UNIVERSITÄT
DES
SAARLANDES

Faculty of Mathematics and Computer Science

Module Descriptions

Computer Science BSc (English)

25th February 2025

List of module categories and modules

Module Category 1

Lecture Series on Topics in Computer Science

Perspectives in Computer Science — PiCS

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
1	**6**	**every winter semester**	**1 semester**	**2**	**2**

responsible Dean of Studies of the Faculty of Mathematics and Computer Science
Dean of Studies of the Department of Computer Science

lecturers Lecturers of the department

entrance requirements none

assessments / exams Demonstrate understanding of the content of at least three lectures, e.g by written paper or test.

course types / weekly hours `2 h lecture`

total workload
```
   30 h of classes
+  30 h private study
=  60 h (= 2 ECTS)
```

grade The module is passed overall if the examination performance has been passed (ungraded).

language English / Deutsch

aims / competences to be developed

Early motivation and overview of the central scientific topics of computer science, as well as of the competencies of the computer science department in Saarbrücken.

content

Lectures by weekly changing lecturers offer a cross-section of research topics in computer science in Saarbrücken. The topics span an attractive range from the latest research to challenging problems of industrial practice.

literature & reading

Material will be provided suitable to the individual lectures.

additional information

This module is identical in content to the German-language module *Perspektiven der Informatik*.

Module Category 2

Fundamentals of Mathematics

Mathematics for Computer Scientists 1 MfCS1

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
1	**6**	**every winter semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Joachim Weickert

lecturers Prof. Dr. Joachim Weickert
Prof. Dr. Mark Groves
Prof. Dr. Henryk Zähle
Prof. Dr. Christian Bender

entrance requirements none

assessments / exams
- Regular and active participation in tutorials and completion of weakly exercise sheets. An overall score of 50 percent on the tutorial sheets is required to qualify for the examination.
- Examination at the end of the module.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade To be determined from performance in examinations and tutorials. Exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

- Basic mathematical knowledge required in the context of a computer science or bioinformatics degree.
- Ability to formalise and abstract
- Ability to acquire further mathematical knowledge with the help of text books

content

The numbers in parentheses indicate the total number of 2 hour lectures.

```
DISCRETE MATHEMATICS AND ONE-DIMENSIONAL ANALYSIS

A. Fundamentals of discrete mathematics (8)
    1. sets (1)
    2. logic (1)
    3. methods of mathematical proof, including induction (1)
    4. relations (1)
    5. maps (2)
        - injective, surjective, bijective
        - cardinality, countability
        - pigeon-hole principle
    6. prime numbers and divisors (1)
    7. modular arithmetic (1)
```

```
B. One-dimensional analysis (22)

B.1 Numbers, sequences and series (8)
     8. Axiomatics of real numbers, supremum, infimum (1)
     9. complex numbers (1)
    10. sequences (1 1/2)
    11. big O notation (1/2)
    12. series: convergence tests, absolute convergence (2)
    13. power series (1/2)
    14. representations of numbers (1/2)
    15. binomial coefficients and binomial series (1)

B.2 One-dimensional differential calculus (8)
    16. continuity (1)
    17. elementary functions (1)
    18. differentiability (1 1/2)
    19. mean-value theorems and L'Hopital's rule (1/2)
    20. Taylor's theorem (1)
    21. local extrema, convexity, curve sketching (2)
    22. numerical differentiation (1)

B.3 One-dimensional integral calculus (6)
    23. definite integrals (2)
    24. indefinite integrals and the antiderivative (1)
    25. improper integrals (1)
    26. numerical methods for integration (1)
    27. curves and arc length (1)
```

literature & reading

To be announced before the start of the module on the relevant internet page.

additional information

This module is identical in content to the German-language module *Mathematik für Informatiker 1*.

Mathematics for Computer Scientists 2 MfCS2

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
2	**6**	**every summer semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Joachim Weickert

lecturers Prof. Dr. Joachim Weickert
Prof. Dr. Mark Groves
Prof. Dr. Henryk Zähle
Prof. Dr. Christian Bender

entrance requirements *Mathematics for Computer Scientists 1* is recommended.

assessments / exams
- Regular and active participation in tutorials and completion of weakly exercise sheets. An overall score of 50 percent on the tutorial sheets is required to qualify for the examination.
- Examination at the end of the module.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade To be determined from performance in examinations and tutorials. Exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

- Basic mathematical knowledge required in the context of a computer science or bioinformatics degree.
- Ability to formalise and abstract
- Ability to acquire further mathematical knowledge with the help of text books

content

The numbers in parentheses indicate the total number of 2 hour lectures.

```
LINEAR ALGEBRA

C. Algebraic structures (5)
    29. groups (2)
    30. rings and fields (1)
    31. polynomial rings over fields (1/2)
    32. Boolean algebras (1/2)

D. Linear algebra (21)
    33. vector spaces (2)
        - definition, examples
        - linear maps
        - subspaces
        - linear span, linear dependence, basis, exchange theorem
```

34. linear transformations (image, kernel) (1)
35. matrix representations of linear transformations (1 1/2)
    - interpretation as linear transformations
    - multiplication by composition
    - ring structure
    - inverses
36. rank of a matrix (1/2)
37. Gaussian algorithmn for systems of linear equations (2)
    - Gaussian elimination (1)
    - Back substitution (1)
38. iterative methods for systems of linear equations (1)
39. determinants (1)
40. Euclidean vector spaces, scalar products (1)
41. functional-analytic generalisations (1)
42. orthogonality (2)
43 Fourier series (1)
44. orthogonal matrices (1)
45. eigenvalues and eigenvectors (1)
46. eigenvalues and eigenvectors of symmetric matrices (1)
47. quadratic forms and positive-definite matrices (1)
48. quadrics (1)
50. matrix norms and eigenvalue estimates (1)
51. numerical calculation of eigenvalues and eigenvectors (1)

literature & reading

To be announced before the start of the module on the relevant internet page.

additional information

This module is identical in content to the German-language module *Mathematik für Informatiker 2*.

Mathematics for Computer Scientists 3 MfCS3

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
3	**6**	**every winter semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Joachim Weickert

lecturers Prof. Dr. Joachim Weickert
Prof. Dr. Mark Groves
Prof. Dr. Henryk Zähle
Prof. Dr. Christian Bender

entrance requirements *Mathematics for Computer Scientists 1* and *2* are recommended.

assessments / exams
- Regular and active participation in tutorials and completion of weakly exercise sheets. An overall score of 50 percent on the tutorial sheets is required to qualify for the examination.
- Examination at the end of the module.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade To be determined from performance in examinations and tutorials. Exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

- Basic mathematical knowledge required in the context of a computer science or bioinformatics degree.
- Ability to formalise and abstract
- Ability to acquire further mathematical knowledge with the help of text books

content

The numbers in parentheses indicate the total number of 2 hour lectures.

```
STOCHASTICS, NUMERICAL ANALYSIS AND MULTIDIMENSIONAL ANALYSIS

E. Numerical complements (3)
    52 Banach fixed-point theorem (1)
    53. interpolation, including splines (2)

F. Multidimesional analysis and numerical analysis (11)
    54. continuity and differential operators for scalar-valued functions (2)
    55. differential operators for vector-valued functions (1)
    56. total differentiability (1/2)
    57. mean value theorem and Taylor's theorem (1 1/2)
    58. extrema of functions of several variables (1)
    59. Newton's method (1)
    60. extrema with side conditions (1)
```

61. multiple integrals(1)
62. inverse functions and the transformation rule (1)
63. calculus of variations (1)

G. Stochastics (16)
64. basic concepts (probability, sample space) (1/3)
65. combinatorics (2/3)
66. generating functions (1)
67. conditional probabilities (1)
68. random variables, expectated values, variance (2)
    (system reliability, variance, covariance, Jensen)
69. estimates of deviations from the mean (1)
    (moments, Markov bounds, Chebyshev, Chernoff, weak law of large numbers)
70. important discrete distributions (1)
71. important continuous distributions (1) (including central limit theorem)
72. multivariate distributions and sums of random variables (1)
73. parameter estimation and confidence intervals (1)
74. hypothesis testing (1)
75. method of least squares (1)
76. robust statistics (2/3)
77. propagation of uncertainty (1/3)
78. markov chains (2)
79. pseudo-random numbers and Monte-Carlo method (1)

literature & reading

To be announced before the start of the module on the relevant internet page.

additional information

This module is identical in content to the German-language module *Mathematik für Informatiker 3*.

Module Category 3

Fundamentals of Computer Science

Big Data Engineering BDE

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4	**6**	**every summer semester**	**1 semester**	**4**	**6**

responsible Prof. Dr. Jens Dittrich

lecturers Prof. Dr. Jens Dittrich

entrance requirements *Programming 1*, *Programming 2*, *Software Engineering Lab*, *Mathematics for Computer Scientists 1*, as well as *Fundamentals of Algorithms and Data Structures* (all recommended)

assessments / exams Successful participation in the exercises/project entitles the student to take part in the final exam.

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in exams, exercises, and (optionally) practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The lecture provides basic knowledge of fundamental concepts of data management and data analysis in Big Data Engineering.

As part of the exercises, a project can be carried out during the semester. This can be, for example, a social network (Facebook style) or any other project where data management techniques can be practiced (e.g., natural science data, image data, other web applications, etc.). First, this project will be modeled in E/R, then realized and implemented in a database schema. Then the project is extended to manage and analyze unstructured data as well. Altogether, all fundamental techniques that are important for managing and analyzing data are thus demonstrated on a single project.

content

```
1 Introduction and classification
    Classification and delimitation: "Big Data"
    Value of Data: The gold of the 21st century
    Importance of database systems
    What is data?
    Modeling vs Reality
    Costs of inadequate modeling
    Using a database system vs developing it yourself
    Positive examples for apps
    Requirements
    References
    Lecture mode

2 Data modeling
    Motivation
```

E/R
Relational Model
domains, attributes
entity type vs entity
relation type vs relation
Hierarchical Data
keys, foreign keys
inheritance
Redundancy, normalization, denormalization

3 query languages
Relational Algebra
Graph-oriented query languages

4 SQL
Basics
Relationship to relational algebra
CRUD-style vs analytical SQL
SQL standards
joins, grouping, aggregation, having
PostgreSQL
Integrity constraints
Transaction concept
ACID
Views

5 Basic query optimization
Overview
from WHAT to HOW
Costs of different operations
EXPLAIN
Physical Design
Indexes, Tuning
Database tuning
Rule-based query optimization
Cost-based query optimization

6 Automatic Concurrency control
Serializability theory
Isolation levels
Pessimistic concurrency control
lock-based approaches, 2PL-variants

7 Grahical Data
recursion in SQL, WITH RECURSIVE
graph-oriented query languages: e.g. Cypher, Neo4J

8 Database Security
SQL injection
passwords
salt and pepper

9 Ethical Aspects of Big Data
mass surveillance
NSA
the "big data arithmetic"
counter measures

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module was formerly also known as *Informationssysteme*. This module is identical in content to the German language module *Big Data Engineering*.

