Course Module Catalogue: Computer Science for Digital Media (M.Sc.)

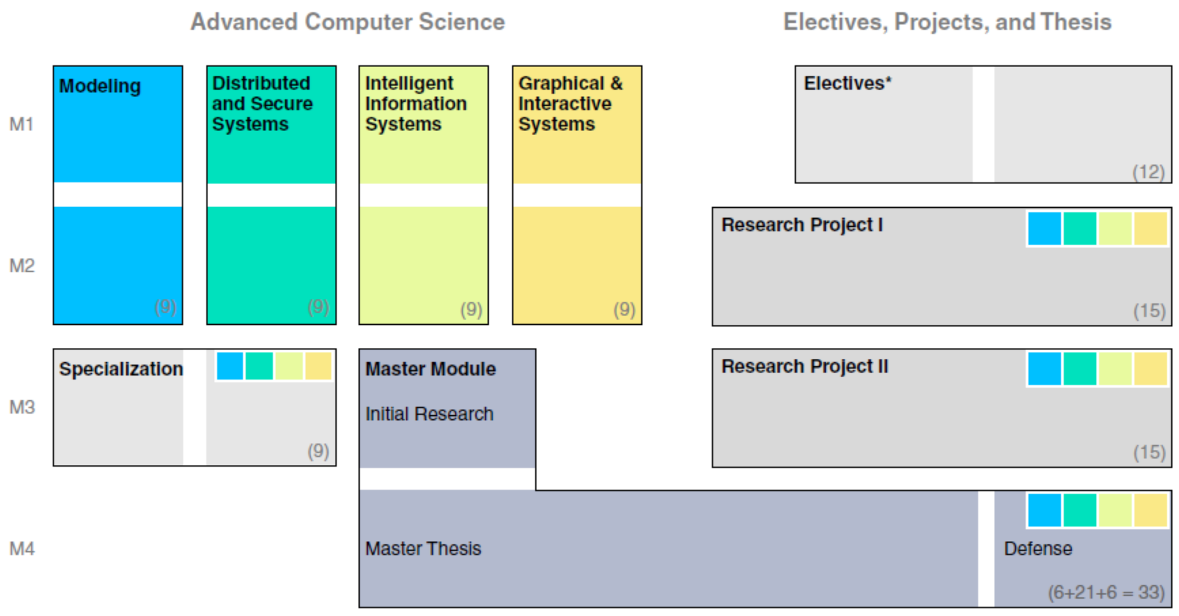
The Master's degree course in Computer Science for Digital Media lasts 4 semesters and comprises 120 credit points. Its aim is to prepare students for careers as computer scientists, with its main focus being on digital media. The course module plan, which follows the Society of Computer Science's recommendation of so-called *Type 2 degree programmes*, is divided into four subject modules, one specialist module, one elective module, two research projects and one module for the final paper.

The **four subject modules** deal with the following aspects of computer science: mathematical modeling; distributed systems and IT security; intelligent information systems; graphical and interactive systems. The fifth subject module gives the student the opportunity to **specialise** with a more precise focus on one or two of the aforementioned aspects of computer science.

As part of the **elective module**, students are free to attend courses from other departments of the Faculty of Media, other university faculties or language courses, thus acquiring additional knowledge and skills.

The **research projects** not only aim to expand relevant specialist skills, but can also in some cases cover interdisciplinary projects. Beyond that, they serve as a means of developing further key compteneces such as teamwork, project management and presentational skills.

Preparation for the **final paper** begins as early as the third semester with an initial research phase. This is followed by a period of four months in which students must produce the paper itself. The final stage of the course module (and of the entire course of studies) is the defence of the Master's paper.



Module Title	Modelling	Module number

Semester (optional)	Frequency	Regularity and duration	ECTS credit points	Workload [hours]	Language	Module coordinator
	Every semester	During the semester, on a weekly basis	9	67.5 in-class, 160.00 self-study, 42.5 exam preparation (incl. exam). Total: 270.	English	Andreas Jakoby

Type and application of module	Formal requirements for participation	Examination requirements
M.Sc. Computer Science for Digital Media	Admission to M.Sc. programme "Computer Science for Digital Media". See course descriptions for further requirements, if any.	The overall grade for the module is calculated as the weighted mean of the grades obtained in the component courses. See course descriptions for the examination requirements specific to the component courses.

Target qualifications

A model is concept which is used to understand a subject of a system. It is formed after a process of conceptualization and generalization. The goal of the module is to develop an understanding of specific principles in algorithm design and mathematical models. Students should

- understand the specific challenges posed be mathematical models and algorithm design principles
- master techniques required to analyse or develop algorithms and mathematical models
- learn about the application of these techniques to specific problems and tasks
- learn to recognise the advantages of alternative approaches for solving these problems and tasks
- make well-informed decisions about an approach in order to solve problems
- recognise the state of research in a specific sub-field of modelling

More specifically, students should acquire in-depth knowledge of specific fields taught in the wider field of modelling. The specific fields are taught in component courses (see below). It is not permissible for students already to have studied these fields in depth in a previous Bachelor's programme. After completing the component courses, students should be able to undertake original research, or at least independent academic work at the Master's thesis level in these specific fields. For each component course, there is a more detailed list of target qualifications.

