modern cryptographic procedures are based. The depth of the treatment of the methods goes well beyond that taught in the previous courses. Students will be able to analyse and design current and future cryptographic methods. They also have an awareness of the methodology and power of various attack scenarios.

Contents
An introduction to modern methods of symmetric and asymmetric cryptography is provided. For this purpose, an attacker model is defined and the security of the presented encryption, hash and signature methods is proven under well-defined complexity measures in this attacker model.

Topic Overview:

- Secure encryption against KPA, CPA and CCA attackers
- Pseudo-random functions and permutations
- Message Authentication Codes
- Collision-resistant hash functions
- Block ciphers
- Construction of random number generators
- Diffie-Hellman key exchange
- Trapdoor one-way permutations
- Public key encryption: RSA, ElGamal, Goldwasser-Micali, Rabin, Paillier
- One-way signatures
- Signatures from collision-resistant hash functions
- Random Oracle Model

Teaching methods
Lecture and exercises

Examination forms
Written module final exam (120 minutes)

Requirements for the award of credits
Passed written final module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

8/150: B.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

8/149: B.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

8/99: M.Sc. IT-Sicherheit/ Netze und Syteme [PO 22]

8/96: M.Sc. IT-Sicherheit/ Netze und Syteme [PO 20]

8/97: M.Sc. Computer Science

8/105: M.Sc. Angewandte Informatik

Module title: Mathematics for Modeling and Data Analysis

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Summer Semester	1 semester

Courses	Contact time	Self-study	Group size
Mathematics for Modeling and Data Analysis (211047)	60 h	90 h	30 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Laurenz Wiskott
Lecturers: Prof. Dr. Laurenz Wiskott

Module application
B.Sc. Angewandte Informatik

M.Sc. Computer Science

Precognitions
Basic knowledge of calculus and linear algebra

Learning goals
After the successful completion of this course the students
- know the material covered in this course, see Content,
- do have an intuitive understanding of the basic concepts and can work with that,
- can communicate about all this in English

Contents
This course covers mathematical methods that are relevant for modeling and data analysis.
Particular emphasis is put on an intuitive understanding as is required for a creative command of mathematics.
The following topics are covered:
- Functions and how to visualize them
- Vector spaces
- Matrices as transformations
- Systems of linear differential equations
- Qualitative analysis of nonlinear differential equations
- Bayesian theory
- Markov chains

Teaching methods
This course is given with the flipped/inverted classroom concept. First, the students work through online material by themselves. In the lecture time slot we then discuss the material, find connections to other topics, ask questions and try to answer them. In the tutorial time slot the newly acquired knowledge is applied to analytical exercises and thereby deepened. I encourage all students to work in teams during self-study time as well as in the tutorial.

Examination forms
Written (digital) final module exam (90 minutes).

Requirements for the award of credits
Passed written Exam

Significance of the grade for the final grade (for a total of 120 ECTS)

5/168: B.Sc. Angewandte Informatik [PO 22]

5/170: B.Sc. Angewandte Informatik [PO 20]

5/97: M.Sc. Computer Science

Module title: Quantum Information and Computation

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Quantum Information and Computation (212011)	60 h	90 h	40 participants

Teaching language	Requirements for participation
German or English (depends on audience)	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Michael Walter
Lecturers: Prof. Dr. Michael Walter

Module application
B.Sc. Informatik

B.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. Angewandte Informatik

M.Sc. IT-Sicherheit/ Informationstechnik (On request)

M.Sc. IT-Sicherheit/ Netze und Systeme (On request)

M.Sc. Computer Science

Precognitions
Familiarity with linear algebra (in finite dimensions) and
probability (with finitely many outcomes) at the level of a first Bachelor's course; we will briefly remind you of the more difficult bits in class. In addition, some mathematical maturity, since we will discuss precise mathematical statements and rigorous proofs. No background in physics is required.

Learning goals
You will learn fundamental concepts, algorithms, and results in quantum information and computation. After successful completion of this course, you will know the theoretical model of quantum information and computation, how to generalize computer science concepts to the quantum setting, how to design and analyze quantum algorithms and protocols for a variety of computational problems, and how to prove complexity theoretic lower bounds. You will be prepared for an advanced course or a research or thesis project in this area.

Contents
This course will give an introduction to quantum information and quantum computation from the perspective of theoretical computer science.
Topics to be covered will likely include:

- Fundamentals of quantum computing: quantum bits, states and operations
- The power of quantum entanglement: nonlocal games
- Entanglement as a resource: superdense coding and teleportation
- Quantum circuit model of computation
- Quantum computing with oracles: Deutsch-Jozsa, Bernstein-Vazirani, Simon
- Quantum Fourier transform and phase estimation
- Shor\'s factoring algorithm
- Grover\'s search algorithm and beyond: how to solve SAT on a quantum computer?
- From no cloning to quantum money: a peek at quantum cryptography

The course should be of interest to students of computer science, mathematics, physics, and related disciplines. Students interested in a BSc or MSc project in quantum information, computing, cryptography, etc. are particularly encouraged to participate.

Teaching methods
Lecture with Exercise

Examination forms
Final written module exam (180 minutes)

Requirements for the award of credits
Passed written exam

Significance of the grade for the final grade (for a total of 120 ECTS)

5/158: B.Sc. Informatik [PO 22]

5/165: B.Sc. Informatik [PO 20]

5/150: B.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/149: B.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/105: M.Sc. Angewandte Informatik

5 /91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/ 99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/97: M.Sc. Computer Science

Module title: Software Languages

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Software Languages (212034)	60 h	90 h	25 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Thorsten Berger
Lecturers: Prof. Dr. Thorsten Berger

Module application
M.Sc. Computer Science

Precognitions

- Object-Oriented Programming
- Theoretical Computer Science (especially formal languages, grammars, set notation)
- Software Engineering
- Functional Programming

Learning goals
Knowledge and understanding

- explain core concepts for engineering software languages, e.g., meta-modeling, abstract syntax, concrete syntax, static semantics, dynamic semantics, model/program transformation
- explain linguistic architectures for software languages (e.g., meta-modeling hierarchy)
- explain how domain-specific languages can be realized within a contemporary language workbench

Competence and skills

- engineer domain-specific languages for a specific engineering problem
- define abstract syntax by modeling domains and creating meta-models
- define static semantics with syntactic constraints or type systems
- define concrete syntax with Xtext, Sirius, or a comparable technology
- define dynamic semantics with model transformations (compilation) or interpretation
- reason about configuration spaces expressed in variability models

Judgement and approach

- identify use-cases and the potential of using DSLs for a given domain/problem

select and justify appropriate language technology for a given domain/problem

Contents
The course teaches the theory and the pragmatics of using and developing high-level software languages (Domain-Specific Languages, or DSLs) for the effective production of quality software. This includes methods, design patterns, guidelines, and testing practices for defining concrete syntax, abstract syntax, and semantics of languages. The course attempts to be close to technology, while covering multiple paradigms and solutions.

We cover:

- Domain analysis of a problem domain and designing meta-models
- Engineering external and internal DSLs (we focus mostly on external DSLs)
- Designing the concrete syntax of languages

- Implementation of language semantics using declarative and imperative transformations, code generators and interpreters, in various scenarios, e.g., from text to models, from models to text, involving XML, database, etc.
- Implement declarative constraints and type rules for DSLs
- Testing and quality assurance of language implementations
- Model-driven engineering, especially engineering of highly configurable systems or software product lines (we focus on feature modeling syntax and semantics)

Teaching methods
The teaching of this course consists of different forms: lectures, interactive quizzes, group work, group supervision, and practical assignments.

Examination forms
Oral exam (20-30 minutes) or written exam (120 minutes), depending on the number of participants. Will be announced at the beginning of the course.

Requirements for the award of credits
Active participation in the group project and passed final module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

Module title: Theory of machine learning

Modul code	Credit points	Workload	Term	Frequency	Duration
	9 CP	270 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Theorie des maschinellen Lernens (211052)	90 h	180 h	20 participants

Teaching language	Requirements for participation
German	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Asja Fischer
Lecturers: Prof. Dr. Asja Fischer

Module application
M.Sc. Angewandte Informatik

M.Sc. Computer Science

Precognitions

Learning goals
Students are familiarised with mathematical models for machine learning. They acquire the ability to evaluate and compare learning algorithms according to the degree to which they achieve (precisely described) success criteria. They acquire techniques both for designing efficient learning algorithms and for proving the inherent hardness of a problem.
After the successful completion of the module

- students know the most important learning machines (such as Support Vector Machines and related models),
- students understand the difference between empirical and real error rate and know techniques to deal with the problem of overfitting the data (with a model that is too complex),
- can differentiate between uniform and non-uniform learnability of a Hypothesis class and know the appropriate theories and learning rules.

Contents
The subject of the lecture is the statistics-based theory of machine learning. In particular, the method of structured risk minimisation is taught as well as the statistical theorems on which it is based. Techniques for designing efficient learning algorithms are discussed as well as information- or computational-theoretic barriers that make certain learning problems appear to be inefficiently solvable.

Teaching methods
Lecture with exercise

Examination forms
Final oral module exam (30 minutes)

Requirements for the award of credits
Passed final oral module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

9/105: M.Sc. Angewandte Informatik

9/97: M.Sc. Computer Science

Module title: Deterministic Network Calculus

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Deterministic Network Caluculus (211054)	60 h	90 h	participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Steffen Bondorf
Lecturers: Prof. Dr. Steffen Bondorf

Module application
M.Sc. Computer Science

M.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. Angewandte Informatik

Precognitions
Mathematics (functional analysis), computer networks / distributed systems

Learning goals
Upon successful completion of the module, students will be able to,

- model complex, networked systems as deterministic queueing systems,
- perform worst-case performance analyses of existing systems or models,
- understand the challenges of performance dimensioning of planned systems, and explain how central mechanisms in computer networks work using network calculus
- differentiate the presented methods from each other and apply them to scientific questions.