Concurrent Programming CP

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4	**6**	**every summer semester**	**1 semester**	**4**	**6**

responsible Prof. Dr.-Ing. Holger Hermanns

lecturers Prof. Dr.-Ing. Holger Hermanns
Prof. Dr. Bernd Finkbeiner
Prof. Dr. Verena Wolf

entrance requirements *Programming 1* and *2*, *Software Engineering Lab*, and *Introduction to Theoretical Computer Science* (recommended).

assessments / exams Two exams (mid-term and end-term), practical project.

A re-exam for the mid-term will take place before the end-term, a re-exam for the end-term takes place within the last weeks before the start of lectures of the following semester.

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload

```
   50 h of classes
+ 130 h private study
= 180 h (= 6 ECTS)
```

grade Is determined from performance in written examinations, as well as the preparatory examinations. The exact modalities will be announced by the person responsible for the module.

language English / Deutsch

aims / competences to be developed

The participants of this course get acquainted with concurrency in computation as a far-reaching and foundational principle with respect to both theory and application of modern computing sciences. By analysing and applying different formal models, the participants gain a deeper understanding of concurrency, and learn to apply formal computing concepts correctly. The theoretical knowledge acquired in the first half of the lecture is in the second half applied to practical programming. Therein, participants learn using the programming paradigms “shared memory” and “message passing” starting off with the programming language pseuCo before applying their skills to Java and (partially) to Rust. In addition, participants learn to describe various phenomena of concurrent programming using formal models, and to derive concrete solutions for practical problems from them. Moreover, the participants examine existing practitioneer’s concepts with respect to their reliability. A specific aspect of this professional practice is the tactically adequate reaction to concurrency problems under tight time constraints.

content

Concurrency as a Concept

- potential parallelism
- actual parallelism
- conceptional parallelism

Concurrency in Practice

- object orientation

- operating systems
- multi-core processors, coprocessors
- programmed parallelism
- distributed systems (client-server, peer-to-peer, databases, the Internet)

Problems of Concurrency

- resource conflicts
- fairness
- mutual exclusion
- deadlock
- livelock
- starvation

Foundations of Concurrency

- sequential vs. concurrent processes
- states, events and transitions
- transition systems
- observable behaviour
- determinism vs. non-determinism
- algebras and operators

CCS - The Calculus of Communicating Systems

- constructing processes: sequence, choice, recursion
- concurrency and interaction
- structural operational semantics
- equivalence of observations
- implementation relations
- CCS with message passing

Programming Concurrency

- pseuCo
- message passing in pseuCo and Go
- shared memory in pseuCo and Java
- shared objects and threads in Java
- shared objects and threads as transition systems

Programming and Analysis Support

- deadlock detection
- verification of safety and liveness
- model-based design supporting concurrency
- software architectures supporting concurrency

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module is identical in content to the German-language module *Nebenläufige Programmierung*.

Elements of Machine Learning — EML

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5	**6**	**every winter semester**	**1 semester**	**4**	**6**

responsible Prof. Dr. Jilles Vreeken
Prof. Dr. Isabel Valera

lecturers Prof. Dr. Jilles Vreeken
Prof. Dr. Isabel Valera

entrance requirements The lecture assumes basic knowledge in statistics, linear algebra, and programming. It is advisable to have successfully completed *Mathematics for Computer Scientists 2* and *Statistics Lab*. The exercises use the programming language R. We will give a basic introduction to R in the first tutorial. In addition, for preparation the following materials are useful: *R for Beginners* by Emmanuel Paradis (especially chapters 1, 2, 3 and 6) and *An introduction to R* (Venables/Smith).

assessments / exams Prerequisite for admission to the examination is a cumulative 50% of the points of the theoretical and a cumulative 50% of the points of the practical tasks on the exercise sheets. Depending on the number of participants, the examinations are either written or oral. The final modality will be announced in the first two weeks of the lecture.

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in exams.

language English

aims / competences to be developed

In this course we will discuss the foundations – the elements – of machine learning. In particular, we will focus on the ability of, given a data set, to choose an appropriate method for analyzing it, to select the appropriate parameters for the model generated by that method and to assess the quality of the resulting model. Both theoretical and practical aspects will be covered. What we cover will be relevant for computer scientists in general as well as for other scientists involved in data analysis and modeling.

content

The lecture covers basic machine learning methods, in particular the following contents:

- Introduction to statistical learning
- Overview over Supervised Learning
- Linear Regression
- Linear Classification
- Splines
- Model selection and estimation of the test errors
- Maximum-Likelihood Methods
- Additive Models

- Decision trees
- Boosting
- Dimensionality reduction
- Unsupervised learning
- Clustering
- Visualization

literature & reading

The course broadly follows the book *An Introduction to Statistical Learning with Applications in R*, Springer (2013). In some cases, the course receives additional material from the book *The Elements of Statistical Learning*, Springer (second edition, 2009). The first book is the introductory text, the second covers more advanced topics. Both books are available as free PDFs. Any change of, or additional material will be announced before the start of the course on the course webpage.

Fundamentals of Data Structures and Algorithms

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
3	**6**	**every winter semester**	**1 semester**	**4**	**6**

responsible Prof. Dr. Raimund Seidel

lecturers Prof. Dr. Raimund Seidel
Prof. Dr. Markus Bläser
Prof. Dr. Karl Bringmann

entrance requirements *Programming 1* and *2*, and *Mathematics for Computer Scientists 1* and *2* or comparable courses in mathematics are recommended.

assessments / exams Successful completion of the exercise sheets entitles to take part in the exam.

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
  60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Students get to know the most important methods of designing algorithms and data structures: divide-and-conquer, dynamic programming, incremental construction, ”greedy algorithms”, decimation, forming hierarchies, randomization. They learn to analyze algorithms and data structures for their time and space requirements with respect to the usual RAM machine model and to compare them on this basis. Various kinds of analysis are considered (worst case, amortized, expected case).

Students get acquainted with important efficient data structures and algorithms. They should acquire the ability to apply theoretial analyses and considerations to given methods in order to check their applicability to actually occuring scenarios. Moreover, students should school their skills in developing or adjusting algorithms and data structures with performance guarantees in mind.

content

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module is identical in content to the German-language module *Grundzüge von Algorithmen und Datenstrukturen*.

Introduction to Theoretical Computer Science

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
3	**6**	**every winter semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Raimund Seidel

lecturers Prof. Dr. Raimund Seidel
Prof. Dr. Bernd Finkbeiner
Prof. Dr. Markus Bläser
Prof. Dr. Karl Bringmann

entrance requirements *Programming 1* and *2* and *Mathematics for Computer Scientists 1* and *2* or comparable courses in mathematics are recommended.

assessments / exams Successful completion of the exercises entitles the student to take the exam.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Students know various models of computation and their relative strengths and abilities.

For selected problems they can show, whether they are solvable in a certain model of computation or not.

They understand the formal notion of computability as well as non-computability.

They can reduce problems to each other.

They are familiar with basics of bounding resources (time, space) for computations and the resulting complexity theory.

content

The language classes of the Chomsky hierarchy and their various definitions via grammars and automata; closure properties; classification of particular languages (”pumping lemmas”);

determinism and non-determinism;

Turing machines and equivalent models of general computability (e.g. µ-recursive function, random acces machins), reducibility, decidability, undecidability;

the complexity measures time and space; the complexity classes P and NP;

the basics of the theory of NP-completeness.

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module is identical in content to the German-language module *Grundzüge der Theoretischen Informatik*.

Programming 1 — Prog1

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
1	**6**	**every winter semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Gert Smolka

lecturers Prof. Dr. Gert Smolka
Prof. Dr.-Ing. Holger Hermanns
Prof. Bernd Finkbeiner, Ph.D

entrance requirements none

assessments / exams
- Weekly exercises / tests
- Midterm and endterm exam
- Re-examination at end of semester

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Grade combines performance in exams and weekly exercises.

language English

aims / competences to be developed

- functional programming, higher-order and typed
- practical programming skills using an interpreter, debugging, testing
- recursive data structures and recursive algorithms (numbers, lists, trees)
- exceptions
- type abstraction and modularity
- data structures with mutable state, exceptions
- correctness proofs and runtime estimates
- structure of programming languages
- formal description of programming languages (syntax and semantics)
- implementation of programming languages (parsers, interpreters, compilers, stack machines)

content

see above

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module is identical in content to the German-language module *Programmierung 1*.

Programming 2 — Prog2

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
2	**6**	**every summer semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Sebastian Hack

lecturers Prof. Dr. Sebastian Hack
Prof. Dr. Jörg Hoffmann

entrance requirements *Programming 1* and *Mathematics for Computer Scientists 1* and mathematics courses in the study semester or comparable knowledge from other mathematics courses (recommended)

assessments / exams Examination performances are given in two parts, which contribute equally to the final grade. To pass the entire course, each part must be passed individually.

In the **practical part**, students must implement a series of programming tasks independently. These programming tasks allow students to practise language concepts and also introduce more complex algorithms and data structures. Automatic tests check the quality of the implementations. The grade of the practical part is largely determined by the test results.

In the **lecture part**, students must complete written examinations and work on exercises. The exercises deepen the material of the lecture. Admission to the written examination depends on the successful completion of the exercises.

In the practical part, a follow-up task can be offered.

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload

```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

This course teaches the foundations of imperative and object-oriented programming.

In more detail students learn:
* how computers execute programs and how to write programs in assembly language * to implement, debug, and test smaller C programs * to design, implement, debug, and test mid-size Java programs * the basics of object-oriented programming * a basic understanding of formal semantics, type systems, correctness, testing, and verification of imperative languages

content

- Programming at the machine level (assembly)
- Imperative programming
- Object-oriented programming
- Classes and objects
- Inheritance, sub-typing, and dynamic dispatch
- Formal semantics and a type system of a simple imperative language

- Type safety, undefined behavior and their implications
- Foundations of testing and verification

as well as lectures specifically designed for the individual programming tasks.

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module is identical in content to the German-language module *Programmierung 2*.

System Architecture — SysArch

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
2	**6**	**every summer semester**	**1 semester**	**6**	**9**

responsible Prof. Dr. Jan Reineke

lecturers Prof. Dr. Jan Reineke

entrance requirements *Programming 1*, *Programming 2* (in the same semester), and *Mathematics for Computer Scientists 1* or comparable courses in mathematics are recommended.

assessments / exams The course consists of two parts, which each have to be passed individually in order pass the course as a whole.

In the *projects part*, students have to independently implement a series of projects. These projects deepen the practical comprehension of the lecture material in the areas of computer architecture and operating systems.

In the *lecture part*, students must pass the written exams and work on written assignments and/or quizzes. Successful completion of the written assignments and/or the quizzes is a prerequisite for participation in the written exams.

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload

```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined based on the performance in exams, exercises, and projects. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Students shall understand the functionality and the most important properties of modern computer architectures and operating systems.