Contents

See course descriptions.

Didactic concept

Unless otherwise specified in the description of a component course: lectures and practical sessions combined with individual and group-based studies related to theoretical and practical aspects of the course contents. Practical sessions can include project-oriented and laboratory work based on concrete problems. Theoretical aspects can include reading, understanding and presenting recent publications. Classes consist of one 90-minute lecture and one 45-minute practical session per week during the semester. Postdoctoral researchers, doctoral students and teaching assistants supervise students and are available for intensive discussion and feedback.

Special information

See course descriptions

Lectures / courses included in the module (optional)	SWS / ECTS credit points (optional)
The module consists of two of the following courses to be chosen by the students:	
• Advanced Analysis	• 4.5
• Advanced Numerical Mathematics	• 4.5
• Advanced Type Theory for Functional Programming	• 4.5
• Discrete Optimization	• 4.5
• Geometry	• 4.5
• Introduction to Functional Programming with Haskell	• 4.5
• Logics and Semantic Web	• 4.5
• Online Computation	• 4.5
• Randomized Algorithms	• 4.5
• On special application to the examination committee: Watermarking & Steganography	• 4,5

Course Title	Online Computation
Coordinator	Andreas Jakoby
Assigned Module(s)	Modeling, Specialist Module
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Final oral exam (max. 45 min.).
Specific target qualifications	Online computation is a model for algorithms and problems which require decision under uncertainty. In an online problem, the algorithm does not know the entire input from the beginning; the input is revealed in a sequence of steps. An online algorithm should make its computation based only on the observed past and without any secure knowledge about the forthcoming sequence in the future. The effects of a decision taken cannot be undone. The goal of this course is understand the principles of designing and analysing schemes for online computations and for competing online problems. The course deals with the following topics: • basic concepts for analysing the competitive ratio (including potential function, factoring and partitioning techniques) • various concrete online algorithms (including MTF, BIT, RMTF for the list-accessing problem and LRU, CLOCK, LFU, LFD, RAND, MARK for the paging problem) • locality of request sequences (e.g. use of the access graph model or Markov chains) • extensive versus strategic forms of games • strategy forms for games and their connections to online problems • equivalence theorems for linear games • request-answer systems with respect to different types of adversaries • competitive analysis and zero-sum games • Yao's principle for obtaining lower bounds Students should understand the application of competitive analysis for solving concrete problems. They should be able to distinguish efficient from inefficient solutions. Students should master concepts and approaches such as • analysing the competitive ration of concrete online algorithms using the potential method • analysing the competitive ration of concrete online algorithms using factoring and phase partitioning • analysing the competitive ration of concrete online algorithms using game theory • knowing how to analyse different online problems with respect to oblivious and adaptive adversaries • knowing how to analyse online problems using the minimax theorem in order to tackle problems from online computation and competitive analysis. They should be able to understand proposed competitive analysis problems, to compare different proposals for online computations, to make well-informed decisions about the preferred proposal and, if necessary, to find their own solutions to given online problems. Students should develop an understanding of the current state of research in online computation and competitive analysis. With appropriate supervision, students should be able to tackle research problems within this field.
Contents	• Some Basic Concepts: optimization problems, competitive ratio, games and adversaries • The Potential Function Method: amortized costs, interleaving moves • The List-Accessing Problem • The Sleator-Tarjan Result for MTF • Some Lower Bounds for the List Accessing Problem • The List-Factoring Technique • The Phase-Partitioning Technique • competitive ratio of randomised algorithms • Randomised Algorithms and the List-Accessing Problem: BIT and RMTF • Phase Partitioning and the BIT-Algorithm • Algorithm COMB • ThePpaging Problem • Some Deterministic Paging Algorithms • The Full Access Cost Model • Adversary Models • Some Randomised Paging Algorithms: Rand and Mark

	• Locality and the Paging Problem • LRU in the Access Graph Model • The FAR Algorithm • Distributional Paging and the Markov Chain Model • Game-Theoretic Foundations • Extensive and Strategic Form of Games • Randomised Strategies for Games • Equivalence Theorems for Linear Games • Memoryless Behavioral Paging Algorithm • Memoryless Mixed Paging Algorithm • Request-Answer System - adversaries and their interaction with algorithms • Request-Answer System - the competitive ratio • Competitive Analysis and Zero-Sum Games • Generalizing the Minimax Theorem • Yao’s Principle: obtaining lower bounds
Special information	Allan Borodin, Ran El-Yaniv, Online Computation and Competitive Analysis, CAMBRIDGE UNIVERSITY PRESS, 2005