Contents
Distributed systems are ubiquitous nowadays and their interconnectedness is fundamental for the continuous distribution and thus availability of data. The provision of data in real time is one of the most important non-functional aspects that safety-critical networks must ensure. The formal verification of data communication with respect to worst-case deadlines is fundamental for the certification of newly developed x-by-wire systems. This verification allows the take-off of aeroplanes, the steering of cars without mechanical connection and the operation of safety-critical industrial plants. Therefore, several methods for worst-case modelling and analysis of real-time systems have been developed. One of them is Deterministic Network Calculus (DNC), a versatile technique that can be used in various areas such as packet switching, task scheduling, system on chip, software defined networks, data centre networks and network virtualisation. DNC is a method for deriving deterministic bounds on two of the most prioritised performance metrics in communication systems:

- the end-to-end delay of data flows and
- the amount of memory a server needs to buffer all in

Teaching methods
Lecture with Exercise

Examination forms
Final oral module exam (30 minutes)

Requirements for the award of credits
Passed final oral module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/105: M.Sc. Angewandte Informatik

Module title: Energy-Aware Computing Systems

Modul code	Credit points	Workload	Term	Frequency	Duration
	6 CP	180 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Energy-Aware Computing Systems (212030)	60 h	120 h	20 participants

Teaching language	Requirements for participation
English	none

Module coordinators and Lecturers
Module coordinators: Prof. Dr.-Ing. Timo Hönig
Lecturers: Prof. Dr.-Ing. Timo Hönig

Module application

M.Sc. Computer Science

M.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. Angewandte Informatik

Precognitions

Learning goals
Students who have successfully attended the lecture and exercises have followed the learning objectives and have acquired the competence listed below. Students can

- understand the importance of electrical energy as an operating resource to computing systems
- make trade-off decisions in regard to efficient system design (i.e., power demand vs. performance), in particular of operating systems
- model the energy demand for individual synchronous and asynchronous operations
- apply strategies to reduce the energy demand for software activities based on specific hardware properties (i.e., sleep states)
- analyse software for critical sections that cause high energy demand

Contents
Electrical energy is the single most important operating resource for computer systems. Although the energy demand of computers is an invisible system property by itself, the impact of energy demand is omnipresent and obvious in various forms of appearance. Sudden system failures (i.e., system breakdowns) and recurrent standard system operations (i.e., power management) serve as practical examples. The lecture discusses the design of energy-aware computing systems and focuses on the following topics:

- power and energy management
- energy accounting
- energy demand analysis
- energy-aware operating-system architecture
- hardware power management (i.e., DVFS, throttling, sleep states)
- thermal management
- storage and file systems
- memory management
- network, wireless, and protocols
- energy-aware server/cluster
- compiler optimisations and code transformation
- display technology
- electricity grid

The lecture is linked to the exercises through research papers. The students read the papers in preparation for lectures. From there, the research papers are the basis for discussion and build the starting point for the

assignments of the exercises. As part of the exercises, the students apply concepts and strategies from the research papers to systems and evaluate the impact on the system's energy efficiency.

Teaching methods
The lecture is held with a seminar character. Research papers on energy-aware computing and system design are prepared by the students and discussed and analysed during the sessions. Additionally, the lecture imparts theoretical knowledge of fundamental concepts for the individual topics.

As part of the exercises, the students apply their acquired knowledge by adapting system software and system configurations to improve energy efficiency. They analyse the results by conducting performance and energy-demand evaluations.

Examination forms
Final oral module exam (30 minutes)

Requirements for the award of credits

Passed final module exam and successful participation in the exercises.

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Informatik

6/105: M.Sc. Angewandte Informatik

6/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

6/84: M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

Module title: High-Performance Computing on Clusters

Modul code	Credit points	Workload	Term	Frequency	Duration
	6 CP	180 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
High-Performance Computing on Clusters (127511)	60 h	120 h	50 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Andreas Vogel
Lecturers: Prof. Dr. Andreas Vogel

Module application
M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. Bauingenieurwesen

M.Sc. Computational Engineering

Precognitions

Learning goals
After successfully completing the module the students
- are enabled to design and create programs for parallel computing clusters
- can critically evaluate distributed-memory systems and programming patterns
- can assess the mathematical properties of iterative solvers and their scalability.

Contents
The lecture deals with the parallelization on cluster computers. Distributed-memory programming concepts (MPI) are introduced and best-practice implementation is presented based on applications from scientific computing including the finite element method and machine learning. Special attention is paid to scalable solvers for systems of equations on distributed memory systems, focusing on iterative schemes such as simple splitting methods (Richardson, Jacobi, Gauß-Seidel, SOR), Krylov-methods (Gradient descent, CG, BiCGStab) and, in particular, the multigrid method. The mathematical foundations for iterative solvers are reviewed, suitable object-oriented interface structures are developed and an implementation of these solvers for modern parallel computer architectures is developed. Numerical experiments and self-developed software implementations are used to discuss and illustrate the theoretical results.

Teaching methods
Beamer, computer lab, numerical experiments

Examination forms
Written module final exam (120 minutes)

Requirements for the award of credits
Passed written Exam

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: High-Performance Computing on Multi- and Manycore Processors

Modul code	Credit points	Workload	Term	Frequency	Duration
	6 CP	180 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
High-Performance Computing on Multi- and Manycore Processors (126509)	60 h	120 h	40 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Andreas Vogel
Lecturers: Prof. Dr. Andreas Vogel

Module application
M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. Bauingenieurwesen

M.Sc. Computational Engineering

Precognitions

Learning goals
After successfully completing the module, the students
- are enabled to design and create programs for multi- and manycore processors,
- can critically evaluate multi-threaded programs and shared-memory access patterns,
- are able to survey advanced scientific topics independently and present their findings.

Contents
The lecture addresses parallelization on multicore processors. Thread-based programming concepts and techniques, including pthreads, C++11 threads, OpenMP and SYCL, are introduced and best practices are highlighted using applications from scientific computing.
An overview of the relevant hardware aspects including multicore architectures and memory hierarchies is provided. An in-depth introduction to multi-threaded programming on multicore systems with special emphasis on shared-memory parallelization is given and parallelization patterns, thread management and memory access strategies are discussed.
In hands-on sessions, programming exercises are used to discuss and illustrate the presented content.

Teaching methods
Lecture with Exercise

Examination forms
Written module final exam (120 minutes).

Requirements for the award of credits

Passed written module final exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: Operating-System Concepts and Implementations

Modul code	Credit points	Workload	Term	Frequency	Duration
	6 CP	180 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Operating-System Concepts and Implementation	60 h	120 h	20 participants

Teaching language	Requirements for participation
English	none

Module coordinators and Lecturers
Module coordinators: Prof. Dr.-Ing. Timo Hönig
Lecturers: Prof. Dr.-Ing. Timo Hönig

Module application
M.Sc. Computer Science

M.Sc. IT-Sicherheit/ Netze und Systeme

M.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. Angewandte Informatik

Precognitions

Learning goals
The main learning outcome is a thorough understanding of different operating-system concepts and architectures (e.g., library, monolith, microkernel) and key operating-system components and subsystems. For example, the understanding and practical application of:

- interrupt concepts and implementations
- synchronisation concepts
- pseudo-parallelism in operating systems (e.g., continuations, coroutines, threads, context switches)
- scheduler dimensions of an operating system (e.g., fairness, throughput, preemptive vs. non-preemptive)
- inter-process communication implementations
- memory management and the corresponding memory protection/isolation mechanisms

Orthogonally, important learnings are social skills (e.g., cooperative work in small groups) and the confident use of common software and operating-system development tools (e.g., compilers and debugger).

Contents
Building upon the basic knowledge of operating systems and their functionality, this lecture deepens the understanding of fundamental operating-system concepts, architectures, and their implementations. This includes different operating-system architectures for general-purpose operating systems (e.g., Linux, Windows) and special-purpose operating systems (e.g., for embedded or automotive appliances).

Operating-system concepts and components, for example, are:

- memory management
- interrupt subsystem
- input/output subsystem
- processes and threads
- scheduler subsystem

- filesystem implementations
- driver model
- multicore subsystem

Teaching methods
The lecture imparts the theoretical knowledge about operating-system concepts and discusses the respective advantages and disadvantages.

The understanding of those concepts is deepened by implementing operating-system components in small groups as part of the corresponding exercises. This results in a fully functional operating system executable on modern hardware (e.g., x86-64, ARM).

Examination forms
Final oral module exam (30 minutes)

Requirements for the award of credits
Passed final oral module examination and successful completion of the exercises.

Significance of the grade for the final grade (for a total of 120 ECTS)
6/97: M.Sc. Computer Science

Module title: Real Time Networks and Systems

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Real Time Networks and Systems (212032)	45 h	105 h	participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr.-Ing. Steffen Bondorf
Lecturers: Prof. Dr.-Ing. Steffen Bondorf

Module application
M.Sc. Computer Science

Precognitions

Learning goals
Students will gain a deep knowledge of modelling and analysis of (hard) real-time systems, in particular deterministic queuing theory (networking) as well as scheduling (networks, systems).

Contents
The scheduling of shared resources (e.g. forwarding in networks, CPU time of local systems) plays a central role in the performance of modern systems. In real-time systems, requirements for the duration of data transmissions or calculations are set in the form of (hard) deadlines. Compliance with these deadlines must be proven or (constructively) guaranteed. For this purpose, the lecture presents modelling and analysis methods for (distributed) real-time systems.

Teaching methods
The lecture is held as a seminar-style class, the practical exercises on the computer may include other forms of teaching such as group and project work.

Examination forms
Oral final module exam (30 minutes)

Requirements for the award of credits
Passed final module exam and successful participation in exercises

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

Module title: Geometric Algorithms

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Geometrische Algorithmen (211056)	60 h	90 h	20 participants

Teaching language	Requirements for participation
German	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Maike Buchin
Lecturers: Prof. Dr. Maike Buchin

Module application
M.Sc. Computer Science

M.Sc. Angewandte Informatik

Precognitions
Mandatory: Basic knowledge of algorithms and data structures. Recommended: Fundamentals of stochastics.