Furthermore students shall understand the design principles underlying their implementations.

content

1. Computer architecture
    a. Boolean algebra and combinatorial circuits
    b. Number representations and arithmetic circuits
    c. Instruction set architectures
    d. Microarchitectures, in particular, the design of a basic reduced instruction set machine, and performance optimizations such as pipelining and caches

2. Operating systems
    a. Virtualization mechanisms
    b. Scheduling algorithms
    c. File systems

literature & reading

Will be announced before the start of the course on the course page on the internet.

additional information

This module is identical in content to the German-language module *Systemarchitektur*.

Module Category 4

Practical Skills Classes

Software Engineering Lab — SE Lab

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
2-3	**6**	**lecture free time after SS**	**7 weeks**	**BLOCK**	**9**

responsible Prof. Dr. Sven Apel

lecturers Prof. Dr. Sven Apel
Dr. Norman Peitek

entrance requirements Participation in the Software Engineering Lab requires extensive programming skills as taught in the courses *Programming 1* and *Programming 2*. A passing grade in *Programming 2* is required to enroll in this course.

Students are required to bring their own laptops.

assessments / exams The goal of the Software Engineering Lab is to develop a non-trivial software system in a team effort. In this course, a number of documents (design models, documentation, implementation plan, etc.) and artifacts (source code, tests, etc.) need to be developed and submitted. Correctness, completeness, quality, and timely submission of all documents and artifacts are major grading criteria.

The Software Engineering Lab consists of two phases: exercise phase and group phase.

In the *exercise phase*, participants will complete an entry exam, covering current topics from the lecture. Only participants that have passed the exercise phase will be admitted to the group phase.

In the *group phase*, participants will first design and then implement and test a substantial software system in a team effort, and submit both their design and their implementation (including tests) for evaluation. All documents (design models, documentation, implementation plan, etc.) and artifacts (source code, tests, etc.) of the group phase will be evaluated based on the principles and quality criteria conveyed in the lectures. To pass the group phase, students must pass both the design submission and the implementation submission, and prove individually their substantial contribution to the group project.

More details on the exams will be announced at the beginning of the course.

course types / weekly hours Daily exercises and lectures (first few weeks)
Daily project work with tutoring

total workload

```
  35 h of lectures and exercises
+ 235 h project work
= 270 h (= 9 ECTS)
```

grade ungraded

language English

aims / competences to be developed

Participants acquire the ability to solve complex software development problems individually and in teams.

Participants are aware of common problems and pitfalls of software development and know how to address them.

Participants are able to accomplish and coordinate software development tasks based on a set of given requirements. For this purpose, they are able to select proper methods and techniques to minimize risks and maximize software quality.

Participants know about foundations and principles of software design, including cohesion, coupling, modularity, encapsulation, abstraction, and information hiding. They are acquainted with a whole array of design patterns, knowing their aim

and individual strengths and weaknesses. They are able to apply design patterns beneficially and to judge and improve the quality of software designs.

Participants master fundamental techniques and tools for software testing, debugging, and version control.

content

- Software design
- Software testing
- Team work
- Debugging

literature & reading

- Software Engineering. I. Sommerville, Addison-Wesley, 2004.
- Software Engineering: A Practioner's Approach. R. Pressman, McGraw Hill Text, 2001.
- Using UML: Software Engineering with Objects and Components. P. Stevens, et al., Addison-Wesley, 1999.
- UML Distilled. M. Fowler, et al., Addison-Wesley, 2000.
- Objects, Components and Frameworks with UML, D. D'Souza, et al., Addison-Wesley, 1999.
- Designing Object-Oriented Software. R. Wirfs-Brock, et al., Prentice Hall, 1990.
- Design Patterns. Elements of Reusable Object-Oriented Software. E. Gamma, et al., Addison-Wesley, 1995.
- Head First Design Patterns. E. Freeman, et al. O'Reilly, 2004.
- Software Architecture: Perspectives on an Emerging Discipline. M. Shaw, et al., Prentice-Hall, 1996.
- Refactoring: Improving the Design of Existing Code. M. Fowler, et al., Addison-Wesley, 1999.
- Software Testing and Analysis: Process, Principles and Techniques. M. Pezze, Wiley. 2007.

additional information

This module is identical in content to the German-language module *Softwarepraktikum*.

Module Category 5

Seminars

Proseminar

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
3	**6**	**every semester**	**1 semester**	**2**	**5**

responsible Dean of Studies of the Faculty of Mathematics and Computer Science
Dean of Studies of the Department of Computer Science

lecturers Lecturers of the department

entrance requirements Basic knowledge of the relevant sub-field of the study program.

assessments / exams
- Thematic presentation with subsequent discussion
- Active participation in the discussion
- short written report and/or project possible

course types / weekly hours `2 h proseminar`

total workload
```
  30 h of lectures and exercises
+ 120 h project work
= 150 h (= 5 ECTS)
```

grade Will be determined from the performance in the presentation and the written report and/or the seminar project. The exact modalities will be announced by the respective instructor.

language English or German

aims / competences to be developed

At the end of the proseminar, students have gained a basic understanding of current or fundamental aspects of a specific subfield of computer science.

In particular, they have gained basic competence in independent scientific research, classification, summarization, discussion, criticism and presentation of scientific findings.

Compared to the seminar, the focus of the proseminar is on the acquisition of basic scientific working methods.

content

With guidance, the following will be practiced hands-on:

- Reading and understanding scientific papers
- Discussion of the scientific work in the group
- Analyzing, summarizing and reporting the specific topic
- Presentation techniques

Specific in-depth study related to the individual topic of the seminar.

The typical procedure of a proseminar is usually as follows:

- Preparatory discussions for topic selection
- Regular meetings with discussion of selected contributions
- if applicable, work on a project related to the topic
- Presentation and, if necessary, writing a report on one of the presentations

literature & reading

Material is selected according to the topic.

additional information

The proseminars available will be announced prior to the beginning of the semester and will vary by study programme.

Seminar

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4	**6**	**every semester**	**1 semester**	**2**	**7**

responsible Dean of Studies of the Faculty of Mathematics and Computer Science
Dean of Studies of the Department of Computer Science

lecturers Lecturers of the department

entrance requirements Basic knowledge of the relevant sub-field of the study program.

assessments / exams
- Thematic presentation with subsequent discussion
- Active participation in the discussion
- short written report and/or project possible

course types / weekly hours `2 h seminar (weekly)`

total workload
```
  30 h of lectures and exercises
+ 180 h project work
= 210 h (= 7 ECTS)
```

grade Will be determined from the performance in the presentation and the written report and/or the seminar project. The exact modalities will be announced by the respective instructor.

language English or German

aims / competences to be developed

At the end of the seminar, students have primarily gained a deep understanding of current or fundamental aspects of a specific subfield of computer science.

They have gained further competence in independent scientific research, classifying, summarizing, discussing, criticizing and presenting scientific findings.

content

Largely independent research of the seminar topic:

- Reading and understanding of scientific papers
- Analysis and evaluation of scientific papers
- Discussion of the scientific work in the group
- Analyzing, summarizing and reporting the specific topic
- Developing common standards for scientific work
- Presentation techniques

Specific in-depth study related to the individual topic of the seminar.

The typical procedure of a seminar is usually as follows:

- Preparatory discussions for topic selection
- Regular meetings with discussion of selected presentations
- if applicable, work on a project related to the topic
- Presentation and, if necessary, writing a report on one of the presentations

literature & reading

Material is selected according to the topic.

additional information

The seminars available will be announced prior to the beginning of the semester and will vary by study programme.

Module Category 6

Core Lectures

Algorithms and Data Structures — AlgoDat

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Kurt Mehlhorn

lecturers Prof. Dr. Raimund Seidel
Prof. Dr. Kurt Mehlhorn

entrance requirements For graduate students: C, C++, Java

assessments / exams
- Regular attendance of classes and tutorials
- Passing the midterm and the final exam
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The students know standard algorithms for typical problems in the area's graphs, computational geometry, strings and optimization. Furthermore, they master a number of methods and data-structures to develop efficient algorithms and analyze their running times.

content

- graph algorithms (shortest path, minimum spanning trees, maximal flows, matchings, etc.)
- computational geometry (convex hull, Delaunay triangulation, Voronoi diagram, intersection of line segments, etc.)
- strings (pattern matching, suffix trees, etc.)
- generic methods of optimization (tabu search, simulated annealing, genetic algorithms, linear programming, branch-and-bound, dynamic programming, approximation algorithms, etc.)
- data-structures (Fibonacci heaps, radix heaps, hashing, randomized search trees, segment trees, etc.)
- methods for analyzing algorithms (amortized analysis, average-case analysis, potential methods, etc.

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Artificial Intelligence — AI

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Jörg Hoffmann

lecturers Prof. Dr. Jörg Hoffmann

entrance requirements *Programming 1*, *Programming 2*, *Fundamentals of Data Structures and Algorithms*, and *Elements of Machine Learning* or other courses in machine learning are recommended.

assessments / exams
- Regular attendance of classes and tutorials
- Solving of weekly assignments
- Passing the final written exam
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from the performance in exams. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Knowledge about basic methods in Artificial Intelligence

content

Search:

- Uninformed- and informed search procedures
- Monte-Carlo tree search

Planning:

- Formalism and complexity
- Critical-path heuristics
- Delete relaxation heuristics
- Abstraction heuristics

Markov decision processes:

- Discounted reward and expected cost
- Value iteration
- Informed search
- Reinforcement learning

Games:

- Adversarial search
- Learning from self-play

literature & reading

Russel & Norvig Artificial Intelligence: A Modern Approach;
further reading will be announced before the start of the course on the course page on the Internet.

Automated Reasoning AR

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Christoph Weidenbach

lecturers Prof. Dr. Christoph Weidenbach

entrance requirements *Introduction to Computational Logic*

assessments / exams
- Regular attendance of classes and tutorials
- Weekly assignments
- Practical work with systems
- Passing the final and mid-term exam
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The goal of this course is to provide familiarity with logics, calculi, implementation techniques, and systems providing automated reasoning.

content

Propositional Logic – CDCL, Superposition - Watched Literals
First-Order Logic without Equality – (Ordered) Resolution,
Equations with Variables – Completion, Termination
First-Order Logic with Equality – Superposition (SUP) - Indexing

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Compiler Construction CC

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Sebastian Hack

lecturers Prof. Dr. Sebastian Hack

entrance requirements For graduate students: none

assessments / exams
- Regular attendance of classes and tutorials
- Written exam at the end of the course, theoretical exercises, and compiler-laboratory project.
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The students learn, how a source program is lexically, syntactically, and semantically analyzed, and how they are translated into semantically equivalent machine programs. They learn how to increase the efficiency by semantics-preserving transformations. They understand the automata-theoretic foundations of these tasks and learn, how to use the corresponding tools.

content

Lexical, syntactic, semantic analysis of source programs, code generation for abstract and real machines, efficiency-improving program transformations, foundations of program analysis.