Course Title	Randomised Algorithms
Coordinator	Andreas Jakoby
Assigned Module(s)	Modeling, Specialist Module
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Final oral exam (max. 45 min.).
Specific target qualifications	For many problems, randomised algorithms are the only known efficient solution method. For some other problems we can find randomised algorithms that are much simpler and more understandable than any known deterministic method. The goal of this course is to understand the principles of designing and analysing randomised algorithms. The course deals with the following topics: • basic concepts and inequalities from probability (including Chernoff bounds, Markov's, Chebyshev's, and Jensen's inequality) • various randomised algorithms (e.g. for verifying matrix multiplication, sorting, computing the median, cut problems, packet routing, hashing, satisfiability, Hamiltonian cycles, independent sets, K_4-subgrapaph problem, etc.) • average-case analysis of algorithms • balls and bins • random graphs • basic probabilistic methods • Lovasz Local Lemma • derandomisation • Markov chains Students should be able to apply the above tools, algorithm, and concepts to solve concrete problems. Furthermore, they should appreciate the limits and constraints of the above topics. Students should be able formalise and generalise their own solutions using randomization. Students should master concepts and approaches such as • using the principle of deferred decisions to analyse randomised algorithms • decreasing error probability by running randomised algorithms multiple times • distinguishing between Las Vegas and Monte Carlo algorithms • using the moment-generating function and Chertoff bounds for analysing the tail probabilities • using the balls and bins model to solve concrete problems and analyse randomised algorithms • using randomised graphs to analyse the average complexity of hard problems • using basic probabilistic methods to solve problems • knowing how to apply Lovasz Local Lemma to analyse hard problems • knowing techniques for derandomisation based on probabilistic methods Students should understand the current state of research in randomised algorithms, specifically of the design, analysis and application of randomised algorithms. With appropriate supervision, students should be able to tackle research problems in randomised algorithms.
Contents	• Some Notes on Probability • Verifying Matrix Multiplication • A Randomised Min-cut Algorithm: Karger's min-cut algorithm • A Randomised Version of Quicksort • Coupon Collector's Problems • A Randomised Algorithm for Computing the Median • Types of RandomisedAalgorithms: Las Vegas versus Monte Carlo • Randomised Computational Complexity Ttheory • Moment-Generating Functions and Chernoff Bounds (versions for independent Poisson trials) • Estimating Probability from Samples • Packet Routing in Sparse Networks (including bit-fixing routing mechanism hypercubes) • Balls and Bins • Applications for Balls and Bins: bucket sort, hashing with bit strings, Bloom filters, breaking symmetry • Models for Random Graphs • Hamiltonian Cycles in Random Graphs • Basic Probabilistic Methods

	• The Counting Argument and Edge Coloring • The Expectation Argument • Applications for Probabilistic Methods: maximum satisfiability, finding a large cut • Derandomisation Using Conditional Expectations and Finding a Large Cut • Applications for Sample and Modify: independent sets, graphs with large girth • The Second Moment Method and Applications for Threshold Behaviour in Random Graphs • Conditional Expectation Inequality and Applications for K_4-Subgraph Problem • Lovasz Local Lemma • Lovasz Local Lemma and Application to Edge-Disjoint Paths • Lovasz Local Lemma and Application to Satisfiability • Explicit Constructions Using the Local Lemma • Explicit Constructions for Satisfiability • Lovasz Local Lemma: the General Case • Markov Chains • A Randomised Algorithm for 2-Satisfiability
Special information	Michael Mitzenmacher, Eli Upfal, Probability and Computing - Randomised Algorithms and Probabilistic Analysis, CAMBRIDGE UNIVERSITY PRESS, 2005

Course Title	Advanced Analysis
Coordinator	Prof. Dr. rer.nat. habil. Klaus Gürlebeck
Assigned Module(s)	Modelling
Formal requirements for participation	Analysis (course)
Examination requirements	Written examination
Specific target qualifications	Many real-world problems lead to mathematical models in the form of partial differential equations. These models can be transformed into numerical models and used for physically correct simulations, optimisations or parameter identifications. Students will be provided with the necessary tools to model and solve linear problems. The course will deal with and understand the following topics: • basics of ordinary differential equations • classification of partial differential equations • partial differential equations and coordinate transforms • solution of partial differential equation in unbounded domains, initial value problems • solutions to boundary value problems in bounded domains by series expansions • error estimates • integral representation formulas • concrete models and their simulations. Students should be able to apply the above tools and the theory to solve concrete problems. Furthermore, they should be able to create computer simulations with computer algebra systems. Students should be able to understand • the idea of mathematical modelling • the mathematical assumptions and the resulting restrictions • how to evaluate and check the correctness of a model or of a solution in order to solve problems from mathematical physics, mechanics and image processing and create accurate simulations. They should be able to identify a suitable mathematical model and to adapt it to the given situation if necessary. Students should understand special problems at research level and be able to work with them in form of supervised projects.
Contents	• Classification of Partial Differential Equations • Coordinate Transforms, Canonical Forms • Analytical Solution Methods • Modelling of Real-World Problems
Special information	Burg/Haf/Wille: Höh. Math. f. Ing., Bde. 3-5, Taylor: Partial Differential Equations I-III. Maple