Learning goals
After successful completion of the module

- students know basic geometric algorithms and data structures
- can analyse and design algorithms according to the sweep paradigm
- can design and analyse incremental algorithms, especially randomised incremental algorithms
- can analyse and design geometric algorithms according to the divide-and-conquer principle
- students can select suitable data structures for range queries.

Contents
Algorithmic geometry deals with the design and analysis of algorithms and data structures for geometric problems. For this purpose, we first look at some basic problems, such as calculating the convex hull of a set of points, the intersections of a set of distances or a triangulation of a simple polygon. We then look at algorithms for computing well-known geometric structures, such as the Voronoi diagram, the Delaunay triangulation and arrangements. We also look at data structures for efficient queries on geometric data, such as rangetrees, kd-trees and quadtrees. Three main types of algorithms are used: incremental, divide-and-conquer, and sweep. Some of these occur as randomised algorithms.

Teaching methods
Lecture as combined slide and blackboard lecture and associated exercises.

Examination forms
Final oral module exam (30 minutes)

Requirements for the award of credits
Passed final oral module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/105: M.Sc. Angewandte Informatik

5/97: M.Sc. Computer Science

Module title: Computational complexity theory

Modul code	Credit points	Workload	Term	Frequency	Duration
	9 CP	270 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Computational complexity theory (211028)	90 h	180 h	participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Thomas Zeume
Lecturers: Prof. Thomas Zeume

Module application
M.Sc. Computer Science

M.Sc. IT-Sicherheit/Informationstechnik

M.Sc. IT-Sicherheit/Netze und Systeme

M.Sc. Angewandte Informatik (until SS 23)

Precognitions
Knowledge from a basic course in theoretical computer science (basics of complexity theory including NP-completeness and reductions) is expected.

Learning goals
Students learn to classify algorithmic problems in terms of their complexity and thus identify suitable algorithmic techniques for their solution. In particular, they can apply algorithmic methods for NP-complete problems. They can deal with different computational models and are able to prove simple statements about them. They learn to evaluate their own and other people's approaches to solutions in discourse.

Contents
Complexity theory examines and classifies computational problems in terms of their algorithmic difficulty. The aim is to determine the inherent resource consumption in terms of various resources such as computing time or memory space, and to group problems with similar resource consumption into complexity classes. The best-known complexity classes are certainly P and NP, which comprise the problems that can be solved or verified in polynomial time. The question of whether P and NP are different is considered one of the most important open questions in theoretical computer science, and even in mathematics. However, P and NP are only two examples of complexity classes. Other classes arise, among others, in the investigation of the required memory space, the efficient parallelisability of problems, the solvability by random algorithms, and the approximate solvability of problems. The lecture aims to give a broad overview of the basic concepts and results of complexity theory:

- Classical results for space and time complexity classes: e.g., the correspondence between games and memory constraints, the proof that with more space or time, more problems can be solved, other basic relations between time and space-based classes, and the complexity world between NP and PSPACE.
- Basic features of complexity theory of parallel, random and approximate algorithms.
- Introduction to selected recent topics: Complexity theory of interactive computing, probabilistic reasoning and fine-grained complexity.

Teaching methods
Lecture with exercise

Examination forms
Oral module final exam (20-30 minutes)

Requirements for the award of credits
Passed final oral module exam

Significance of the grade for the final grade (for a total of 120 ECTS)

9/97: M.Sc. Computer Science

9/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

9/84: M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

9/99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO22]

9/96: M.Sc. IT-Sicherheit/ Netze und Systeme [PO20]

Module title: Quantum Information and Computation

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Quantum Information and Computation (212011)	60 h	90 h	40 participants

Teaching language	Requirements for participation
German or English (depends on audience)	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Michael Walter
Lecturers: Prof. Dr. Michael Walter

Module application
B.Sc. Informatik

B.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. Angewandte Informatik

M.Sc. IT-Sicherheit/ Informationstechnik (On request)

M.Sc. IT-Sicherheit/ Netze und Systeme (On request)

M.Sc. Computer Science

Precognitions
Familiarity with linear algebra (in finite dimensions) and
probability (with finitely many outcomes) at the level of a first Bachelor's course; we will briefly remind you of the more difficult bits in class. In addition, some mathematical maturity, since we will discuss precise mathematical statements and rigorous proofs. No background in physics is required.

Learning goals
You will learn fundamental concepts, algorithms, and results in quantum information and computation. After successful completion of this course, you will know the theoretical model of quantum information and computation, how to generalize computer science concepts to the quantum setting, how to design and analyze quantum algorithms and protocols for a variety of computational problems, and how to prove complexity theoretic lower bounds. You will be prepared for an advanced course or a research or thesis project in this area.

Contents
This course will give an introduction to quantum information and quantum computation from the perspective of theoretical computer science.
Topics to be covered will likely include:

- Fundamentals of quantum computing: quantum bits, states and operations
- The power of quantum entanglement: nonlocal games
- Entanglement as a resource: superdense coding and teleportation
- Quantum circuit model of computation
- Quantum computing with oracles: Deutsch-Jozsa, Bernstein-Vazirani, Simon
- Quantum Fourier transform and phase estimation
- Shor\'s factoring algorithm
- Grover\'s search algorithm and beyond: how to solve SAT on a quantum computer?
- From no cloning to quantum money: a peek at quantum cryptography

The course should be of interest to students of computer science, mathematics, physics, and related disciplines. Students interested in a BSc or MSc project in quantum information, computing, cryptography, etc. are particularly encouraged to participate.

Teaching methods
Lecture with Exercise

Examination forms
Final written module exam (180 minutes)

Requirements for the award of credits
Passed written exam

Significance of the grade for the final grade (for a total of 120 ECTS)

5/158: B.Sc. Informatik [PO 22]

5/165: B.Sc. Informatik [PO 20]

5/150: B.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/149: B.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/105: M.Sc. Angewandte Informatik

5 /91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/ 99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/97: M.Sc. Computer Science

Module title: Blockchain Security and Privacy

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Blockchain security and privacy (212007)	60 h	90 h	40 participants

Teaching language	Requirements for participation
English	none

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Ghassan Karame
Lecturers: Prof. Dr. Ghassan Karame

Module application
M.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. IT-Sicherheit/ Netze und Systeme

M.Sc. Computer Science

Precognitions
none

Learning goals
Upon completion of this course, students are expected to be able to:

1. Reason about the security and privacy definitions of open payment systems.
2. Explain the security of PoW blockchains in light of the state of the art reported attacks.
3. Reason about possible network security and cryptographic countermeasures to deter attacks on blockchains.
4. Explain best security/privacy practices to strengthen the security of existing blockchains, and extract relevant lessons for the design of next-generation blockchain technologies.

Contents
The main objective of the course is to provide a comprehensive overview of the security and privacy of blockchain technologies.

Course participants will be also introduced to the basic security and privacy provisions of existing popular currencies, and will be exposed to the state-of the-art attacks and threats reported against existing systems/deployments. The participants will also reason on the effectiveness of combining network-level security primitives, with novel cryptographic primitives to deter attacks on payment systems.

Teaching methods
Lecture with Exercise

Examination forms
Written module final exam (120 minutes)

Requirements for the award of credits
Passed final written module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: Master IT-Sicherheit | Informationstechnik [PO 22]

5/84: Master IT-Sicherheit | Informationstechnik [PO 20]

5/99: Master IT-Sicherheit | Netze und Systeme [PO 22]

5/96: Master IT-Sicherheit | Netze und Systeme [PO 20]

5/97: Master Computer Science

Module title: Cryptography on hardware-based platforms

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Kryptographie auf hardwarebasierten Plattformen (212019)	60 h	90 h	50 participants

Teaching language	Requirements for participation
German	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr.-Ing. Tim Güneysu
Lecturers: Prof. Dr.-Ing. Tim Güneysu

Module application

B.Sc. IT-Sicherheit/ Informationstechnik

B.Sc. Angewewandte Informatik [until WS 22/23]

M.Sc. IT-Sicherheit/ Netze und Systeme [until WS 22/23]

M.Sc. Computer Science

Precognitions

Learning goals
The students know the concepts of practical hardware development with abstract hardware description languages (VHDL) and the simulation of hardware circuits on FPGAs. They master standard techniques of hardware-related processor development and are able to implement symmetric and asymmetric crypto systems on modern FPGA systems.

Contents
Due to their complexity, cryptographic systems place high demands on small processors and embedded systems in particular. In combination with the demand for high data throughput at lowest hardware costs, fundamental problems arise for the developer, which are to be illuminated in this lecture. The lecture deals with the most interesting aspects of how to implement current cryptographic methods on practical hardware systems. Cryptosystems such as the block cipher AES, the hash functions SHA-1 as well as asymmetric systems RSA and ECC are dealt with. Furthermore, special hardware requirements such as the generation of true randomness (TRNG) and the use of Physically Unclonable Functions (PUF) are discussed. The efficient implementation of these cryptosystems, especially with regard to optimisation for high speed, is discussed on modern FPGAs and implemented in practical exercises using the hardware description language VHDL. A Moodle course is offered to accompany the lecture, which provides additional content as well as the practical exercises.

Teaching methods
Lecture with exercise

Examination forms
Written module final exam (120 minutes)

Requirements for the award of credits
Passed final written module exam; successful participation in the exercise can earn up to 10 percent bonus points, which can be credited to the result of the module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)
5/150: B.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/149: B.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/97: M.Sc. Computer Science

Module title: Cryptographic protocols

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Kryptographische Protokolle (211031)	60 h	90 h	participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Eike Kiltz
Lecturers: Prof. Dr. Eike Kiltz

Module application
M.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. IT-Sicherheit/ Netze und Systeme

M.Sc. Angewandte Informatik [until SS 23]

M.Sc. Computer Science

Precognitions
Content of the module Cryptography

Learning goals
- Deepening understanding in provable security
- Writing error-free security reductions
- New techniques for security proofs
- Learning advanced cryptographic constructions

Contents
The lecture deals with advanced cryptographic protocols and their applications.