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Complexity Theory CT

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Markus Bläser

lecturers Prof. Dr. Raimund Seidel
Prof. Dr. Markus Bläser

entrance requirements undergraduate course on theory of computation (e.g. *Grundzüge der Theoretischen Informatik*) is highly recommend.

assessments / exams
- Regular attendance of classes and tutorials
- assignments
- exams (written or oral)

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be calculated from the results in the assignments and/or exams, as announced by the Lecturer at the beginning of the course

language English

aims / competences to be developed

The aim of this lecture is to learn important concepts and methods of computational complexity theory. The student shall be enabled to understand recent topics and results in computational complexity theory.

content

Relation among resources like time, space, determinism, nondeterminism, complexity classes, reduction and completeness, circuits and nonuniform complexity classes, logarithmic space and parallel complexity classes, Immerman-Szelepcsenyi theorem, polynomial time hierarchy, relativization, parity and the polynomial methods, Valiant-Vazirani theorem, counting problems and classes, Toda's theorem, probabilistic computations, isolation lemma and parallel algorithms for matching, circuit identity testing, graph isomorphism and interactive proofs.

literature & reading

Arora, Barak: Computational Complexity – A Modern Approach, Cambridge University Press
Oded Goldreich: Computational Complexity – A Conceptual Approach, Cambridge University Press
Dexter Kozen: Theory of Computation, Springer
Schöning, Pruim: Gems of Theoretical Computer Science, Springer

Computer Algebra CA

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Frank-Olaf Schreyer

lecturers Prof. Dr. Frank-Olaf Schreyer

entrance requirements For graduate students: none

assessments / exams
- Regular attendance of classes and tutorials
- Solving the exercises, passing the midterm and the final exam.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Solving problems occuring in computer algebra praxis
The theory behind algorithms

content

Arithmetic and algebraic systems of equations in geometry, engineering and natural sciences

- integer and modular arithmetics, prime number tests
- polynomal arithmetics and factorization
- fast Fourier-transformation, modular algorithms
- resultants, Gröbnerbasen
- homotopy methods for numerical solving
- real solutions, Sturm chains and other rules for algebraic signs Arithmetic and algebraic systems of equations in geometry, engineering and natural sciences
- integer and modular arithmetics, prime number tests
- polynomal arithmetics and factorization
- fast Fourier-transformation, modular algorithms
- resultants, Gröbnerbasen
- homotopy methods for numerical solving
- real solutions, Sturm chains and other rules for algebraic signs

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Computer Graphics CG

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Philipp Slusallek

lecturers Prof. Dr. Philipp Slusallek

entrance requirements Solid knowledge of linear algebra is recommended.

assessments / exams
- Successful completion of weekly exercises (30% of final grade)
- Successful participation in rendering competition (10%)
- Mid-term written exam (20%, final exam prerequisite)
- Final written exam (40%)
- In each of the above a minimum of 50% is required to pass

A re-exam typically takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade The grade is derived from the above assessments. Possible changes will be announced at the beginnning of each semester.

language English

aims / competences to be developed

This course provides the theoretical and practical foundation for computer graphics. It gives a wide overview of topics, techniques, and approaches used in various aspects of computer graphics but has some focus on image synthesis or rendering. The first part of the course uses ray tracing as a driving applications to discuss core topics of computer graphics, from vector algebra all the way to sampling theory, the human visual system, sampling theory, and spline curves and surfaces. A second part then uses rasterization approach as a driving example, introducing the camera transformation, clipping, the OpenGL API and shading langue, plus advanced techniques.

As part of the practical exercises the students incrementally build their own ray tracing system. Once the basics have been covered, the students participate in a rendering competition. Here they can implement their favorite advanced algorithm and are asked to generate a high-quality rendered image that shows their techniques in action.

content

- Introduction
- Overview of Ray Tracing and Intersection Methods
- Spatial Index Structures
- Vector Algebra, Homogeneous Coordinates, and Transformations
- Light Transport Theory, Rendering Equation
- BRDF, Materials Models, and Shading
- Texturing Methods
- Spectral Analysis, Sampling Theory
- Filtering and Anti-Aliasing Methods

- Recursive Ray Tracing & Distribution Ray-Tracing
- Human Visual System & Color Models
- Spline Curves and Surfaces
- Camera Transformations & Clipping
- Rasterization Pipeline
- OpenGL API & GLSL Shading
- Volume Rendering (opt.)

literature & reading

Will be announced in the lecture.

Continuous Optimization — OPT

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Peter Ochs

lecturers Prof. Dr. Peter Ochs

entrance requirements Undergraduate mathematics (e.g. *Mathematik für Informatiker I*, *II* and *III*) and some elementary programming knowledge is recommended.

assessments / exams
- Regular attendance of classes and tutorials
- Solving accompanying exercises
- Successful partcipation in the final or re-exam

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

After taking this course, students will have an overview of classical optimization methods and analysis tools for continuous optimization problems, which allows them to model and solve practical problems. Moreover, in the tutorials, some experience will be gained to implement and numerically solve practical problems.

content

1. Introduction
   - Mathematical Optimization
   - Applications
   - Performance of Numerical Methods
   - Existence of a Solution
   - The Class of Convex Optimization Problems
2. Unconstrained Optimization
   - Optimality Conditions
   - Descent Methods
   - Gradient Descent Method
   - Conjugate Gradient Method
   - Newton's Method
   - Quasi-Newton Methods
   - Gauss-Newton Method
   - Computing Derivatives
3. Constrained Optimization
   - Motivation

- Optimality Conditions for Constrained Problems
- Method of Feasible Directions
- Linear Programming
- Quadratic Programming
- Sequential Quadratic Programming (SQP)
- Penalty and Barrier Methods

literature & reading

- J. Nocedal und S. J. Wright: Numerical Optimization. Springer, 2006.
- F. Jarre und J. Stoerr: Optimierung. Springer, 2004.
- D. Bertsekas: Nonlinear Programming. Athena Scientific, 1999.
- Y. Nesterov: Introductory Lectures on Convex Optimization - A Basic Course. Kluwer Academic Publisher, 2004.
- T. Rockafellar and R. J.-B. Wets: Variational Analysis. Springer-Verlag Berlin Heidelberg, 1998.

Convex Analysis and Optimization — CAO

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Peter Ochs

lecturers Prof. Dr. Peter Ochs

entrance requirements Undergraduate mathematics (e.g. *Mathematik für Informatiker I*, *II* and *III*) and some elementary programming knowledge is recommended.

assessments / exams
- Regular attendance of classes and tutorials
- Solving accompanying exercises
- Successful participation in the final or re-exam

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

After taking the course, students know about the most relevant concepts of convex analysis and convex optimization. They are able to read and understand related scientific literature. Moreover, they can rate the difficulty of convex optimization problems arising in applications in machine learning or computer vision and select an efficient algorithm accordingly. Moreover, they develop basic skills in solving practical problems with Python.

content

1. Introduction
   - Introduction
   - Applications
2. Convex Geometry
   - Foundations
   - Convex Feasibility Problems
3. Convex Analysis Background
   - Preliminaries
   - Convex Functions
4. Smooth Convex Optimization
   - Optimality Conditions
   - Gradient Descent Method
   - Lower complexity bounds
   - Accelerated and Inertial Algorithms

5. Non-smooth Convex Analysis
    - Continuity of Convex Functions
    - Convexity from Epigraphical Operations
    - The Subdifferential
6. Non-smooth Convex Optimization
    - Fermat's Rule
    - Duality in Optimization and Primal / Dual Problems
    - Algorithms
    - Lower complexity bounds
    - Saddle Point Problems

literature & reading

- T. Rockafellar: Convex Analysis. Princeton University Press, 1970.
- Y. Nesterov: Introductory Lectures on Convex Optimization: A Basic Course. Kluwer Academic Publishers, 2004.
- D.P. Bertsekas: Convex Analysis and Optimization. Athena Scientific, 2003.
- S. Boyd: Convex Optimization. Cambridge Univeristy Press, 2004.
- H. H. Bauschke and P. L. Combettes: Convex Analysis and Monotone Operator Theory in Hilbert Spaces. Springer, 2011.
- T. Rockafellar and R. J.-B. Wets: Variational Analysis. Springer-Verlag Berlin Heidelberg, 1998.

Cryptography — Crypto

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Dr. Nico Döttling

lecturers Prof. Dr. Cas Cremers
Dr. Nico Döttling
Dr. Antoine Joux
Dr. Lucjan Hanzlik
Dr. Julian Loss

entrance requirements For graduate students: Basic knowledge in theoretical computer science required, background knowledge in number theory and complexity theory helpful

assessments / exams
- Oral / written exam (depending on the number of students)
- A re-exam is normally provided (as written or oral examination).

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The students will acquire a comprehensive knowledge of the basic concepts of cryptography and formal definitions. They will be able to prove the security of basic techniques.

content

- Symmetric and asymmetric encryption
- Digital signatures and message authentication codes
- Information theoretic and complexity theoretic definitions of security, cryptographic reduction proofs
- Cryptographic models, e.g. random oracle model
- Cryptographic primitives, e.g. trapdoor-one-way functions, pseudo random generators, etc.
- Cryptography in practice (standards, products)
- Selected topics from current research

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Cyber-Physical Systems

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Martina Maggio

lecturers Prof. Dr. Martina Maggio

entrance requirements none

assessments / exams
- Written exam at the end of the course.
- A re-exam takes place before the start of the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorials
= 6 h (weekly)
```

total workload
```
   75 h lectures
+  15 h mandatory assignments
+ 180 h individual study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams and assignments. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

By completing the Cyber-Physical Systems course, students will acquire the ability to model, analyze, control, and implement embedded systems that interact with the physical world, equipping them to design reliable and efficient systems for a variety of applications in modern technology.

content

Cyber-Physical Systems are embedded systems that integrate computation with physical processes. These systems are ubiquitous in our daily lives, powering technologies such as smart watches, household appliances, mobile phones, and automotive control systems. In fact, the majority of modern computing devices are embedded systems, with an estimated 98% of new CPUs being embedded in larger systems.

This course provides a comprehensive foundation for understanding, designing, and programming cyber-physical systems, emphasizing their theoretical and practical aspects. It is structured into three interconnected parts:

1. *Models:* Students will learn how to represent the physical systems that embedded systems interact with, exploring dynamical systems in both continuous and discrete time. Additionally, the course will briefly introduce more advanced models, which combine discrete state systems with dynamical systems.
2. *Control:* This module focuses on principles for modifying the behavior of physical systems through computation. Students will study and apply control techniques such as state feedback and PID control, learning how these methods influence the interaction between embedded systems and their environments.
3. *Implementation:* The final course part addresses practical challenges in embedded systems programming. Topics include scheduling, communication, and fault tolerance. This ensures that students are equipped to implement robust and efficient embedded systems in real-world scenarios.

By the end of this course, students will possess the skills needed to design and implement cyber-physical systems that meet specific functional and performance requirements, preparing them for roles in cutting-edge industries where embedded systems play a critical role, such as the automotive industry and for research in the cyber-physical systems domain.

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module was formerly also known as *Embedded Systems*.