Course Title	Advanced Numerical Mathematics
Coordinator	Prof. Dr. rer.nat. habil. Klaus Gürlebeck
Assigned Module(s)	Modeling
Formal requirements for participation	Courses Analysis, Linear Algebra and Numerical Mathematics
Examination requirements	Oral examination, 30 minutes with 30 minutes for preparation
Specific target qualifications	The lecture course introduces concepts, algorithms, and theoretical background for the numerical solution of partial differential equations. The accompanying practical classes are concerned with theoretical as well as applied tasks in order to expand understanding of the field. This will be completed by classes in the computer lab. The computer simulations are based on Matlab programs. The course will deal with the following topics: • numerical linear algebra • the iterative solution of linear and non-linear systems of algebraic equations • discretization and numerical solution of ordinary and partial differential equations • finite difference methods • approximation, stability and convergence • error estimates • concrete models and their simulations, based on Matlab Students should be able to apply the above tools and the theory to solve concrete problems. Furthermore, they should be able to create numerical computer simulations with Matlab. Students should be able to understand • the idea of mathematical modelling and discretization • the mathematical assumptions and the resulting restrictions • how to evaluate the quality of a numerical model • how to improve the efficiency of a numerical method in order to solve practical problems from mathematical physics and engineering in order to create accurate simulations. They should be able to adapt standard models to the given situation if necessary. Students should be able to understand special problems at research level and be able to work with them in the form of supervised projects.
Contents	• Numerical Linear Algebra • Discretization of Ordinary and Partial Differential Equations • Finite Difference Methods, Approximation, Stability and Convergence • Numerical Simulations
Special information	Varga, Matrix iterative analysis. Hermann, Numerische Mathematik Kress, Numerical Analysis Matlab

Course Title	Advanced Type Theory for Functional Programming
Coordinator	Dr. rer. nat. Dmitrii Legatiuk
Assigned Module(s)	Modeling
Formal requirements for participation	No specific requirements for this course
Examination requirements	Submission of a project given during semester with weight of 50% of total grade. The project has to be presented at the final oral examination (max. 30 min) with a time for preparation (max. 30 min).
Specific target qualifications	Type theory is a solid part of modern programming languages. Development of new concepts and understanding principles of programming languages require a careful consideration of types and their role in programming. Functional programming is a paradigm in which type theory is naturally included via formalism of typed versions of lambda calculus. Another modern formalism closely related to functional programming and type theory is category theory, which is, in simple terms, the abstract theory of functions. Haskell is an example of a modern programming language in which both concepts come naturally together. The goal of this course is to present advanced topics in functional programming related to type and category theories. Students should understand the following topics: • basics of category theory • Cartesian closed categories • the relationship between typed lambda calculus and category theory • polymorphism • type inference and type checking • type operators and higher-order polymorphism • functors and monads in Haskell. Students should be able to apply the above tools and theories to solve concrete problems. Furthermore, they should appreciate the limits and constraints of the above theories. Students should be able formalise and generalise their own solutions using the above topics. Students should master concepts and approaches such as • thinking in an abstract manner • understanding what lies behind mathematical formalism • learning advanced functional programming in order to tackle problems from programming and its application to digital media. They should be able to understand proposed programming problems, to make well-informed decisions about the preferred proposal and, if necessary, to find their own solutions to given programming problems. Students should develop an understanding of the current state of research in type theory and functional programming. With appropriate supervision, students should be able to tackle research problems.
Contents	• Category Theory • Typed Versions of Lambda Calculus • Polymorphism • Type Checking
Special information	B. Pierce, Basic category theory for computer scientists B. Pierce, Types and programming languages Haskell Platform