Topic overview:

- Game-based security definitions and proofs
- Bilinear maps
- Digital Signatures
- Identification Protocols
- Zero-Knowledge Proofs
- Identity-based Encryption
- CCA-secure encryption

Teaching methods
Lecture with exercise

Examination forms
Oral (30 minutes) or written final module examination (120 minutes), depending on the number of participants. Will be communicated at the beginning of the course.

Requirements for the award of credits
Passed oral or written final module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/96: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 20]

5/97: M.Sc. Computer Science

Module title: Microarchitectural Attacks and Defenses

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Microarchitectural Attacks and Defenses (212064)	45 h	105 h	30 participants

Teaching language	Requirements for participation
English	none

Module coordinators and Lecturers
Module coordinators: Prof. Yuval Yarom
Lecturers: Prof. Yuval Yarom

Module application
M.Sc. ITS - Informationstechnik

M.Sc. ITS - Netze und Systeme

M.Sc. Computer Science

Precognitions
The course assumes that students can program in C or learn the language as they go. They need enough experience to be able to program on remote machines, using SSH. Basic understanding of how computers work, assembly language, and the role of the operating system is required. Understanding of basic concepts in computer security (security domains, vulnerabilities, etc.) and familiarity with basic cryptography (AES, RSA, ECC) is helpful.

Learning goals
- Diagnose microarchoitectural vulnerabilities
- Assess software for resilience against microarchitectural vulnerabilities
- Design and program proof-of-concept exploits of vulnerable software and hardware
- Design and implement countermeasures for software executing on vulnerable hardware

Contents
The course covers the area of microarchitectural attacks and defences. It starts with cache attacks, covering the main techniques (Prime+Probe, Evict+Time, and Flush+Reload). Building on this basis it explores variants of the attacks targetting other storage elements as well as attacks that exploits bandwidth limitations. In parallel with exploring these attacks, the course will describe various countermeasures, with special focus on constant-time programming. The course then switches to speculative execution attacks, identifying and classifying the various attacks, defences, and counter-attacks. The course further covers several related attacks, including Rowhammer and voltage- and frequency-based attacks. Additionally, the course pays special attention to attack scenarios exploring, in particular, attacks on the operating system kernel, web-based and other remote attacks, and attacks on trusted execution environments. The course puts special focus on practical implementation of both attack and defence techniques.

Teaching methods
Lecture with excercise

Examination forms
Project work with assignments.

Requirements for the award of credits
Passed project work with assignments.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. ITS - Informationstechnik [PO 22]

5/84: M.Sc. ITS - Informationstechnik [PO 20]

5/99: M.Sc. ITS - Netze und Systeme [PO 22]

5/96: M.Sc. ITS - Netze und Systeme [PO 20]

5/97: M.Sc. Computer Science

Module title: Quantum Cryptography

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	siehe Prüfungsordnung	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Quantum Cryptography (212016)	60 h	90 h	participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Michael Walter
Lecturers: Prof. Michael Walter
Dr. Giulio Malavolta

Module application
M.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. IT-Sicherheit/ Netze und Systeme

M.Sc. Computer Science

Precognitions
None

Learning goals
You will learn fundamental concepts, algorithms, protocols, and results in quantum (and quantum-resistant) cryptography. After successful completion of this course, you will know how to generalize cryptographic concepts to the quantum setting, how quantum algorithms can attack well-known cryptographic protocols, and how to design and analyze classical and quantum protocols for protecting classical and quantum data against quantum adversaries. You will be prepared for a research or thesis project in this area.

Contents
This course will give an introduction to the interplay of quantum information and cryptography, which has recently led to much excitement and insights – including by researchers at CASA right here on our very own campus. We will begin with a brief introduction to both fields and discuss in the first half of the course how quantum computers can attack classical cryptography and how to overcome this challenge – either by protecting against the power of quantum computers or by leveraging the power of quantum information. In the second half of the course, we will discuss how to generalize cryptography to protect quantum data and computation.

Topics to be covered will likely include:

* Basic quantum computing

* Basic cryptography

* Quantum attacks on classical cryptography

* Quantum random oracles and compressed oracle technique

* Quantum-resistant cryptography in light of the NIST competition

* Classical vs quantum information

* Quantum money

* Quantum key distribution

* Quantum complexity theory

* Quantum pseudorandomness

* From classical to quantum fully homomorphic encryption

* Classical verification of quantum computation

* Quantum rewinding

This course should be of interest to students of computer science, mathematics, physics, and related disciplines. Students interested in a Master's project in quantum or quantum-resistant cryptography, quantum information, quantum computing, and similar are particularly encouraged to participate.

Teaching methods
Lecture with exercise

Examination forms
Oral (30 minutes) or written final module examination (120 minutes), depending on the number of participants. Will be communicated at the beginning of the course.

Requirements for the award of credits
Passed oral or written final module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91 M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/84 M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/99 :M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/96 :M.Sc. IT-Sicherheit/ Netze und Systeme [PO 20]

5/97: M.Sc. Computer Science

Module title: Quantum Information and Computation

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Quantum Information and Computation (212011)	60 h	90 h	40 participants

Teaching language	Requirements for participation
German or English (depends on audience)	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Michael Walter
Lecturers: Prof. Dr. Michael Walter

Module application
B.Sc. Informatik

B.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. Angewandte Informatik

M.Sc. IT-Sicherheit/ Informationstechnik (On request)

M.Sc. IT-Sicherheit/ Netze und Systeme (On request)

M.Sc. Computer Science

Precognitions
Familiarity with linear algebra (in finite dimensions) and
probability (with finitely many outcomes) at the level of a first Bachelor's course; we will briefly remind you of the more difficult bits in class. In addition, some mathematical maturity, since we will discuss precise mathematical statements and rigorous proofs. No background in physics is required.

Learning goals
You will learn fundamental concepts, algorithms, and results in quantum information and computation. After successful completion of this course, you will know the theoretical model of quantum information and computation, how to generalize computer science concepts to the quantum setting, how to design and analyze quantum algorithms and protocols for a variety of computational problems, and how to prove complexity theoretic lower bounds. You will be prepared for an advanced course or a research or thesis project in this area.

Contents
This course will give an introduction to quantum information and quantum computation from the perspective of theoretical computer science.
Topics to be covered will likely include:

- Fundamentals of quantum computing: quantum bits, states and operations
- The power of quantum entanglement: nonlocal games
- Entanglement as a resource: superdense coding and teleportation
- Quantum circuit model of computation
- Quantum computing with oracles: Deutsch-Jozsa, Bernstein-Vazirani, Simon
- Quantum Fourier transform and phase estimation
- Shor\'s factoring algorithm
- Grover\'s search algorithm and beyond: how to solve SAT on a quantum computer?
- From no cloning to quantum money: a peek at quantum cryptography

The course should be of interest to students of computer science, mathematics, physics, and related disciplines. Students interested in a BSc or MSc project in quantum information, computing, cryptography, etc. are particularly encouraged to participate.

Teaching methods
Lecture with Exercise

Examination forms
Final written module exam (180 minutes)

Requirements for the award of credits
Passed written exam

Significance of the grade for the final grade (for a total of 120 ECTS)

5/158: B.Sc. Informatik [PO 22]

5/165: B.Sc. Informatik [PO 20]

5/150: B.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/149: B.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/105: M.Sc. Angewandte Informatik

5 /91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/ 99: M.Sc. IT-Sicherheit/ Netze und Systeme [PO 22]

5/97: M.Sc. Computer Science

Module title: Autonomous Vehicles and Artificial Intelligence

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Autonomous Vehicles and Artificial Intelligence (211044)	60 h	90 h	25 participants

Teaching language	Requirements for participation
English	none

Module coordinators and Lecturers
Module coordinators: Prof. Thorsten Berger
Lecturers: Prof. Dr. Thorsten Berger, Dr. Sven Peldszus

Module application
B.Sc. Informatik [until SS 23]

B.Sc. IT-Sicherheit [until SS 23]

B.Sc. Angewandte Informatik [until SS 23]

M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. IT-Sicherheit/Informationstechnik

M.Sc. IT-Sicherheit/Netze und Systeme [until SS 23]

Precognitions
The Software Engineering lecture or a comparable course, Programming experiences e.g. as part of other courses.

Learning goals
- Understanding requirements on autonomous vehicles
- Understanding the architecture of autonomous vehicles
- Ability to build a self-driving car with ROS2
- Understanding and applying quality assurance for autonomous vehicles

Contents
Autonomous driving is the future of individual mobility and all major manufacturers are working on fully autonomous vehicles. While there are robust and good solutions for the individual problems in autonomous driving, the main challenge lies in their integration. Altogether, an autonomous vehicle’s software is the biggest problem. Therefore, the key in self-driving vehicles is about getting the software right. In this course, we will investigate the different aspects of self-driving vehicles as well as the importance and application of artificial intelligence in this domain. The course will primarily focus on the following topics:
- Requirements on autonomous vehicles
- Architecture of autonomous vehicles
- Operation systems and frameworks for robotic systems
- Specification and Implementation of autonomous vehicles based on ROS2
- Artificial intelligence for autonomous vehicles
- Simulation of autonomous vehicles ∙ Localization and perception
- Mission planning
- Quality assurance for autonomous vehicles In the course’s lecture, we provide the required theoretical background and practically apply the course’s content in exercises by building a self-driving robot.