Data Networks DN

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr.-Ing. Holger Hermanns

lecturers Prof. Dr.-Ing. Holger Hermanns
Prof. Dr. Anja Feldmann

entrance requirements For graduate students: none

assessments / exams
- Regular attendance of classes and tutorials
- Qualification for final exam through mini quizzes during classes
- Possibility to get bonus points through excellent homework
- Final exam
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

After taking the course students have

- a thorough knowledge regarding the basic principles of communication networks,
- the fundamentals of protocols and concepts of protocol,
- Insights into fundamental motivations of different pragmatics of current network solutions,
- Introduction to practical aspects of data networks focusing on internet protocol hierarchies

content

Introduction and overview

Cross section:

- Stochastic Processes, Markov models,
- Fundamentals of data network performance assessment
- Principles of reliable data transfer
- Protokols and their elementary parts
- Graphs and Graphalgorithms (maximal flow, spanning tree)
- Application layer:
- Services and protocols
- FTP, Telnet
- Electronic Mail (Basics and Principles, SMTP, POP3, ..)
- World Wide Web (History, HTTP, HTML)

- Transport Layer:
- Services and protocols
- Addressing
- Connections and ports
- Flow control
- QoS
- Transport Protocols (UDP, TCP, SCTP, Ports)
- Network layer:
- Services and protocols
- Routing algorithms
- Congestion Control
- Addressing
- Internet protocol (IP)
- Data link layer:
- Services and protocols
- Medium access protocols: Aloha, CSMA (-CD/CA), Token passing
- Error correcting codes
- Flow control
- Applications: LAN, Ethernet, Token Architectures, WLAN, ATM
- Physical layer
- Peer-to-Peer and Ad-hoc Networking Principles

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Database Systems DBS

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Jens Dittrich

lecturers Prof. Dr. Jens Dittrich

entrance requirements especially Saarland University CS department's undergraduate lecture *Big Data Engineering* (former *Informationssysteme*), *Programmierung 1* and *2*, *Algorithmen und Datenstrukturen* as well as *Nebenläufige Programmierung*

For graduate students:

- motivation for databases and database management systems;
- the relational data model;
- relational query languages, particularly relational algebra and SQL;
- **solid** programming skills in Java and/or C++
- undergrad courses in algorithms and data structures, concurrent programming

assessments / exams

- Passing a two-hour written exam at the end of the semester
- Successful demonstration of programming project (teams of up to three students are allowed); the project may be integrated to be part of the weekly assignments

Grades are based on written exam; 50% in weekly assignments (in paper and additionally paper or electronic quizzes) must be passed to participate in the final and repetition exams.

A repetition exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

This class may be run as a flipped classroom, i.e. 2 hours of lectures may be replaced by self-study of videos/papers; the other 2 hours may be used to run a group exercice supervised by the professor called "the LAB")

total workload

```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined based on project, midterm and best of endterm and reexam.

language English

aims / competences to be developed

Database systems are the backbone of most modern information systems and a core technology without which today's economy – as well as many other aspects of our lifes – would be impossible in their present forms. The course teaches the architectural and algorithmic foundations of modern database management systems (DBMS), focussing on database systems internals rather than applications. Emphasis is made on robust and time-tested techniques that have led databases to be considered a mature technology and one of the greatest success stories in computer science. At the same time, opportunities for exciting research in this field will be pointed out.

In the exercise part of the course, important components of a DBMS will be treated and where possible implemented and their performance evaluated. The goal this is to work with the techniques introduced in the lecture and to understand them and their practical implications to a depth that would not be attainable by purely theoretical study.

content

The course ”Database Systems” will introduce students to the internal workings of a DBMS, in particular:

- storage media (disk, flash, main memory, caches, and any other future storage medium)
- data managing architectures (DBMS, streams, file systems, clouds, appliances)
- storage management (DB-file systems, raw devices, write-strategies, differential files, buffer management)
- data layouts (horizontal and vertical partitioning, columns, hybrid mappings, compression, defragmentation)
- indexing (one- and multidimensional, tree-structured, hash-, partition-based, bulk-loading and external sorting, differential indexing, read- and write-optimized indexing, data warehouse indexing, main-memory indexes, sparse and dense, direct and indirect, clustered and unclustered, main memory versus disk and/or flash-based)
- processing models (operator model, pipeline models, push and pull, block-based iteration, vectorization, query compilation)
- processing implementations (join algorithms for relational data, grouping and early aggregation, filtering)
- query processing (scanning, plan computation, SIMD)
- query optimization (query rewrite, cost models, cost-based optimization, join order, join graph, plan enumeration)
- data recovery (single versus multiple instance, logging, ARIES)
- parallelization of data and queries (horizontal and vertical partitioning, shared-nothing, replication, distributed query processing, NoSQL, MapReduce, Hadoop and/or similar and/or future systems)
- read-optimized system concepts (search engines, data warehouses, OLAP)
- write-optimized system concepts (OLTP, streaming data)
- management of geographical data (GIS, google maps and similar tools)
- main-memory techniques

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Digital Signal Processing

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**every summer semester**	**1 semester**	**4**	**6**

responsible Prof. Dr. Dietrich Klakow

lecturers Prof. Dr. Dietrich Klakow

entrance requirements Sound knowledge of mathematics as taught in engineering, computer science or physics is recommended.

assessments / exams Final exam

course types / weekly hours
```
  2 h lecture
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
  60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade The grade is determined by result of the final exam. A re-exam takes place half a year after the first exam.

language English

aims / competences to be developed

The students will get familiar with advanced signal processing techniques in particular those that are relevant to speech processing. There will be practical and theoretical exercises.

content

1. Introduction
2. Signal Representations
3. Filtering and Smoothing
4. Linear Predictive Coding
5. Microphone Arrays
6. Object Tracking and the Kalman-Filter
7. Wiener Filter
8. Feature Extraction from Audio Signals
9. KL-Transform and Linear Discriminant Analysis
10. Basics of Classification
11. Speaker Recognition
12. Musical Genre Classification

literature & reading

- Dietrich W. R. Paulus, Joachim Hornegger “Applied Pattern Recognition”, Vieweg
- Peter Vary, Ulrich Heute, Wolfgang Hess “Digitale Sprachsignalverarbeitung”, Teubner Verlag
- Xuedong Huang, Hsiao-Wuen Hon “Spoken Language Processing”, Prentice Hall
- C. Bishop „Pattern Recognition and Machine Learning“, Springer

Further reading will be announced in the lecture.

Distributed Systems DS

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Peter Druschel, Ph.D.

lecturers Prof. Peter Druschel, Ph.D.
Allen Clement, Ph.D

entrance requirements *Operating Systems* or *Concurrent Programming*

assessments / exams
- Regular attendance at classes and tutorials.
- Successful completion of a course project in teams of 2 students. (Project assignments due approximately every 2 weeks.)
- Passing grade on 2 out of 3 written exams: midterm, final exam, and a re-exam that takes place during the last two weeks before the start of lectures in the following semester.
- Final course grade: 50% project, 50% best 2 out of 3 exams.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Introduction to the principles, design, and implementation of distributed systems.

content

- Communication: Remote procedure call, distributed objects, event notification, Inhalt dissemination, group communication, epidemic protocols.
- Distributed storage systems: Caching, logging, recovery, leases.
- Naming. Scalable name resolution.
- Synchronization: Clock synchronization, logical clocks, vector clocks, distributed snapshots.
- Fault tolerance: Replication protocols, consistency models, consistency versus availability trade-offs, state machine replication, consensus, Paxos, PBFT.
- Peer-to-peer systems: consistent hashing, self-organization, incentives, distributed hash tables, Inhalt distribution networks.
- Data centers. Architecture and infrastructure, distributed programming, energy efficiency.

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Geometric Modelling GM

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Hans-Peter Seidel

lecturers Prof. Dr. Hans-Peter Seidel
Dr. Rhaleb Zayer

entrance requirements calculus and basic programming skills

assessments / exams
- Regular attendance and participation.
- Weekly Assignments (10% bonus towards the course grade; bonus points can only improve the grade; they do not affect passing)
- Passing the written exams (mid-term and final exam).
- The mid-term and the final exam count for 50% each, but 10% bonus from assignments will be added.
- A re-exam takes place at the end of the semester break or early in the next semester.

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

Practical assignments in groups of 3 students (practice)
Tutorials consists of a mix of theoretical + practical assignments.

total workload

```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be based on the performance in exams, exercises and practical tasks. The detailed terms will be announced by the module coordinator.

language English

aims / competences to be developed

Gaining knowledge of the theoretical aspect of geometric modelling problems, and the practical solutions used for modelling and manipulating curves and surfaces on a computer. From a broader perspective: Learning how to represent and interact with geometric models in a discretized, digital form (geometric representations by functions and samples; design of linear function spaces; finding “good” functions with respect to a geometric modelling task in such spaces).

content

- Differential geometry Fundamentals
- Interpolation and Approximation
- Polynomial Curves
- Bezier and Rational Bezier Curves
- B-splines, NURBS
- Spline Surfaces
- Subdivision and Multiresolution Modelling
- Mesh processing
- Approximation of differential operators
- Shape Analysis and Geometry Processing

literature & reading

Will be announced before the term begins on the lecture website.

Human Computer Interaction — HCI

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Jürgen Steimle

lecturers Prof. Dr. Jürgen Steimle

entrance requirements undergraduate students: *Programmierung 1* and *2*
graduate students: none

assessments / exams Regular attendance of classes and tutorials
Successful completion of exercises and course project
Final exam
A re-exam takes place (as written or oral examination).

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload

```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

This course teaches the theoretical and practical foundations for human computer interaction. It covers a wide overview of topics, techniques and approaches used for the design and evaluation of modern user interfaces.

The course covers the principles that underlie successful user interfaces, provides an overview of input and output devices and user interface types, and familiarizes students with the methods for designing and evaluating user interfaces. Students learn to critically assess user interfaces, to design user interfaces themselves, and to evaluate them in empirical studies.

content

- Fundamentals of human-computer interaction
- User interface paradigms, input and output devices
- Desktop & graphical user interfaces
- Mobile user interfaces
- Natural user interfaces
- User-centered interaction design
- Design principles and guidelines
- Prototyping

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Image Processing and Computer Vision — IPCV

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Joachim Weickert

lecturers Prof. Dr. Joachim Weickert

entrance requirements Undergraduate mathematics (e.g. Mathematik für Informatiker I-III) and elementary programming knowledge in C

assessments / exams
- For the homework assignments one can obtain up to 24 points per week. Actively participating in the classroom assignments gives 12 more points per week, regardless of the correctness of the solutions. To qualify for both exams one needs 2/3 of all possible points.
- Passing the final exam or the re-exam.
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from the performance in the exam or the re-exam. The better grade counts.

language English

aims / competences to be developed

Broad introduction to mathematical methods in image processing and computer vision. The lecture qualifies students for a bachelor thesis in this field. Together with the completion of advanced or specialised lectures (9 credits at least) it is the basis for a master thesis in this field.