Course Title	Discrete Optimisation
Coordinator	Andreas Jakoby
Assigned Module(s)	Modeling, Specialist Module
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Final oral exam (max. 45 min.).
Specific target qualifications	Discrete optimisation is about finding optimal solutions for discrete problems. Finding efficient algorithms for discrete optimisation problems is one of the main topics in algorithm design. The goal of this course is to understand the principles of analysing discrete optimisation problems and designing efficient algorithms for such problems. Students should understand the following topics and methods: • discrete optimisation problems and complexity theory • heuristic and local search strategies for optimisation problems • backtracking for discrete optimisation problems • branch-and-bound schema • convex optimisation problems and linear programming • Simplex-Algorithm and Ellipsoid-Algorithm • greedy algorithms • approximability of concrete problems Students should be able to apply the above concepts to solve concrete problems. Furthermore, they should appreciate the limits and constraints of the above schemes. Students should be able formalise and generalise their own solutions using the above tools. Students should master concepts and approaches such as • analyzing the intractability of discrete optimisation problems • using different algorithmic concepts to design efficient algorithms for discrete optimisation problems • using transformations to solve optimisation problems • knowing the limits of efficient optimisation algorithms • using approximations to find adequate solutions in order to tackle optimisation problems. They should understand the current state of research in discrete optimisation, specifically of the design, analysis and application of schemes for optimisation problems. With appropriate supervision, students should be able to tackle research problems in discrete optimisation.
Contents	• Introduction • Heuristic Search - a general algorithm for search problems • Heuristic Search - best first search • Best First Search with Duplicate Elimination • The A^* -algorithm • Backtracking for Discrete Optimisation Problems • Knapsack Problem, Traveling Salesperson Problem, MAXCLIQUE Problem • General Backtracking Strategy • Backtracking with Bounding Function • Branch-and-Bound Schema • Alpha-Beta-Search for 2-Party-Games • Local Search • Hopfield neural networks • Maximum-Cut Approximation via Local Search • Application to Vertex Cover • Local Search for Discrete Optimisation Problems • The Metropolis Algorithm and Simulated Annealing • Linear Programming • Complexity of Linear Programs • Geometry of Linear Programs • Basic Feasible Solution

	• The Simplex-Algorithm • The Ellipsoid-Algorithm • Affine Transformations and Ellipsoids • Precision of Computation • Greedy Algorithms and Bounds on the Optimum: a load balancing problem • Greedy Algorithms and Knowing the Optimum: the center selection problem • A First Step to the Pricing Method: the Set Cover Problem • Approximations via Reductions: the Vertex Cover Problem • The Pricing Method for the Vertex Cover Problem • Arbitrary Good Approximations: the Knapsack Problem • An Introduction to Linear Programming and Rounding: Vertex Cover revisited • Linear Programming and Rounding: generalized load balancing
Special information	T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, MIT Press, 1990 J. Kleinberg, E. Tardos, Algorithm Design, Addison Wesley, 2005 W. Kocay, D. Kreher, Graphs, Algorithms and Optimisation, CRC 2005 D. Kreher, D. Stinson, Combinatorial Algorithms, CRC Press, 1999 C. H. Papadimitriou, K. Steiglitz, Combinatorial Optimisation: Algorithms and Complexity, Dover Books on Computer Science, 2000

Course Title	Geometry
Coordinator	Reinhard Illge
Assigned Module(s)	Modeling
Formal requirements for participation	(No specific requirements for this course)
Examination requirements	Active participation in problem session: solving at least two problems identified in the session and presenting the solutions on the blackboard. Final oral exam (max. 45 min.).
Specific target qualifications	One of the defining features of mathematics, already formulated in ancient Greece, is to achieve truth by proof, to find laws by logical conclusions. This approach was explained in Euclid`s Elements and completed by David Hilbert in his book "Grundlagen der Geometrie". The goal of this course is to present synthetic geometry as a prime example for the systematic development of a theory, based on a few axioms. Students should understand the following topics: • the set-up of a geometry based on logical conclusions requires some basic truth called axioms • proof of propositions by applying (only) the already known principles • the concept of coordinates without the use of length and angular measurement • the generation of the complete set of congruence maps in the plane by a single type of maps (line reflections) • possibilities for classifying motions • generalising the congruence maps into similarity maps by adding central similarities • the study of different types of symmetries and their relation to certain finite groups • identifying the special cases where a commutative law applies • the study of some properties of figures in the plane and the space • presentation of the idea of the Golden Ratio, which is widespread in nature, art, and architecture • study of the Archimedes procedure to calculate the circumference as the probably oldest numerical algorithm Students should be able to apply the above tools and topics for solving concrete problems. Furthermore, they should appreciate the limits and constraints of the above, e.g that the change of the parallel axiom leads to another, Non-Euclidean geometry. Students should be able to formalise and generalise their own solutions using the above tools. Students should master concepts and approaches such as • a strict use of the style of thinking known as deduction • solving some practical tasks using geometrical methods (e.g. certain minimal problems) • solving some construction tasks using only ruler and compass • the strong set-up of a theory by logical structures • navigation using modern means such as GPS, based on long-standing geometric ideas in order to tackle problems from geometry and its application to digital media. They should be able to understand proposed geometrical problems, to make well-informed decisions about the preferred proposal and, if necessary, to find their own solutions to given geometrical problems. Students should develop an understanding of the current state of research in Geometry. With appropriate supervision, students should be able to tackle research problems.
Contents	• Axiomatic Approach to Euclid´s Geometry • Congruence Maps (motions) • Similarity Maps • Plane Figures • Spatial Figures
Special information	Walter Meyer: Geometry and Its Applications, Elsevier 2011