Teaching methods
Lecture with exercises

Examination forms Final oral module exam (30 minutes)
Requirements for the award of credits Passed final module exam and successful participation in the exercises
Significance of the grade for the final grade (for a total of 120 ECTS) 5/97: M.Sc. Computer Science 5/105: M.Sc. Angewandte Informatik 5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO 22] 5/84: M.Sc. IT-Sicherheit/Informationstechnik [PO 20]

Module title: Autonomous Vehicles and Artificial Intelligence Lab

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150h	see examination regulations	Winter Semester	1 semester

Courses	Contact time	Self-study	Group size
Autonomous Vehicles and Artificial Intelligence Lab (212035)	60h	90h	participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr Thorsten Berger
Lecturers: Prof. Dr. Thorsten Berger
Dr. Sven Peldszus

Module application

M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. IT-Sicherheit/Informationstechnik

Precognitions
Recommended:

- Autonomous Vehicles and Artificial Intelligence (Lecture, 211044)
- Programming experience in Python or C++ (e.g., as part of other courses)
- Software Engineering (Lecture, 212000)

Learning goals
Knowledge

- being able to explain relevant theoretical knowledge of artificial intelligence and autonomous vehicles

Skill and abilities

- define and validate requirements on autonomous vehicles

- instantiate an architecture for autonomous vehicles

- build a self-driving car with ROS2

- manage and integrate artificial intelligence in complex software-intensive systems

- organize a team and its development process for a complex software-intensive system

- perform quality assurance for autonomous vehicles

- document the development process and create the artifacts that would be needed for a certification according to the ISO Road vehicles standards

- communicate with group members and stakeholders professionally (speech and writing)

Contents

Autonomous driving is the future of individual mobility, and all major manufacturers are working on fully autonomous vehicles. While there are robust and well-researched solutions for the individual problems in autonomous driving, the main challenge lies in their integration. Altogether, an autonomous vehicle’s software is the biggest problem. Therefore, the key in self-driving vehicles is about getting the software right.

In this course, we will practically explore the different aspects of self-driving vehicles as well as the importance and application of artificial intelligence in this domain by developing a self-driving race car. To this end, the participants will work with ROS2-based model cars. This aims to provide the students with practical experience in developing an autonomous race car and organizing the development process.

Teaching methods

The main learning sequence in the course is a major practical project. The project is carried out in groups that iteratively develop an autonomous race car, applying and consolidating theoretical knowledge about autonomous driving and software development. To support learning, the Autonomous Vehicles and Artificial Intelligence Lab is based on seminar-style lectures that provide a platform for seeking feedback and more information. Based on the information gathered, students update and refine their solutions for an autonomous race car. Continuous reflection on practice and theory is facilitated by the ongoing production of a final project report in parallel with the development of the race car, in which students reflect on their own learning in the course, the way they and their team approach their development process, and their technical solutions. Students receive regular feedback and guidance to support their learning.

Examination forms

The final grade is determined based on the participation in the development of the self-driving race car, written project reports, the developed autonomous race car, and an oral presentation of the group results. Individual grades will be given by compiling the assessment of individual and group-based outcomes.

Requirements for the award of credits

Actively participate in the development of an autonomous race car, regularly attend seminar-style lectures, and successfully complete all deliverables.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/Informationstechnik [PO 20]

Module title: Foundations of Programming Languages, Verification, and Security

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Foundations of Programming Languages, Verification, and Security (211044)	60 h	90 h	20 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Dr. Roberto Blanco
Dr. Catalin Hritcu
Lecturers: Dr. Roberto Blanco
Dr. Catalin Hritcu

Module application
M.Sc. Computer Science

M.Sc IT-Sicherheit/Netze und Systeme

M.Sc. IT-Sicherheit/Informationstechnik

M.Sc. Mathematik

Precognitions
This advanced course for MSc and PhD students requires having attended the Proofs are Programs course or having a working knowledge of the contents of the Logical Foundations book (https://mpi-sp-pap-2023.github.io/book), including familiarity with logic, mechanized proofs, and functional programming in the Coq proof assistant.

Learning goals
After successful completion of this course, students will be able to

l understand how to define in Coq the syntax of simple programming languages: variants of a simple imperative language and the simply-typed lambda calculus;

l define the big-step and small-step operational semantics of such simple languages;

l formally define type systems for such languages as inductive relations;

l work out the metatheory of such languages, by proving results such as type soundness;

l understand the semantic foundations of Hoare Logic and Relational Hoare Logic;

l use Hoare Logic for verifying the correctness of simple imperative programs, both formally in Coq and informally on paper;

l understand the semantic foundations of Secure Information Flow Control and Noninterference.

l use Relational Hoare Logic for proving program equivalence as well as Noninterference of simple imperative programs;

Contents
Complex proofs on paper are difficult to write, check, and maintain. This holds not only for interesting proofs in mathematics, but also for complex formal proofs about interesting programs. For this reason, machine-checked proofs created with the help of interactive tools called proof assistants are gaining increased traction in academia and industry. Proof assistants have been used to prove the correctness and security of realistic compilers,

operating systems, cryptographic libraries, or smart contracts, and also to construct machine-checked proofs for challenging mathematical results.

This course will use the Coq proof assistant [2] to lay down the foundations of Programming Languages, Verification, and Security. The Coq proof assistant enables us to program formal proofs interactively and it machine-checks the correctness of the proofs along the way. We will use Coq to define the syntax and semantics of programming languages, to define type systems, and to prove theorems such as type soundness. We will also formalize Hoare Logic and Relational Hoare Logic in Coq and use them to prove the correctness and security of simple imperative programs. Finally, the course will introduce static and dynamic enforcement mechanisms for Secure Information Flow Control and Cryptographic Constant Time as well as their formal noninterference guarantees.

This hands-on course is based on the Programming Languages Foundations online textbook [1], which is itself formalized and machine-checked in the Coq proof assistant. The many exercises in each book chapter are to be solved weekly mostly in Coq, from easy exercises allowing the students to practice concepts from the lecture, building incrementally to slightly more interesting programs and proofs and also to various optional challenges.

Teaching methods
This course consists of lectures and weekly exercises, in which the students will solve problems using the Coq proof assistant for which they can get help from a tutor.

Examination forms
Written final exam (120 minutes) and exercise sheets.

Requirements for the award of credits
There will be a mandatory written final exam (120 minutes) that counts for 60% of the grade and weekly exercise sheets that have to be submitted on time and that count for 40% of the grade. We will also have an optional midterm exam that helps students practice for the final exam, but only counts for bonus points, up to 10% of the final grade. One can additionally get bonus points up to 5% of the final grade by solving all exercise sheets.

To pass the course and receive credit points one has to attend the final exam and the weighed sum of your scores including bonus points (which can add up to a maximum of 115%) has to be at least 50%.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/84: M.Sc. IT-Sicherheit/Informationstechnik [PO20]

5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO22]

5/96: M.Sc. IT-Sicherheit/Netze und Systeme [PO20]

5/99: M.Sc. IT-Sicherheit/Netze und Systeme [PO22]

Module title: High-Performance Computing on Clusters

Modul code	Credit points	Workload	Term	Frequency	Duration
	6 CP	180 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
High-Performance Computing on Clusters (127511)	60 h	120 h	50 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Andreas Vogel
Lecturers: Prof. Dr. Andreas Vogel

Module application
M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. Bauingenieurwesen

M.Sc. Computational Engineering

Precognitions

Learning goals
After successfully completing the module the students

- are enabled to design and create programs for parallel computing clusters
- can critically evaluate distributed-memory systems and programming patterns
- can assess the mathematical properties of iterative solvers and their scalability.

Contents
The lecture deals with the parallelization on cluster computers. Distributed-memory programming concepts (MPI) are introduced and best-practice implementation is presented based on applications from scientific computing including the finite element method and machine learning. Special attention is paid to scalable solvers for systems of equations on distributed memory systems, focusing on iterative schemes such as simple splitting methods (Richardson, Jacobi, Gauß-Seidel, SOR), Krylov-methods (Gradient descent, CG, BiCGStab) and, in particular, the multigrid method. The mathematical foundations for iterative solvers are reviewed, suitable object-oriented interface structures are developed and an implementation of these solvers for modern parallel computer architectures is developed. Numerical experiments and self-developed software implementations are used to discuss and illustrate the theoretical results.

Teaching methods
Beamer, computer lab, numerical experiments

Examination forms
Written module final exam (120 minutes)

Requirements for the award of credits
Passed written Exam

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: High-Performance Computing on Multi- and Manycore Processors

Modul code	Credit points	Workload	Term	Frequency	Duration
	6 CP	180 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
High-Performance Computing on Multi- and Manycore Processors (126509)	60 h	120 h	40 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Andreas Vogel
Lecturers: Prof. Dr. Andreas Vogel

Module application
M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. Bauingenieurwesen

M.Sc. Computational Engineering

Precognitions

Learning goals
After successfully completing the module, the students
- are enabled to design and create programs for multi- and manycore processors,
- can critically evaluate multi-threaded programs and shared-memory access patterns,
- are able to survey advanced scientific topics independently and present their findings.

Contents
The lecture addresses parallelization on multicore processors. Thread-based programming concepts and techniques, including pthreads, C++11 threads, OpenMP and SYCL, are introduced and best practices are highlighted using applications from scientific computing.
An overview of the relevant hardware aspects including multicore architectures and memory hierarchies is provided. An in-depth introduction to multi-threaded programming on multicore systems with special emphasis on shared-memory parallelization is given and parallelization patterns, thread management and memory access strategies are discussed.
In hands-on sessions, programming exercises are used to discuss and illustrate the presented content.

Teaching methods
Lecture with Exercise

Examination forms
Written module final exam (120 minutes).

Requirements for the award of credits

Passed written module final exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: Software Languages

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Software Languages (212034)	60 h	90 h	25 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Thorsten Berger
Lecturers: Prof. Dr. Thorsten Berger

Module application
M.Sc. Computer Science

Precognitions

- Object-Oriented Programming
- Theoretical Computer Science (especially formal languages, grammars, set notation)
- Software Engineering
- Functional Programming

Learning goals
Knowledge and understanding

- explain core concepts for engineering software languages, e.g., meta-modeling, abstract syntax, concrete syntax, static semantics, dynamic semantics, model/program transformation
- explain linguistic architectures for software languages (e.g., meta-modeling hierarchy)
- explain how domain-specific languages can be realized within a contemporary language workbench

Competence and skills

- engineer domain-specific languages for a specific engineering problem
- define abstract syntax by modeling domains and creating meta-models
- define static semantics with syntactic constraints or type systems
- define concrete syntax with Xtext, Sirius, or a comparable technology
- define dynamic semantics with model transformations (compilation) or interpretation
- reason about configuration spaces expressed in variability models

Judgement and approach

- identify use-cases and the potential of using DSLs for a given domain/problem

select and justify appropriate language technology for a given domain/problem

Contents
The course teaches the theory and the pragmatics of using and developing high-level software languages (Domain-Specific Languages, or DSLs) for the effective production of quality software. This includes methods, design patterns, guidelines, and testing practices for defining concrete syntax, abstract syntax, and semantics of languages. The course attempts to be close to technology, while covering multiple paradigms and solutions.