content

```
Inhalt

1. Basics
   1.1 Image Types and Discretisation
   1.2 Degradations in Digital Images
2. Colour Perception and Colour Spaces
3. Image Transformations
   3.1 Continuous Fourier Transform
   3.2 Discrete Fourier Transform
   3.3 Image Pyramids
   3.4 Wavelet Transform
4. Image Compression
5. Image Interpolation
6. Image Enhancement
   6.1 Point Operations
```

```
   6.2 Linear Filtering and Feature Detection
   6.3 Morphology and Median Filters
   6.3 Wavelet Shrinkage, Bilateral Filters, NL Means
   6.5 Diffusion Filtering
   6.6 Variational Methods
   6.7 Deconvolution Methods
7. Texture Analysis
8. Segmentation
   8.1 Classical Methods
   8.2 Variational Methods
9. Image Sequence Analysis
   9.1 Local Methods
   9.2 Variational Methods
10. 3-D Reconstruction
   10.1 Camera Geometry
   10.2 Stereo
   10.3 Shape-from-Shading
11. Object Recognition
   11.1 Hough Transform
   11.2 Invariants
   11.3 Eigenspace Methods
```

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Information Retrieval and Data Mining — IRDM

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Gerhard Weikum

lecturers Prof. Dr. Gerhard Weikum

entrance requirements Good knowledge of undergraduate mathematics (linear algebra, probability theory) and basic algorithms.

assessments / exams
- Regular attendance of classes and tutor groups
- Presentation of solutions in tutor groups
- Passing 2 of 3 written tests (after each third of the semester)
- Passing the final exam (at the end of the semester)

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined by the performance in written tests, tutor groups, and the final exam. Details will be announced on the course web site.

language English

aims / competences to be developed

The lecture teaches models and algorithms that form the basis for search engines and for data mining and data analysis tools.

content

Information Retrieval (IR) and Data Mining (DM) are methodologies for organizing, searching and analyzing digital Inhalts from the web, social media and enterprises as well as multivariate datasets in these contexts. IR models and algorithms include text indexing, query processing, search result ranking, and information extraction for semantic search. DM models and algorithms include pattern mining, rule mining, classification and recommendation. Both fields build on mathematical foundations from the areas of linear algebra, graph theory, and probability and statistics.

literature & reading

Will be announced on the course web site.

Introduction to Computational Logic — ICL

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Gert Smolka

lecturers Prof. Dr. Gert Smolka

entrance requirements none

assessments / exams
- Regular attendance of classes and tutorials.
- Passing the midterm and the final exam.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

- structure of logic languages based on type theory
- distinction notation / syntax / semantics
- structure and formal representation of mathematical statements
- structure and formal representation of proofs (equational and natural deduction)
- solving Boolean equations
- proving formulas with quantifiers
- implementing syntax and deduction

content

Type Theory:

- functional representation of mathematical statements
- simply typed lambda calculus, De Bruijn representation and substitution, normalization, elimination of lambdas
- Interpretations and semantic consequence
- Equational deduction, soundness and completeness
- Propositional Logic
- Boolean Axioms, completeness for 2-valued interpretation
- resolution of Boolean equations, canonical forms based on decision trees and resolution

Predicate Logic (higher-order):

- quantifier axioms
- natural deduction
- prenex and Skolem forms

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Machine Learning — ML

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Isabel Valera

lecturers Prof. Dr. Isabel Valera

entrance requirements The lecture gives a broad introduction into machine learning methods. After the lecture the students should be able to solve and analyze learning problems.

assessments / exams
- Regular attendance of classes and tutorials.
- 50% of all points of the exercises have to be obtained in order to qualify for the exam.
- Passing 1 out of 2 exams (final, re-exam).

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Determined from the results of the exams, exercises and potential projects. The exact grading modalities are announced at the beginning of the course.

language English

aims / competences to be developed

The lecture gives a broad introduction into machine learning methods. After the lecture the students should be able to solve and analyze learning problems.

content

- Bayesian decision theory
- Linear classification and regression
- Kernel methods
- Bayesian learning
- Semi-supervised learning
- Unsupervised learning
- Model selection and evaluation of learning methods
- Statistical learning theory
- Other current research topics

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Operating Systems — OS

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Peter Druschel, Ph.D.

lecturers Prof. Peter Druschel, Ph.D.
Björn Brandenburg, Ph.D

entrance requirements For graduate students: none

assessments / exams Regular attendance at classes and tutorials
Successful completion of a course project in teams of 2 students
Passing 2 written exams (midterm and final exam)
A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload

```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Introduction to the principles, design, and implementation of operating systems

content

Process management:

- Threads and processes, synchronization
- Multiprogramming, CPU Scheduling
- Deadlock

Memory management:

- Dynamic storage allocation
- Sharing main memory
- Virtual memory

I/O management:

- File storage management
- Naming
- Concurrency, Robustness, Performance

Virtual machines

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Optimization — Opti

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Kurt Mehlhorn

lecturers Prof. Dr. Kurt Mehlhorn
Dr. Andreas Karrenbauer

entrance requirements For graduate students: none

assessments / exams
- Regular attendance of classes and tutorials
- Solving accompanying exercises, successful partcipation in midterm and final exam
- Grades: Yes
- The grade is calculated from the above parameters according to the following scheme: 20%, 30%, 50%
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The students learn to model and solve optimization problems from theory as from the real world

content

Linear Programming: Theory of polyhedra, simplex algorithm, duality, ellipsoid method * Integer linear programming: Branch-and-Bound, cutting planes, TDI-Systems * Network flow: Minimum cost network flow, minimum mean cycle cancellation algorithm, network simplex method * Matchings in graphs: Polynomial matching algorithms in general graphs, integrality of the matching polytope, cutting planes * Approximation algorithms: LP-Rounding, greedy methods, knapsack, bin packing, steiner trees and forests, survivable network design

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Security — Sec

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Michael Backes

lecturers Prof. Dr. Michael Backes
Prof. Dr. Cas Cremers

entrance requirements For graduate students: none

assessments / exams
- Regular attendance of classes and tutorials
- Passing the final exam
- A re-exam is normally provided (as written or oral examination).

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined by the performance in exams, tutor groups, and practical tasks. Details will be announced by the lecturer at the beginning of the course.

language English

aims / competences to be developed

Description, assessment, development and application of security mechanisms, techniques and tools.

content

- Basic Cryptography,
- Specification and verification of security protocols,
- Security policies: access control, information flow analysis,
- Network security,
- Media security,
- Security engineering

literature & reading

Will be announced on the course website

Semantics

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Gert Smolka

lecturers Prof. Dr. Gert Smolka

entrance requirements For graduate students: core lecture Introduction to Computational Logic

assessments / exams
- Regular attendance of classes and tutorials.
- Passing the midterm and the final exam

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Understanding of

- Logical structure of programming languages
- Formal models of programming languages
- Type and module systems for programming languages

content

Theory of programming languages, in particular:

- Formal models of functional and object-oriented languages
- Lambda Calculi (untyped, simply typed, System F, F-omega, Lambda Cube, subtyping, recursive types, Curry-Howard Correspondence)
- Algorithms for type checking and type reconstruction

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Software Engineering SE

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Sven Apel

lecturers Prof. Dr. Sven Apel

entrance requirements
- Knowledge of programming concepts (as taught in the lectures *Programmierung 1* and *Programmierung 2*)
- Basic knowledge of software processes, design, and testing (as taught and applied in the lecture *Softwarepraktikum*)

assessments / exams Beside the lecture and weekly practical exercises, there will be a number of assignments in the form of mini-projects for each student to work on (every two to three weeks). The assignments will be assessed based on the principles covered in the lecture. Passing all assignments is a prerequisite for taking the final written exam. The final grade is determined only by the written exam. Further examination details will be announced by the lecturer at the beginning of the course. In short:
- Passing all assignments (prerequisite for the written exam)
- Passing the written exam

course types / weekly hours
```
  4 h lectures
+ 2 h exercises
= 6 h (weekly)
```

total workload
```
   90 h of classes and exercises
+ 180 h private study and assignments
= 270 h (= 9 ECTS)
```

grade The grade is determined by the written exam. Passing all assignments is a prerequisite for taking the written exam. The assignments do not contribute to the final grade. Further examination details will be announced by the lecturer at the beginning of the course.

language English

aims / competences to be developed

- The students know and apply modern software development techniques.
- They are aware of key factors contributing to the complexity of real-world software systems, in particular, software variability, configurability, feature interaction, crosscutting concerns, and how to address them.
- They know how to apply established design and implementation techniques to master software complexity.
- They are aware of advanced design and implementation techniques, including collaboration-based design, mixins/traits, aspects, pointcuts, advice.
- They are aware of advanced quality assurance techniques that take the complexity of real-world software systems into account: variability-aware analysis, sampling, feature-interaction detection, predictive performance modeling, etc.
- They appreciate the role of non-functional properties and know how to predict and optimize software systems regarding these properties.
- They are able to use formal methods to reason about key techniques and properties covered in the lecture.

content

- Domain analysis, feature modeling
- Automated reasoning about software configuration using SAT solvers
- Runtime parameters, design patterns, frameworks
- Version control, build systems, preprocessors
- Collaboration-based design
- Aspects, pointcuts, advice
- Expression problem, preplanning problem, code scattering & tangling, tyranny of the dominant decomposition, inheritance vs. delegation vs. mixin composition
- Feature interaction problem (structural, control- & data-flow, behavioral, non-functional feature interactions)
- Variability-aware analysis and variational program representation (with applications to type checking and static program analysis)
- Sampling (random, coverage)
- Machine learning for software performance prediction and optimization

literature & reading

- Feature-Oriented Software Product Lines: Concepts and Implementation. S. Apel, et al., Springer, 2013.
- Generative Programming: Methods, Tools, and Applications: Methods, Techniques and Applications. K. Czarnecki, et al., Addison-Wesley, 2000.
- Mastering Software Variability with FeatureIDE. J. Meinicke, et al., Springer, 2017.

Verification Veri

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
4-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr.-Ing. Holger Hermanns

lecturers Prof. Dr.-Ing. Holger Hermanns
Prof. Bernd Finkbeiner, Ph.D

entrance requirements For graduate students: none

assessments / exams
- Regular attendance of classes and tutorials
- Passing the final exam
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The students become familiar with the standard methods in computer-aided verification. They understand the theoretical foundations and are able to assess the advantages and disadvantages of different methods for a specific verification project. The students gain first experience with manual correctness proofs and with the use of verification tools.

content

- models of computation and specification languages: temporal logics, automata over infinite objects, process algebra
- deductive verification: proof systems (e.g., Floyd, Hoare, Manna/Pnueli), relative completeness, compositionality
- model checking: complexity of model checking algorithms, symbolic model checking, abstraction case studies

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Module Category 7

Advanced Lectures

Audio-Visual Communication and Networks — AVCN

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr.-Ing. Thorsten Herfet

lecturers Prof. Dr.-Ing. Thorsten Herfet

entrance requirements Solid foundation of mathematics (differential and integral calculus) and probability theory. The course will build on the mathematical concepts and tools taught in TC I while trying to enable everyone to follow and to fill gaps by an accelerated study of the accompanying literature. *Signals and Systems* as well as *Digital Transmission and Signal Processing (TC I)* are strongly recommended but not required.

assessments / exams Regular attendance of classes and tutorials Passing the final exam
Oral exam directly succeeding the course. Eligibility: Weekly excersises / task sheets, grouped into two blocks corresponding to first and second half of the lecture. Students must provide min. 50% grade in each of the two blocks to be eligible for the exam.

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload

```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Final Exam Mark

language English

aims / competences to be developed

AVCN will deepen the students' knowledge on modern communications systems and will focus on wireless systems.