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

Course Title	Logics and Semantic Web
Coordinator	Benno Stein
Assigned Module(s)	Modeling, Specialist Module
Formal requirements for participation	(no specific requirement for this course)
Examination requirements	Active participation in lab classes. Final written exam.
Specific target qualifications	The first part of this lecture course (two-thirds) introduces the notions and methods of formal logic, covering propositional logic, predicate logic and the foundations of automated deduction. Based on this, the second part of the lecture explains the inference concepts behind the semantic web. Students should understand the following concepts from logics: • Propositional and predicate logics: ◦ inductive formation of formulae ◦ satisfiability ◦ logical entailment ◦ equivalence of syntax and semantics ◦ correct and sound calculi • Semantic Web ◦ RDF syntax ◦ modeling and inference Students should be able to employ logics as a modeling tool. They should understand and be able to explain the concept of entailment and how to automate theorem proofing. Based on these insights, they should be able to explain the working principles of the semantic web, to model knowledge-based relations and to encode them using an OWL variant. Students should master • semantic approaches to logical entailment • syntactic approaches to logical entailment • RDF as a language for logics in the web • an OWL variant (light, DL, full) in order to model semantic relations in web-based applications. Students should develop an understanding of the current developments of the semantic web, as well as its possibilities and its limits and constraints. With appropriate supervision, they should be able to tackle research problems.
Contents	• Propositional Logic: ◦ syntax ◦ semantics ◦ formula transformation ◦ satisfiability algorithms • Predicate Logic: ◦ syntax ◦ semantics ◦ formula transformation ◦ satisfiability algorithms ◦ decidability • Semantic Web ◦ RDF ◦ RDF schema ◦ Ontologies
Special information	Course material: http://www.uni-weimar.de/en/media/chairs/webis/teaching/lecturenotes/#logics Literature: • Cori/Lascar. *Mathematical Logic* • Fensel. *Spinning the Semantic Web* • D. W. Loveland. *Automated Theorem Proving: A Logical Basis* • Powers. *Practical RDF*

	• M.R.A. Ruth and M.D. Ryan. *Logic in Computer Science - Modelling and Reasoning about Systems* • U. Schöning. *Logic for Computer Scientists*

Module Title	Distributed and Secure Systems	Module number

Semester (optional)	Frequency	Regularity and duration	ECTS credit points	Workload [hours]	Language	Module coordinator
	Every semester..	During the semester, on a weekly basis	9	67.5 in-class, 160.00 self-study, 42.5 exam preparation (incl. exam). Total: 270.	English	Stefan Lucks

Type and application of module	Formal requirements for participation	Examination requirements
M.Sc. Computer Science for Digital Media	Admission to M.Sc programme "Computer Science for Digital Media". See course descriptions for further requirements, if any.	The overall grade for the module is calculated as the weighted mean of the grades obtained in the component courses. See course descriptions for the examination requirements specific to the component courses.

Target qualifications

A distributed system is a model in which components located on a network need to communicate and coordinate their actions. Security means defending a system against malicious adversaries. The goal of the module is to develop an understanding of specific challenges and approaches in orderto learn about concurrency and malice in systems. Students should

- learn about the specific challenges posed by distribution or malice
- master techniques required to analyse or develop distributed or secure systems
- learn about the application of these techniques to specific problems and tasks
- learn to recognise the advantages of alternative approaches for solving these problems and tasks
- make well-informed decisions about approaches in order to solve problems
- recognise the state of research in a specific sub-field of the distributed and secure systems.

More specifically, students should acquire in-depth knowledge of specific fields taught in the wider field of distributed and secure systems. The specific fields are taught in component courses (see below). It is not permissible for students already to have studied these fields in depth in a previous Bachelor's programme. After completing the component courses, students should be able to undertake original research, or at least independent academic work, at Master's thesis level in these specific fields. For each component course, there is a more detailed list of target qualifications.

Contents

See course descriptions.

Didactic concept

Unless otherwise specified in the description of a component course: lectures and practical sessions combined with individual and group-based studies related to theoretical and practical aspects of the course contents. Practical sessions can include project-oriented and laboratory work based on concrete problems. Theoretical aspects can include reading, understanding and presenting recent publications. Classes consist of one 90-minute lecture and one 45-minute practical session per week during the semester. Post-doctoral researchers, doctoral students and teaching assistants supervise students and are available for intensive discussion and feedback.

Special information

See course descriptions

Lectures / courses included in the module (optional)	SWS / ECTS credit points (optional)
The module consists of two of the following courses to be chosen by the students:	
• Safety and Security Engineering	• 4.5
• Advanced Cryptography: Cryptographic Hash functions[1]	• 4.5
• Advanced Cryptography: Secure Channels[1]	• 4.5
• Digital Watermarking	• 4.5
• Quantum Algorithms & Cryptanalysis	• 4.5
• Introduction to Modern Cryptography[2]	• 4.5

[1] The lecture requires a previous introduction to cryptography, such as „Introduction to Modern Kryptography".
[2] The „Introduction to Modern Cryptography" can only be taken by students without a previous introduction to cryptography.