We cover:

- Domain analysis of a problem domain and designing meta-models
- Engineering external and internal DSLs (we focus mostly on external DSLs)
- Designing the concrete syntax of languages

- Implementation of language semantics using declarative and imperative transformations, code generators and interpreters, in various scenarios, e.g., from text to models, from models to text, involving XML, database, etc.
- Implement declarative constraints and type rules for DSLs
- Testing and quality assurance of language implementations
- Model-driven engineering, especially engineering of highly configurable systems or software product lines (we focus on feature modeling syntax and semantics)

Teaching methods
The teaching of this course consists of different forms: lectures, interactive quizzes, group work, group supervision, and practical assignments.

Examination forms
Oral exam (20-30 minutes) or written exam (120 minutes), depending on the number of participants. Will be announced at the beginning of the course.

Requirements for the award of credits
Active participation in the group project and passed final module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

Module title: Autonomous Vehicles and Artificial Intelligence

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Autonomous Vehicles and Artificial Intelligence (211044)	60 h	90 h	25 participants

Teaching language	Requirements for participation
English	none

Module coordinators and Lecturers
Module coordinators: Prof. Thorsten Berger
Lecturers: Prof. Dr. Thorsten Berger, Dr. Sven Peldszus

Module application
B.Sc. Informatik [until SS 23]

B.Sc. IT-Sicherheit [until SS 23]

B.Sc. Angewandte Informatik [until SS 23]

M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. IT-Sicherheit/Informationstechnik

M.Sc. IT-Sicherheit/Netze und Systeme [until SS 23]

Precognitions
The Software Engineering lecture or a comparable course, Programming experiences e.g. as part of other courses.

Learning goals
- Understanding requirements on autonomous vehicles
- Understanding the architecture of autonomous vehicles
- Ability to build a self-driving car with ROS2
- Understanding and applying quality assurance for autonomous vehicles

Contents
Autonomous driving is the future of individual mobility and all major manufacturers are working on fully autonomous vehicles. While there are robust and good solutions for the individual problems in autonomous driving, the main challenge lies in their integration. Altogether, an autonomous vehicle’s software is the biggest problem. Therefore, the key in self-driving vehicles is about getting the software right. In this course, we will investigate the different aspects of self-driving vehicles as well as the importance and application of artificial intelligence in this domain. The course will primarily focus on the following topics:
- Requirements on autonomous vehicles
- Architecture of autonomous vehicles
- Operation systems and frameworks for robotic systems
- Specification and Implementation of autonomous vehicles based on ROS2
- Artificial intelligence for autonomous vehicles
- Simulation of autonomous vehicles ∙ Localization and perception
- Mission planning
- Quality assurance for autonomous vehicles In the course’s lecture, we provide the required theoretical background and practically apply the course’s content in exercises by building a self-driving robot.

Teaching methods
Lecture with exercises

Examination forms Final oral module exam (30 minutes)
Requirements for the award of credits Passed final module exam and successful participation in the exercises
Significance of the grade for the final grade (for a total of 120 ECTS) 5/97: M.Sc. Computer Science 5/105: M.Sc. Angewandte Informatik 5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO 22] 5/84: M.Sc. IT-Sicherheit/Informationstechnik [PO 20]

Module title: Autonomous Vehicles and Artificial Intelligence Lab

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150h	see examination regulations	Winter Semester	1 semester

Courses	Contact time	Self-study	Group size
Autonomous Vehicles and Artificial Intelligence Lab (212035)	60h	90h	participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr Thorsten Berger
Lecturers: Prof. Dr. Thorsten Berger
Dr. Sven Peldszus

Module application

M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. IT-Sicherheit/Informationstechnik

Precognitions
Recommended:

- Autonomous Vehicles and Artificial Intelligence (Lecture, 211044)
- Programming experience in Python or C++ (e.g., as part of other courses)
- Software Engineering (Lecture, 212000)

Learning goals
Knowledge

- being able to explain relevant theoretical knowledge of artificial intelligence and autonomous vehicles

Skill and abilities

- define and validate requirements on autonomous vehicles

- instantiate an architecture for autonomous vehicles

- build a self-driving car with ROS2

- manage and integrate artificial intelligence in complex software-intensive systems

- organize a team and its development process for a complex software-intensive system

- perform quality assurance for autonomous vehicles

- document the development process and create the artifacts that would be needed for a certification according to the ISO Road vehicles standards

- communicate with group members and stakeholders professionally (speech and writing)

Contents

Autonomous driving is the future of individual mobility, and all major manufacturers are working on fully autonomous vehicles. While there are robust and well-researched solutions for the individual problems in autonomous driving, the main challenge lies in their integration. Altogether, an autonomous vehicle’s software is the biggest problem. Therefore, the key in self-driving vehicles is about getting the software right.

In this course, we will practically explore the different aspects of self-driving vehicles as well as the importance and application of artificial intelligence in this domain by developing a self-driving race car. To this end, the participants will work with ROS2-based model cars. This aims to provide the students with practical experience in developing an autonomous race car and organizing the development process.

Teaching methods

The main learning sequence in the course is a major practical project. The project is carried out in groups that iteratively develop an autonomous race car, applying and consolidating theoretical knowledge about autonomous driving and software development. To support learning, the Autonomous Vehicles and Artificial Intelligence Lab is based on seminar-style lectures that provide a platform for seeking feedback and more information. Based on the information gathered, students update and refine their solutions for an autonomous race car. Continuous reflection on practice and theory is facilitated by the ongoing production of a final project report in parallel with the development of the race car, in which students reflect on their own learning in the course, the way they and their team approach their development process, and their technical solutions. Students receive regular feedback and guidance to support their learning.

Examination forms

The final grade is determined based on the participation in the development of the self-driving race car, written project reports, the developed autonomous race car, and an oral presentation of the group results. Individual grades will be given by compiling the assessment of individual and group-based outcomes.

Requirements for the award of credits

Actively participate in the development of an autonomous race car, regularly attend seminar-style lectures, and successfully complete all deliverables.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/97: M.Sc. Computer Science

5/105: M.Sc. Angewandte Informatik

5/91: M.Sc. IT-Sicherheit/Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/Informationstechnik [PO 20]

Module title: Computational Neuroscience: Neural Dynamics

Modul code	Credit points	Workload	Term	Frequency	Duration
	6 CP	180 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Computational Neuroscience: Neural Dynamics (212005)	45 h	135 h	participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Gregor Schöner
Lecturers: Prof. Dr. Gregor Schöner

Module application
M.Sc. Angewandte Informatik

M.Sc. Computer Science

Precognitions
This course requires some basic math preparation, typically as covered in two semesters of higher mathematics (functions, differentiation, integration, differential equations, linear algebra). The course does not make extensive use of the underlying mathematical techniques, but uses the mathematical concepts to express scientific ideas. Students without prior training in the relevant mathematics may be able to follow the course, but will have to work harder to familiarize themselves with the concepts.

Learning goals
- Gain experience in interdisciplinarity bridging computer science and cognitive science.
- Learn the concepts and methods of nonlinear dynamical systems in a concrete applied context.
- Improve familiarity with methods of quantitative natural science, including measurement, graphing observables as a function of experimental control parameters and using models to interpret data.
- Read scientific literature.

Contents
This course provides an introduction into the theoretical cognitive and functional neurosciences from a particular theoretical vantage point, the dynamical systems approach. This approach emphasizes the evolution in time of behavioral and neural patterns as the basis of their analysis and synthesis. Dynamic stability, a concept shared with the classical biological cybernetics framework, is one cornerstone of the approach. Instabilities (or bifurcations) extend this framework and provide a basis for understanding flexibility, task specific adjustment, adaptation, and learning. The course includes tutorial modules that provide mathematical foundations. Theoretical concepts are expounded in reference to a number of experimental model systems which include the coordination of movement, postural stability, the perception of motion, and elementary forms of embodied cognition. In the spirit of Braitenberg ́s "synthetic psychology", autonomous robots are used to illustrate some of the ideas. Exercises are integrated into the lectures. They consist of elementary mathematical exercises, the design of (thought) experiments and their analysis, and the design of simple artificial systems, all on the basis of the theoretical framework exposed in the main lectures.
One exercise takes the form of an essay for which participants read a scientific paper and answer questions in a longer illustrated text.

Teaching methods
Lecture with Exercise

Examination forms
Final oral module exam, bonus points for the final exam can be obtained
By submitting homework and an essay (10 pages).

Requirements for the award of credits
Passed final oral module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

6/105: M.Sc. Angewandte Informatik

6/97: M.Sc. Computer Science

Module title: Computational Neuroscience: Single-Neuron Models

Modul code	Credit points	Workload	Term	Frequency	Duration
	6 CP	180 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Computational Neuroscience: Single-Neuron Models (211039)	60 h	120 h	participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Robert Schmidt
Lecturers: Prof. Dr. Robert Schmidt

Module application
M.Sc. Computer Science

M.Sc. Angewandte Informatik

Precognitions
Programming in Python, mathematical knowledge (linear algebra and calculus) and an interest in neurobiology.

Learning goals
- apply techniques from computational neuroscience to simulate neural activity
- become familiar with different types of single neuron models, their mathematical description, and their different levels of biological abstraction
- acquire skills in modelling neurons, synapses and circuits and connect these models to biology and computation
- understanding of the biological basis for computation in neurons

Contents
This module starts with a primer on neuroscience and the role of computational neuroscience. The next part of the module covers biologically-grounded models of single neurons, including leaky-integrate-and-fire and conductance-based neurons, but also more abstract models of neural activity and spike trains. You will learn how these different computational models describe and simplify the underlying biological processes to a different degree. We will examine in detail how these different neuron models can be used in numerical simulations to address research questions on computation in single neurons and circuits. In the exercises accompanying the lectures you will gain hands-on experience in implementing the different neuron models in Python, running numerical simulations, and performing calculations related to analytical solutions of the model equations and biophysics. The focus is on single neuron models, but we will also make use of available software (e.g. NEST Desktop) to examine how single neuron models can be integrated into simulations of neural networks. While the emphasis throughout the module is on methodological issues, how models can be built, tested and validated at each level, we will also draw connections to specific brain regions to motivate and illustrate the models.