Since from a telecommunications perspective the combination of audio/visual data – meaning inherently high data rate and putting high requirements on the realtime capabilities of the underlying network – and wireless transmission – that is unreliable and highly dynamic with respect to the channel characteristics and its capacity – is the most demanding application domain.

content

As the basic principle the course will study and introduce the building blocks of wireless communication systems. Multiple access schemes like TDMA, FDMA, CDMA and SDMA are introduced, antennas and propagation incl. link budget calculations are dealt with and more advanced channel models like MIMO are investigated. Modulation and error correction technologies presented in Telecommunications I will be expanded by e.g. turbo coding and receiver architectures like RAKE and BLAST will be introduced. A noticeable portion of the lecture will present existing and future wireless networks and their extensions for audio/visual data. Examples include 802.11n and the terrestrial DVB system (DVB-T2).

literature & reading

Will be announced before the start of the course on the course page on the Internet.

additional information

This module was formerly also known as *Telecommunications II*.

Automata, Games and Verification — AGV

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**at least every two years**	**1 semester**	**4**	**6**

responsible Prof. Bernd Finkbeiner, Ph.D

lecturers Prof. Bernd Finkbeiner, Ph.D

entrance requirements none

assessments / exams
- Regular attendance of classes and tutorial
- Final exam
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The students will gain a deep understanding of the automata-theoretic background of automated verification and program synthesis.

content

The theory of automata over infinite objects provides a succinct, expressive and formal framework for reasoning about reactive systems, such as communication protocols and control systems. Reactive systems are characterized by their nonterminating behaviour and persistent interaction with their environment.

In this course we study the main ingredients of this elegant theory, and its application to automatic verification (model checking) and program synthesis.

- Automata over infinite words and trees (omega-automata)
- Infinite two-person games
- Logical systems for the specification of nonterminating behavior
- Transformation of automata according to logical operations

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Automated Debugging

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**at least every two years**	**1 semester**	**4**	**6**

responsible Prof. Dr. Andreas Zeller

lecturers Prof. Dr. Andreas Zeller

entrance requirements *Programmierung 1*, *Programmierung 2* and *Softwarepraktikum*

assessments / exams Projects and mini-tests

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload

```
  60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade The module is passed in its entirety if the examination performance has been passed.

language English

aims / competences to be developed

Finding and fixing software bugs can involve lots of effort. This course addresses this problem by automating software debugging, specifically identifying failure causes, locating bugs, and fixing them. Students learn the basics of systematic debugging, and explore tools and techniques for automated debugging.

content

- Tracking Problems
- The Scientific Method
- Cause-Effect Chains
- Building a Debugger
- Tracking Inputs
- Assertions and Sanitizers
- Detecting Anomalies
- Statistical Fault Localization
- Generating Tests
- Reducing Failure-Inducing Inputs
- Mining Software Archives
- Fixing the Defect
- Repairing Bugs Automatically
- Managing Bugs

literature & reading

The teaching material consists of text, Python code, and Jupyter Notebooks from the textbook “The Debugging Book” (https://www.debuggingbook.org/), also in English.

Correspondence Problems in Computer Vision — CoPCV

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional**	**1 semester**	**4**	**6**

responsible Prof. Dr. Joachim Weickert

lecturers Dr. Pascal Peter

entrance requirements Undergraduate mathematics (e.g. ”Mathematik für Informatiker I-III”) is required, as well as elementary C knowledge (for the programming assignments). Knowledge in image processing or differential equations is useful.

assessments / exams
- Regular attendance of lecture and tutorial
- Written or oral exam and the end of the course

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from performance in exams. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Correspondence problems are a central topic in computer vision. Thereby, one is interested in identifying and matching corresponding features in different images/views of the same scene. Typical corresondence problems are the estimation of motion information from consecutive frames of an image sequence (optic flow), the reconstruction of a 3-D scene from a stereo image pair and the registration of medical image data from different modalities (e.g. CT and MRT). Central part of this lecture is the discussion of the most important correspondence problems as well as the modelling of suitable algorithms for solving them.

content

1. Introduction and Overview
2. General Matching Concepts
   - 2.1 Block Matching
   - 2.2 Correlation Techniques
   - 2.3 Interest Points
   - 2.4 Feature-Based Methods
3. Optic Flow I
   - 3.1 Local Differential Methods
   - 3.2 Parameterisation Models
4. Optic Flow II
   - 4.1 Global Differential Methods
   - 4.2 Horn and Schunck
5. Optic Flow III
   - 5.1 Advanced Constancy Assumptions
   - 5.2 Large Motion

6. Optic Flow IV
    6.1 Robust Data Terms
    6.2 Discontinuity-Preserving Smoothness Terms
7. Optic Flow V
    7.1 High Accuracy Methods
    7.2 SOR and Lienar Multigrid
8. Stereo Matching I
    8.1 Projective Geometry
    8.2 Epipolar Geometry
9. Stereo Matching II
    9.1 Estimation of the Fundamental Matrix
10. Stereo Matching III
    10.1 Correlation Methods
    10.2 Variational Approaches
    10.3 Graph Cuts
11. Medical Image Registration
    11.1 Mutual Information
    11.2 Elastic and Curvature Based Registration
    11.3 Landmarks
12. Particle Image Velocimetry
    12.1 Div-Curl-Regularisation
    12.2 Incompressible Navier Stokes Prior

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Differential Equations in Image Processing and Computer Vision — DIC

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Joachim Weickert

lecturers Prof. Dr. Joachim Weickert

entrance requirements Undergraduate mathematics (e.g. "Mathematik für Informatiker I-III") and some elementary programming knowledge in C is required. Prior participation in "Image Processing and Computer Vision" is useful.

assessments / exams

- For the homework assignments one can obtain up to 24 points per week. Actively participating in the classroom assignments gives 12 more points per week, regardless of the correctness of the solutions. To qualify for both exams one needs 2/3 of all possible points.
- Passing the final exam or the re-exam.
- The re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours

```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

Homework assignments (theory and programming) and classroom assignments.

total workload

```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from the performance in the exam or the re-exam. The better grade counts.

language English

aims / competences to be developed

Many modern techniques in image processing and computer vision make use of methods based on partial differential equations (PDEs) and variational calculus. Moreover, many classical methods may be reinterpreted as approximations of PDE-based techniques. In this course the students will get an in-depth insight into these methods. For each of these techniques, they will learn the basic ideas as well as theoretical and algorithmic aspects. Examples from the fields of medical imaging and computer aided quality control will illustrate the various application possibilities.

content

1. Introduction and Overview
2. Linear Diffusion Filtering
   - 2.1 Basic Concepts
   - 2.2 Numerics
   - 2.3 Limitations and Alternatives
3. Nonlinear Isotropic Diffusion Filtering
   - 3.1 Modeling
   - 3.2 Continuous Theory
   - 3.2 Semidiscete Theory
   - 3.3 Discrete Theory
   - 3.4 Efficient Sequential and Parallel Algorithms

4. Nonlinear Anisotropic Diffusion Filtering
    4.1 Modeling
    4.2 Continuous Theory
    4.3 Discrete Aspects
    4.4 Efficient Algorithms
5. Parameter Selection
6. Variational Methods
    6.1 Basic Ideas
    6.2 Discrete Aspects
    6.3 TV Regularisation and Primal-Dual Methods
    6.4 Functionals of Two Variables
7. Vector- and Matrix-Valued Images
8. Unification of Denoising Methods
9. Osmosis
    9.1 Continuous Theory and Modelling
    9.2 Discrete Theory and Efficient Algorithms
10. Image Sequence Analysis
    10.1 Models for the Smoothness Term
    10.2 Models for the Data Term
    10.3 Practical Aspects
    10.4 Numerical Methods
11. Continuous-Scale Morphology
    11.1 Basic Ideas
    11.2 Shock Filters and Nonflat Morphology
12. Curvature-Based Morphology
    12.1 Mean Curvature Motion
    12.2 Affine Morphological Scale-Space
13. PDE-Based Image Compression
    13.1 Data Selection
    13.2 Optimised Encoding and Better PDEs

literature & reading

- J. Weickert: Anisotropic Diffusion in Image Processing. Teubner, Stuttgart, 1998.
- G. Aubert and P. Kornprobst: Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations. Second Edition, Springer, New York, 2006.
- T. F. Chan and J. Shen: Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods. SIAM, Philadelphia, 2005.
- F. Cao: Geometric Curve Evolutions and Image Processing. Lecture Notes in Mathematics, Vol. 1805, Springer, Berlin, 2003.
- R. Kimmel: The Numerical Geometry of Images. Springer, New York, 2004.
- G. Sapiro: Geometric Partial Differential Equations in Image Analysis. Cambridge University Press, 2001.
- Articles from journals and conferences.

Ethics for Nerds E4N

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**occasional / summer semester**	**1 semester**	**4**	**6**

responsible Prof. Dr.-Ing. Holger Hermanns

lecturers Prof. Dr.-Ing. Holger Hermanns
Kevin Baum
Sarah Sterz

entrance requirements We expect basic knowledge of propositional and first-order logic, an open mind, and interest to look at computer science in ways you probably are not used to.

assessments / exams The details of exam admission and grading are announced at the beginning of each iteration. Typically, participant are graded based on

- an exam or a re-exam (the better mark counts),
- a short essay where the participant has to argue for or against a moral claim in a topic from computer science.

To get the exam admission, participants usually have to get 50% of the points on weekly exercise sheets.

course types / weekly hours