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

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

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

Course Title	Introduction to Modern Cryptography
Coordinator	Stefan Lucks
Assigned Module(s)	Distributed and Secure Systems
Formal requirements for participation	(none)
Examination requirements	Active participation at problem session (minimum 25% of achievable points per problem set), and solving one indiividual assignment. Final oral exam (max. 45 min.).
Specific target qualifications	Cryptography is about communication in the presence of adversaries. The lecture introduces students to the design and analysis of cryptographic systems. Because one needs to understand how systems fail, before one can design and implement better systems, there is also a focus on cryptographic attacks. Students should be able to give examples for • classical cryptosystems (Caesar cipher, substitution cipher, …), • stream ciphers (One-Time Pad, …), • abstract block ciphers and their formal analysis, • practical block ciphers (DES, AES) and differential cryptanalysis, • block cipher modes of operation (ECB, CBC, …) and their strengths and weaknesses, • message authentication codes and their strength and weaknesses, • public-key cryptosystems (RSA, Rabin, Diffie-Hellman), • attacks on complex cryptosystems, and • provably secure cryptosystems and proofs. Students should master basic ideas of the art and science of cryptography, such as • how to formally model security requirements • how to design stream and block ciphers • how to use stream or block ciphers for secure authentication and encryption • how to design public-key cryptosystems • how to use public-key cryptosystems for key exchange and digital signatures Specifically, for a given cryptosystems and a given application • students should decude if the cryptosystem is secure for that application or not, • if secure, students should be able to argue why it is secure, • and, if insecure, students should be able to argue why it is insecure (typically by presenting an attack on the cryptosystem)
Contents	1. Introduction 2. Passwords 3. Stream Ciphers 4. Block Ciphers 5. Security Challenges & Attacks 6. Asymmetric Cryptosystems 7. Insecure Cryptosystems from Secure Bulding Blocks 8. Provable Security
Special information	Students need a decent knowledge about Mathematics, especially Discrete Mathematics, for this course. Students who did already take an introduction to cryptography must not take part at this lecture, and, specfically, are excluded from the exams. To verify this condition, students must present copies of their "transcripts of records" from previous studies, when applying for the exam. On the other hand, students who did not take an introduction to cryptography, previously, must first take this course and pass the exam before they are allowed to take part at lectures on Advanced Cryptography, such as Secure Channels and Cryptographic Hash functions.

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

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

Course Title	Quantum Algorithms & Cryptanalysis
Coordinator	Stefan Lucks
Assigned Module(s)	Distributed and Secure Systems, Specialization. On request to the examination committee: Modelling.
Formal requirements for participation	(none)
Examination requirements	Active participation at the problem session (minimum 25% of achievable points per problem set). A final oral exam (at most 45 min.).
Specific target qualifications	The computational model of a quantum computer is fundamentally different from the classical model of computation. Quantum computers can solve certain problems efficiently, which, to the best of our knowledge, are infeasible on a classical computer. E.g., Shor's celebrated period-finding algorithm, can be used to factorise huge numbers and compute huge discrete logarithms, thus breaking almost all currently used asymmetric cryptosystems. Such exploits assume ECLSQ (Error-Correcting Large-Scale Quantum) computers, which will not be available for many years (if ever). Nevertheless, with the current advent of the first NISQ ("Noisy Intermediate-Scale Quantum") computers, it becomes increasingly important for computer scientists – and especially for cryptographers – to understand how quantum computers work, what quantum computers can do, and what they can't do. The students will master the following topics: • The fundamental difference between a classical state and a quantum state. • Operations over quantum states, modelled as linear operations over the complex numbers. • Basic quantum algorithms, such as ◦ amplitude Amplification and Grover's algorithm to "find a needle in a haystack", ◦ the quantum Fourier transform and Shor's factorization method. • The application of basic quantum algorithms to quantum cryptanalysis: ◦ symmetric key-recovery in time 2**(n/2) using Grover's algorithm, ◦ hash collisions using Grover's algorithm in time 2**(n/3), ◦ factorization and discrete logarithm using Shor's method. • The polynomial method in quantum complexity theory. • Post-quantum asymmetric cryptography. • Quantum error correction. The students will understand • The quantum model of computation. • Its application to design and analyse quantum algorithms. • The application of such algorithms to cryptanalysis. • Some of the limits of quantum computers. The students will conceive knowledge about the state of research in quantum algorithms, with a focus on the application to quantum cryptanalysis. Given some guidance, they will be able to tackle current research problems in quantum cryptanalysis.
Content	1. classical bits, quantum bits, classical states, and quantum states 2. quantum gates and quantum circuits 3. quantum key exchange 4. Deutsch's problem and Simon's problem 5. Grover's amplitude amplification: how to find a needle in a haystack 6. the application of Grover's algorithm to symmetric cryptanalysis 7. quantum Fourier analysis and Shor's algorithm for period finding 8. the application of period finding to asymmetric cryptanalysis 9. lower bounds: the limits of quantum computing 10. symmetric cryptanalysis: Grover's algorithm and beyond 11. post-quantum asymmetric cryptography 12. quantum error correction
Special information	Students are required to understand Mathematics (namely Linear Algebra, Complex Numbers, and Probability Theory) and Theoretical Computer Science (Complexity Theory).