Teaching methods
Lecture with exercise

Examination forms
Written module final exam (120 minutes)

Requirements for the award of credits
Passed the final written module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

6/97: M.Sc. Computer Science

6/105: M.Sc. Angewandte Informatik

Module title: Computational Neuroscience: Vision and Memory

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Computational Neuroscience: Vision and Memory (211049)	60 h	90 h	20 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Laurenz Wiskott
Lecturers: Prof. Dr. Laurenz Wiskott

Module application
M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. Cognitive Sience

Precognitions
The mathematical level of the course is mixed but generally high. The tutorial is almost entirely mathematical. Mathematics required include calculus (functions, derivatives, integrals, differential equations, ...), linear algebra (vectors, matrices, inner product, orthogonal vectors, basis systems, ...), and a bit of probability theory (probabilities, probability densities, Bayes' theorem, ...).

Learning goals
After the successful completion of this course the students:

- know basic neurobiological facts about the visual system and the hippocampus,
- know a number of related models and methods in computational neuroscience,
- understand the mathematics of these methods,
- can communicate about all this in English.

Contents
This lecture covers basic neurobiology and models of selforganization in neural systems, in particular addressing

Learning and self-organization

- Hebbian Learning
- Neural learning dynamics and constrained optimization
- Dynamic field theory

Vision

- Receptive fields
- Neural maps
- Hippocampus
- Navigation
- Episodic memory
- Hopfield Network

Teaching methods
This course is given with the flipped/inverted classroom concept. First, the students work through online material by themselves. In the lecture time slot we then discuss the material, find connections to other topics, ask questions and try to answer them. In the tutorial time slot the newly acquired knowledge is applied to analytical exercises and thereby deepened. I encourage all students to work in teams during self-study time as well as in the tutorial.

Examination forms Written (digital) final module exam (90 minutes).
Requirements for the award of credits Passed final written module exam.
Significance of the grade for the final grade (for a total of 120 ECTS) 5/97: M.Sc. Computer Science 5/105: M.Sc. Angewandte Informatik

Module title: Deep Learning

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Deep Learning (212018)	60 h	90 h	50 participants

Teaching language	Requirements for participation
German	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Asja Fischer
Lecturers: Prof. Dr. Asja Fischer

Module application
M.Sc. IT-Sicherheit/ Informationstechnik

M.Sc. IT-Sicherheit/ Netze und Systeme [Until WS 22/23]

M.Sc. Angewandte Informatik

M.Sc. Computer Science

Precognitions

Basic knowledge of linear algebra and probability theory is an advantage.

Learning goals
The lecture aims to provide an insight into this field. At the beginning, the basic terms and concepts of machine learning are introduced. In the further course, different neural networks, gradient-based optimization methods and generative models will be discussed.

Contents
Deep Learning is a subfield of machine learning that has led to breakthroughs in numerous application areas (such as object and speech recognition and machine translation) in recent years. Deep Learning methods find application in the field of IT Security, among others.

Teaching methods
Lecture with exercise

Examination forms
Written final module exam (120 minutes)

Requirements for the award of credits
Passed final written module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

5/91: M.Sc. IT-Sicherheit/ Informationstechnik [PO 22]

5/84: M.Sc. IT-Sicherheit/ Informationstechnik [PO 20]

5/105: M.Sc. Angewandte Informatik

5/ 97: M.Sc. Computer Science

Module title: Machine Learning: Evolutionary Algorithms

Modul code	Credit points	Workload	Term	Frequency	Duration
	6 CP	180 h	see examination regulations	Winter semester	1 semester

Courses	Contact time	Self-study	Group size
Machine Learning: Evolutionary Algorithms (212008)	60 h	120 h	participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Tobias Glasmachers
Lecturers: Prof. Dr. Tobias Glasmachers

Module application
M.Sc. Angewandte Informatik

M.Sc. Computer Science

Precognitions
The course is designed for Master students of technical subjects in (applied) computer science, mathematics, engineering, and natural sciences. It assumes solid knowledge of math basics, in particular linear algebra (e.g., eigen decomposition) and probability, as well as some Python programming.

Learning goals
Internationalisation: The event will be held in English.

Digitisation: Content is conveyed through vidoes and reading material. Exercises with programming components are provided in the form of Jupyter notebooks.

After successful completion of the module

- students know the most important classes of direct optimisation methods and their algorithmic components,
- students have a deep understanding of evolutionary algorithms, especially for continuous problems,
- students know a number of specific problem difficulties and the algorithmic components that address them,
- students can perform elementary runtime analyses and understand the most important convergence classes
- students can implement optimisation methods themselves and apply them to solve new problems.

Contents
This course provides an in-depth introduction to practical optimization with algorithms from the domain of evolutionary computation.

Evolutionary Algorithms are randomized search and optimization heuristics inspired by principles of biological evolution. The field aims to exploit the principle of the "survival of the fittest" for the solution of technical problems. The resulting optimization algorithms are conceptually simple, widely applicable, and easy to implement. Evolutionary search has applications in science and engineering for the approximate solution of difficult "black box" problems.

The course starts with a general overview of the wide field of optimization, including problem modeling. It then introduces different flavors of evolutionary computation, in particular genetic algorithms, genetic programming, and evolution strategies including covariance matrix adaptation. The lecture develops the basic evolutionary optimization model. Various aspects of evolutionary search in discrete and continuous search spaces are discussed in detail, resulting in a systematic taxonomy of largely modular building blocks.

The second part of the course considers a range of typical challenges, like handling constraints, highly multi-modal problems, noisy objective functions, and multiple objectives. It closes with neuroevolution, a method for training neural network controllers, as a modern application domain.

Teaching methods
Lecture with exercise in flipped classroom format.

Examination forms
Written module final exam (90 minutes).

Requirements for the award of credits
Passed written final module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)
6/105: M.Sc. Angewandte Informatik

6/97: M.Sc. Computer Science

Module title: Machine Learning: Supervised Methods

Modul code	Credit points	Workload	Term	Frequency	Duration
	6 CP	180 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Machine Learning: Supervised Methods (211024)	60 h	120 h	80 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Tobias Glasmachers
Lecturers: Prof. Dr. Tobias Glasmachers

Module application
M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. IT-Sicherheit/ Informationstechnik

Precognitions
recommended: Lecture "Mathematics for Modeling and Data Analysis".

Learning goals
Internationalisation: The event will be held in English.

Digitisation: Content is conveyed through vidoes and reading material. Exercises with programming components are provided in the form of Jupyter notebooks.

After successful completion of the module

- students understand the basics of statistical learning theory
- know the most important algorithms of supervised statistical learning and can apply them to learning problems,
- know the strengths and limitations of different learning methods,
- can use standard software for machine learning to solve new problems.

Contents
Basics of statistical learning theory, cross-section of the most important machine learning algorithms, concrete problem solving with standard software.

Teaching methods
Lecture with exercise in flipped classroom format

Examination forms
Written module final exam (90 minutes).

Requirements for the award of credits
Passed written final module exam

Significance of the grade for the final grade (for a total of 120 ECTS)

6/105: M.Sc. Angewandte Informatik

6/97: M.Sc. Computer Science

6/91: M.Sc IT-Sicherheit/ Informationstechnik [PO 22]

6/84: M.Sc IT-Sicherheit/ Informationstechnik [PO 20]

Module title: Machine Learning: Unsupervised Methods

Modul code	Credit points	Workload	Term	Frequency	Duration
	9 CP	270 h	see examination regulations	Winter semester (no offer in WS 23/24)	1 semester

Courses	Contact time	Self-study	Group size
Machine Learning: Unsupervised Methods (212501)	60 h	210 h	40 participants

Teaching language	Requirements for participation
English	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Laurenz Wiskott
Lecturers: Prof. Dr. Laurenz Wiskott

Module application
M.Sc. Computer Science

M.Sc. Angewandte Informatik

M.Sc. Elektro-und Informationstechnik

Precognitions
The mathematical level of the course is mixed but generally high, including calculus (functions, derivatives, integrals, differential equations, ...), linear algebra (vectors, matrices, inner product, orthogonal vectors, basis systems, ...), and a bit of probability theory (probabilities, probability densities, Bayes' theorem, ...). Programming is
done in Python, thus the students should have a basic knowledge of that as well, or at least be fluent in another programming language.

Learning goals
After the successful completion of this course the students:

- know a number of important unsupervised learning methods,
- can discuss and decide which of the methods are appropriate for a given data set,
- understand the mathematics of these methods,
- know how to implement and apply these methods in python,
- have gained experience in organizing and working in a team,
- know problem solving strategies like brain storming,
- can communicate about all this in English.

Contents
This course covers a variety of shallow unsupervised methods from machine learning such as principal component analysis, independent component analysis, vector quantization, clustering, Bayesian theory and graphical models.

Teaching methods
This course is given in a hybrid of inverted classroom and problem based learning. The course starts with a two-week introduction into unsupervised methods of machine learning, providing an overview. The students then work in groups of about 4 on realistic problems that can be solved with these methods. In the first week of a problem, they develop hypotheses and strategies for a solution and identify which methods they want to learn. Then the course agrees on a method to focus on theoretically, which will then be done in an inverted classroom format. The students then try to solve the problem and present their results in a short talk with slides. Thus the students will not only learn about machine learning but also soft skills.

Examination forms
The exam is a combination of graded presentations for the problems and graded quizzes for the theory. 40% of the grade come from the average group performance on solving the problems. 10% come from the presentations, taking into account slides and presentation style, this is an individual grade of the presenter. 50% come from a digital quiz about the theory of the methods covered. Thus 60% of the grade are individual, 40% come from the

group. In addition you can gain up to 8 bonus points for (i) being voted for as a ’most valuable player (MVP)’ on a project and (ii) creating useful quiz questions.