```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

(may be adjusted before the start of each iteration of the course)

total workload

```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined based on exam performance, essay performance, and possibly exercise outcomes. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Many computer scientists will be confronted with morally difficult situations at some point in their career – be it in research, in business, or in industry. This module equips participants with the crucial assets enabling them to recognize such situations and to devise ways to arrive at a justified moral judgment regarding the question what one is permitted to do and what one should better not do. For that, participants will be made familiar with moral theories from philosophy, as well as different Codes of Ethics for computer scientists. Since one can quickly get lost when talking about ethics and morals, it is especially important to talk and argue clearly and precisely. In order to do prepare for that, the module offers substantial training regarding formal and informal argumentation skills enabling participants to argue beyond the level of everyday discussions at bars and parties. In the end, succesful participants are able to assess a morally controversial topic from computer science on their own and give a convincing argument for their respective assessments.

The module is intended to always be as clear, precise, and analytic as possible. What you won't find here is the meaningless bla-bla, needlessly poetic language, and vague and wordy profundity that some people tend to associate with philosophy.

content

This course covers:

- an introduction to the methods of philosophy, argumentation theory, and the basics of normative as well as applied ethics;
- relevant moral codices issued by professional associations like the ACM, the IEEE, and more;
- starting points to evaluate practices and technologies already in use or not that far away, including for instance: filter bubbles and echo chambers, ML-algorithms as predictive tools, GPS-tracking, CCTV and other tools from surveillance, fitness trackers, big data analysis, autonomous vehicles, lethal autonomous weapons systems and so on;
- an outlook on more futuristic topics like machine ethics, roboethics, and superintelligences;
- and more.

The content of the course is updated regularly to always be up-to-date and cover the currently most relevant topics, technologies, policies, and developments.

literature & reading

Will be announced before the start of the course on the course page.

Internet Transport

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr.-Ing. Thorsten Herfet

lecturers Prof. Dr.-Ing. Thorsten Herfet

entrance requirements
- Motivation for networks and communication
- Practical experience (e.g. through *Hands on Networking*) is recommended
- Knowledge of the fundamentals of communication (e.g. through *Digital Transmission & Signal Processing*) is recommended

assessments / exams
- Regular attendance of classes and tutorials
- Eligibility for exam through quizzes and assignments
- Final Exam
- A re-exam typically takes place during the last two weeks before the start of lectures in the following semester

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
  90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from performance in exams, quizzes and assigments. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Today the majority of all services is available via Internet-connections. Other than in the past this comprises not only data- but also media-services (like Voice Over IP or Video Streaming) and even Cyber-Physical Systems with their networked control loops.

The course introduces the basic characteristics of Internet-based communication (packetization on different layers, packet error detection and correction). It shows how existing protocols like HTTP, TCP and UDP can be shaped and evolved to fulfill the service requirements and how new protocols should be designed to serve the large variety of services.

content

- Introcudion of *EverythingoverIP* and *IPoverEverything*
- Theory of erasure channels (i.i.d, Gilbert-Elliott, channel capacity, minimum redundancy information)
- Wireless link layers (WiFi, PHY-bursts, Logical Link Control with DCF & EDCA, aggregation and ACK-techniques)
- Frame Check Sums, Cyclic Redundancy Checks
- Time Sensitive Networking
- Transport Layer services (flow control, congestion control, error control, segmentation and reassembly)
- QUIC media transport
- Error Coding under predictable reliability and latency (MDS-codes, binary codes)
- Upper layer protocols (HTTP, RTP/RTSP, DASH)

literature & reading

The course will come with a self-contained interactive manuscript. Complementary material will be announced before the start of the course on the course page on the Internet.

additional information

This module was formerly also known as *Future Media Internet* and *Multimedia Transport*.

Introduction to Image Acquisition Methods — IIAM

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**at least every two years**	**1 semester**	**2**	**4**

responsible Prof. Dr. Joachim Weickert

lecturers N.N.

entrance requirements Related core lecture *Computer Vision*

assessments / exams
- Written or oral exam at end of course
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours `2 h lectures (weekly)`

total workload
```
  30 h of classes
+ 90 h private study
= 120 h (= 4 ECTS)
```

grade Will be determined from performance in exams, exercises and practical tasks. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

The course is designed as a supplement for image processing lectures, to be attended before, after or parallel to them.

Participants shall understand

- what are digital images
- how they are acquired
- what they encode and what they mean
- which limitations are introduced by the image acquisition.

This knowledge will be helpful in selecting adequate methods for processing image data arising from different methods.

content

A broad variety of image acquisition methods is described, including imaging by virtually all sorts of electromagnetic waves, acoustic imaging, magnetic resonance imaging and more. While medical imaging methods play an important role, the overview is not limited to them.

Starting from physical foundations, description of each image acquisition method extends via aspects of technical realisation to mathematical modelling and representation of the data.

literature & reading

Will be announced before the start of the course on the course page on the Internet.

Realistic Image Synthesis RIS

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**at least every two years**	**1 semester**	**6**	**9**

responsible Prof. Dr. Philipp Slusallek

lecturers Prof. Dr. Philipp Slusallek
Dr. Karol Myszkowski
Guprit Singh

entrance requirements Related core lecture: *Computer Graphics*.

assessments / exams
- Theoretical and practical exercises (50% of the final grade)
- Final oral exam (other 50%)
- A minimum of 50% of needs to be achieved in each part to pass.
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  4 h lectures
+ 2 h tutorial
= 6 h (weekly)
```

total workload
```
   90 h of classes
+ 180 h private study
= 270 h (= 9 ECTS)
```

grade The final grade is be based on the assessments above. Any changes will be announced at the beginning of the semester.

language English

aims / competences to be developed

At the core of computer graphics is the requirement to render highly realistic and often even physically-accurate images of virtual 3D scenes. In this lecture students will learn about physically-based lighting simulation techniques to compute the distribution of light even in complex environment. The course also covers issues of perception of images, including also HDR technology, display technology, and related topics.

After this course students should be able to build their own highly realistic but also efficient rendering system.

content

- Rendering Equation
- Radiosity and Finite-Element Techniques
- Probability Theory
- Monte-Carlo Integration & Importance Sampling
- Variance Reduction & Advanced Sampling Techniques
- BRDFs and Inversion Methods
- Path Tracing & * Bidirectional Path Tracing
- Virtual Point-Light Techniques
- Density Estimation & Photon Mapping
- Vertex Connection & Merging
- Path Guiding
- Spatio-Temporal Sampling & Reconstruction
- Approaches for Interactive Global Illumination
- Machine Learning Techniques in Rendering

- Human Perception
- HDR & Tone-Mapping
- Modern Display Technology
- Perception-Based Rendering

literature & reading

Litrature will be announced in the first lecture of the semester.

But here are some relevant text books:

- Pharr, Jakob, Humphreys, Physically Based Rendering : From Theory to Implementation, Morgan Kaufmann
- Shirley et al., Realistic Ray Tracing, 2. Ed., AK. Peters, 2003
- Jensen, Realistic Image Synthesis Using Photon Mapping, AK. Peters, 2001
- Dutre, at al., Advanced Global Illumition, AK. Peters, 2003
- Cohen, Wallace, Radiosity and Realistic Image Synthesis, Academic Press, 1993
- Apodaca, Gritz, Advanced Renderman: Creating CGI for the Motion Pictures, Morgan Kaufmann, 1999
- Ebert, Musgrave, et al., Texturing and Modeling, 3. Ed., Morgan Kaufmann, 2003
- Reinhard, Ward, Pattanaik, Debevec, Heidrich, Myszkowski, High Dynamic Range Imaging, Morgan Kaufmann Publishers, 2nd edition, 2010.
- Myszkowski, Mantiuk, Krawczyk. High Dynamic Range Video. Synthesis Digital Library of Engineering and Computer Science. Morgan & Claypool Publishers, San Rafael, USA, 2008.
- Glassner, Principles of Digital Image Synthesis, 2 volumes, Morgan Kaufman, 1995

Trusted AI Planning — TAIP

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
5-6	**6**	**at least every two years**	**1 semester**	**4**	**6**

responsible Prof. Dr. Jörg Hoffmann

lecturers Prof. Dr. Jörg Hoffmann

entrance requirements *Programming 1*, *Programming 2*, *Fundamentals of Data Structures and Algorithms*, and *Elements of Machine Learning* or other courses in machine learning are recommended. The *Artificial Intelligence* core course provides useful background but is not necessary.

assessments / exams
- Regular attendance of classes and tutorials
- Solving of weekly assignments
- Passing the final written exam
- A re-exam takes place during the last two weeks before the start of lectures in the following semester.

course types / weekly hours
```
  2 h lectures
+ 2 h tutorial
= 4 h (weekly)
```

total workload
```
   60 h of classes
+ 120 h private study
= 180 h (= 6 ECTS)
```

grade Will be determined from the performance in exams. The exact modalities will be announced at the beginning of the module.

language English

aims / competences to be developed

Knowledge about methods for learning, verifying and testing action policies in AI Planning; understanding of algorithmic techniques enabling these methods.

content

- Introduction to basic AI concepts needed in the course
- Partial-order reduction
- Dominance pruning
- SAT-based planning
- ASNet action policies
- Safety verification of neural action policies, basic methods
- Safety verification of neural action policies: policy predicate abstraction
- Testing methods for learned action policies, deterministic and probabilistic settings

literature & reading

There is no text book covering the course topics. Links to relevant publications and other material where available will be provided on the slides

additional information

This module was formerly also known as *AI Planning*.

Module Category 8

Bachelor's Seminar and Thesis

Bachelor's Seminar

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
6	6	every semester	1 semester	2	9

responsible Dean of Studies of the Faculty of Mathematics and Computer Science
Dean of Studies of the Department of Computer Science

lecturers Lecturers of the department

entrance requirements Minimum acquisition of 120 CP.

assessments / exams
- Written formulation of the task of the bachelor's thesis and the relevant scientific literature.
- Presentation of the planned assignment with subsequent discussion
- Active participation in the discussion

course types / weekly hours 2 h seminar

total workload
```
  30 h of classes (seminar)
+ 30 h mentoring by the chair
+ 210 h private study
= 270 h (= 9 ECTS)
```

grade Will be determined from the performance in the lecture and the written report. The exact modalities will be announced by the respective instructor.

language English or German

aims / competences to be developed

In the Bachelor's seminar, the student acquires the ability to work scientifically in the context of an appropriate subject area under supervision.

At the end of the Bachelor's seminar, the foundations for the successful completion of the Bachelor's thesis are laid and essential approaches to solving the problem are already determined.

The Bachelor's seminar thus prepares the topic and execution of the Bachelor's thesis.

It also teaches practical skills of scientific discourse. These skills are taught through active participation in a reading circle, in which the discussion of scientifically challenging topics is practised.

content

Familiarisation with a scientific subject area within the field of computer science.

Preparation of a written elaboration of the task of the Bachelor thesis and the relevant scientific literature.

Presentation of the subject area and the planned task of the Bachelor's thesis.

The topic is defined in close consultation with the supervising lecturer.

literature & reading

Scientific articles appropriate to the subject area in close consultation with the supervising lecturer

Bachelor's Thesis

st. semester	std. st. sem.	cycle	duration	SWS	ECTS
6	**6**	**every semester**	**3 months**	**-**	**12**

responsible Dean of Studies of the Faculty of Mathematics and Computer Science
Dean of Studies of the Department of Computer Science

lecturers Lecturers of the department

entrance requirements Successful completion of the *Bachelor's Seminar*.

assessments / exams Written elaboration. It describes both the result of the work and the path that led to the result. The student's own contribution to the results must be clearly recognisable. In addition, presentation of the Bachelor's thesis in a colloquium, in which the independence of the student's performance is also examined.

course types / weekly hours none

total workload
```
  30 h supervision by the chair
+ 330 h private study
= 360 h (= 12 ECTS)
```

grade Assessment of the Bachelor's thesis by the reviewers.

language English or German

aims / competences to be developed

The Bachelor's thesis is a project work that is carried out under supervision. It is intended to enable the candidate to independently solve a problem from the field of computer science within a given period of time and to document the results in a scientifically appropriate form.

content

Work on a current problem from the field of computer science under supervision. Adequate documentation of the results in the form of a scientific thesis.

The topic is defined in close consultation with the instructing lecturer.

literature & reading

Scientific articles appropriate to the subject area in close consultation with the instructing lecturer.