Requirements for the award of credits
Continuous participation and passed exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

9/105: M.Sc. Angewandte Informatik

9/97: M.Sc. Computer Science

Module title: Statistical learning and data mining

Modul code	Credit points	Workload	Term	Frequency	Duration
	5 CP	150 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Statistisches Lernen und Data Mining (150331)	60 h	90	20 participants

Teaching language	Requirements for participation
English	none

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Johannes Lederer
Lecturers: Prof. Dr. Johannes Lederer

Module application
B.Sc. Informatik

M.Sc. Computer Science

Precognitions

Learning goals
After successful completion of the module

- students know standard methods of data analysis
- they understand when which methods are appropriate
- they are able to apply the methods
- they are able to interpret the results

Contents
This course introduces the basic methods of data analysis. Different types of data are considered, especially regression data and classification data. The underlying statistical models are always discussed as well. Likewise, possible applications are presented both in class and in computer exercises. The aim is to teach the whole process of simple data analysis: Data preparation, statistical modelling, selection of a method, implementation of the method, visualisation of the results and interpretation.

Teaching methods
Lecture with media support, in particular data analysis with the computer, tutorials as seminar-based teaching, additional self-study with supplementary materials and tasks provided.

Examination forms
Written module final exam (90 minutes).

Requirements for the award of credits

Passed written final module exam

Significance of the grade for the final grade (for a total of 120 ECTS)

5/158: B.Sc. Informatik [PO 22]

5/165: B.Sc. Informatik [PO 20]

5/97: M.Sc. Computer Science

Module title: Theory of machine learning

Modul code	Credit points	Workload	Term	Frequency	Duration
	9 CP	270 h	see examination regulations	Summer semester	1 semester

Courses	Contact time	Self-study	Group size
Theorie des maschinellen Lernens (211052)	90 h	180 h	20 participants

Teaching language	Requirements for participation
German	None

Module coordinators and Lecturers
Module coordinators: Prof. Dr. Asja Fischer
Lecturers: Prof. Dr. Asja Fischer

Module application
M.Sc. Angewandte Informatik

M.Sc. Computer Science

Precognitions

Learning goals
Students are familiarised with mathematical models for machine learning. They acquire the ability to evaluate and compare learning algorithms according to the degree to which they achieve (precisely described) success criteria. They acquire techniques both for designing efficient learning algorithms and for proving the inherent hardness of a problem.
After the successful completion of the module

- students know the most important learning machines (such as Support Vector Machines and related models),
- students understand the difference between empirical and real error rate and know techniques to deal with the problem of overfitting the data (with a model that is too complex),
- can differentiate between uniform and non-uniform learnability of a Hypothesis class and know the appropriate theories and learning rules.

Contents
The subject of the lecture is the statistics-based theory of machine learning. In particular, the method of structured risk minimisation is taught as well as the statistical theorems on which it is based. Techniques for designing efficient learning algorithms are discussed as well as information- or computational-theoretic barriers that make certain learning problems appear to be inefficiently solvable.

Teaching methods
Lecture with exercise

Examination forms
Final oral module exam (30 minutes)

Requirements for the award of credits
Passed final oral module exam.

Significance of the grade for the final grade (for a total of 120 ECTS)

9/105: M.Sc. Angewandte Informatik

9/97: M.Sc. Computer Science

Module title: Practical Labs

Modul code	Credit points	Workload	Term	Frequency	Duration
	3 CP	90-150 h	see examination regulations	Every winter and summer semester	1 semester

Courses	Contact time	Self-study	Group size
A list of the selectable practical labs with assignment to the specialisation follows.	30 h	60-120 h	1-10 participants

Teaching language	Requirements for participation
German or English	None

Module coordinators and Lecturers
Module coordinators: Studiendekan der Fakultät für Informatik / Dean of Studies of the Faculty of Computer Science
Lecturers: Dozierende im Studiengang M.Sc. Computer Science / Lecturers in the study program M.Sc. Computer Science

Module application
M.Sc. Computer Science

Precognitions
Depending on the chosen practical lab.

Learning goals
The students are able to work in a small team to solve tasks in the field of computer science and to document the results in an engineering manner. They can specifically apply methods of structured analysis and analyse their effect.

Contents
Within the scope of this area, a laboratory practical course of 3 LP - 5 LP is to be completed. This is not a pure programming practical course, but a topic-related course with scientific demands. The practical courses are offered on advanced topics in computer science.

Teaching methods
Practical lab at a chair of the Faculty of Computer Science.

Examination forms
Practical exam and documentation

Requirements for the award of credits
Passed practical exam and documentation

Significance of the grade for the final grade (for a total of 120 ECTS)
unbenotet / ungraded

Module title: Project

Modul code	Credit points	Workload	Term	Frequency	Duration
	10 CP	300 h	see Examination regulations	Every winter and summer semester	1 semester

Courses	Contact time	Self-study	Group size
	15 h	270 h	1-5 participants

Teaching language	Requirements for participation
German or English	none

Module coordinators and Lecturers
Module coordinators: Studiendekan der Fakultät für Informatik / Dean of Studies of the Faculty of Computer Science
Lecturers: Dozierende im Studiengang M.Sc. Computer Science / Lecturers in the study program M.Sc. Computer Science

Module application
M.Sc. Computer Science

Precognitions
Prior knowledge: Good programming skills are required. For most projects, prior technical knowledge is required that is specific to the project.

Learning goals
- Application of the acquired technical knowledge at master level
- Acquisition of technical competence according to the respective project-specific tasks
- Application of project management techniques from the field of software engineering
- Development of own solution strategies
- Documentation and presentation of results

Contents
Within the framework of the project work, a task from areas of computer science is to be solved under the guidance of a supervisor. The project is preferably carried out in industry or alternatively at the Ruhr University. The topic of the project covers any aspect of computer science. It can be carried out in group work.

Teaching methods
Project work

Examination forms
Project work with final report

Requirements for the award of credits
Successfully completed project work and final report graded at least satisfactory.

Significance of the grade for the final grade (for a total of 120 ECTS)
10/97: M.Sc. Computer Science

Module title: Seminars

Modul code	Credit points	Workload	Term	Frequency	Duration
	3 CP	90 h	see Examination regulations	Every winter and summer semester	1 semester

Courses	Contact time	Self-study	Group size
A listing of the elective seminars with assignment to the specialization follows.	30 h	60 h	15 participants

Teaching language	Requirements for participation
German or English	None

Module coordinators and Lecturers
Module coordinators: Studiendekan der Fakultät für Informatik / Dean of Studies of the Faculty of Computer Science
Lecturers: Dozierende im Studiengang M.Sc. Computer Science / Lecturers in the study program M.Sc. Computer Science

Module application
M.Sc. Computer Science

Precognitions
depending on choice of seminar

Learning goals
After successful completion of the module

- students have in-depth scientific knowledge of the selected seminar topic
- have practiced giving a scientific presentation and can independently communicate research results in a didactically well-prepared presentation
- are able to formulate and receive constructive feedback

Contents
Important skills such as literature work and presentation techniques are practiced in the seminar. Students can choose from the wide range of seminars in different thematic areas, such as Resource Efficient System Software Concepts, Advanced Topics in Deep Learning, Distributed and Networked Systems .

Teaching methods
seminar

Examination forms
Seminar presentation and, if applicable, written paper

Requirements for the award of credits
Successful seminar presentation, written paper if applicable, and attendance at 9 of 10 individual appointments.

Significance of the grade for the final grade (for a total of 120 ECTS)
3/97: M.Sc. Computer Science

Module title: Free Elective Moduls

Modul code	Credit points	Workload	Term	Frequency	Duration
	20 CP	600 h	see examination regulations	Each winter and summer semester	1 semester

Courses	Contact time	Self-study	Group size
any	240 h	360 h	10-50 participants

Teaching language	Requirements for participation
depending on choice of event	None

Module coordinators and Lecturers
Module coordinators: Studienfachberatung M.Sc. Computer Science / Study advisory service M.Sc. Computer Science
Lecturers: Diverse / Various

Module application
M.Sc. Computer Science

Precognitions
Depending on choice of event

Learning goals
Participants acquire so-called key skills in the free electives.

Contents
Within the scope of the free elective area, courses with a volume of at least 20 LP must be completed, which can be freely selected from the courses offered by the university and the UA-Ruhr. Non-technical courses (e.g. foreign languages, law, economics or social sciences) as well as technical courses (e.g. engineering sciences, computer science) can be chosen.

Teaching methods
depending on choice of event

Examination forms
depending on choice of event

Requirements for the award of credits
depending on choice of event

Significance of the grade for the final grade (for a total of 120 ECTS)
unbenotet / ungraded

Module title: Master's thesis and colloquium

Modul code	Credit points	Workload	Term	Frequency	Duration
	30 CP	900 h	4	each semester	1 semester

Courses	Contact time	Self-study	Group size
	15 h	885 h	1 participant

Teaching language	Requirements for participation
English	Successfully completed modules amounting to 70 CP.

Module coordinators and Lecturers
Module coordinators: Studiendekan Informatik / Dean of Studies Computer Science
Lecturers: Lehrende im Studiengang Informatik / Lecturers in the Computer Science programme

Module application
M.Sc. Computer Science

Precognitions
Depending on the choice of topic.

Learning goals
After successful completion of the module:

- students can independently work on a scientific topic in a timely manner from research to documentation of the results.
- students can select, apply and further develop suitable scientific procedures and methods, which they have become acquainted with during their studies, in order to solve a concrete problem
- can critically compare and evaluate their results with the state of the art in research
- students can present their own results appropriately in written and spoken form.

Contents
The Master's thesis is a research-oriented, six-month thesis on a specific topic and is written in the last semester of the degree programme. This has a volume of 30 credit points. The Master's thesis is written in English.

Teaching methods
Thesis

Examination forms
Master's thesis and colloquium presentation

Requirements for the award of credits
Both the written Master's thesis and the colloquium presentation must be passed.

The share of the colloquium grade in the overall grade is 10%.

Significance of the grade for the final grade (for a total of 120 ECTS)
30/97: M.Sc. Computer Science