	N. David Mermin: Quantum Computer Science: An Introduction John Preskill: Quantum Computing in the NISQ era and beyond <https://arxiv.org/abs/1801.00862>

Module Title	Intelligent Information Systems	Module number

Semester (optional)	Frequency	Regularity and duration	ECTS credit points	Workload [hours]	Language	Module coordinator
	Every semester	During the semester, on a weekly basis	9	67.5 in-class, 160.00 self-study, 42.5 exam preparation (incl. exam). Total: 270.	English	Benno Stein

Type and application of module	Formal requirements for participation	Examination requirements
M.Sc. Computer Science for Digital Media	Admission to M.Sc. programme "Computer Science for Digital Media". See course descriptions for further requirements, if any.	The overall grade for the module is calculated as the weighted mean of the grades obtained in the component courses. See course descriptions for the examination requirements specific to the component courses.

Target qualifications

The integration of software engineering, machine learning and cognition create next-generation information systems with intelligent behavior. These intelligent information systems are also concerned with searching, accessing, retrieving, storing and treating large collections of digital media and knowledge from multiple heterogeneous sources. The goal of the module is to acquire the relevant theoretical knowledge and practical, hands-on skills for successfully using, evaluating, and developing various types of intelligent information systems. Students should

- understand the specific challenges of software engineering, search and information retrieval, machine learning and cognition
- master techniques required to analyse or develop components of intelligent information systems
- learn about the application of these techniques to specific problems and tasks
- learn to recognise the advantages of alternative approaches for solving these problems and tasks
- make well-informed decisions about an approach in order to solve problems
- recognise the state of research in a specific sub-field of intelligent information systems

More specifically, students should acquire in-depth knowledge of specific fields taught in the wider field of intelligent information systems. The specific fields are taught in component courses (see below). It is not permissible for students already to have studied these fields in depth in a previous Bachelor's programme. After completing the component courses, students should be able to undertake original research, or at least independent academic work at the Master's thesis level in these specific fields. For each component course, there is a more detailed list of target qualifications.

Contents

See course descriptions.

Didactic concept

Unless otherwise specified in the description of a component course: lectures and practical sessions combined with individual and group-based studies related to theoretical and practical aspects of the course contents. Practical sessions can include project-oriented and laboratory work based on concrete problems. Theoretical aspects can include reading, understanding and presenting recent publications. Classes consist of one 90-minute lecture and one 45-minute practical session per week during the semester. Postdoctoral researchers, doctoral students and teaching assistants supervise students and are available for intensive discussion and feedback.

Special information

See course descriptions

Lectures / courses included in the module (optional)	SWS / ECTS credit points (optional)
The module consists of two of the following courses to be chosen by the students:	
• Cognitive Systems	• 4.5
• Image Analysis and Object Recognition	• 4.5
• Introduction to Machine Learning and Data Mining	• 4.5
• Machine Learning for Software Engineering	• 4.5
• Search Algorithms	• 4.5
• Software Product Line Engineering	• 4.5
• Web Search and Information Retrieval	• 4.5

Course Title	Cognitive Systems
Coordinator	Sven Bertel
Assigned Module(s)	Intelligent Information Systems
Formal requirements for participation	Bachelor's degree in a relevant field of study
Examination requirements	Active participation in labs (minimum 50% of achievable points across all lab sections). Final oral exam (max. 45 min.).
Specific target qualifications	This course will provide a systematic introduction to the interdisciplinary field of natural and artificial cognitive systems. It will present the relevant computational and psychological concepts, theories, methods, and terminology. Students should learn about predominant theories, models, techniques, methods, and concepts of information processing in and presentation for humans as well as selected artificial systems. Students should understand the technical approaches for user simulation and modelling. Students should be able to assess selected approaches for complex problems for appropriateness and effectiveness, and should be able to justify choices of methods. Students should understand the following topics: • general and individual cognitive factors with relevance for interface design • model fit, model quality, overfitting, model validation • limits/constraints and flexibility of cognitive theories • modelling error and variation • cognitive architectures (ACT-R, SOAR, COGENT, EPIC), their structures, function and applications • artificial neural networks: architecture, training, gradient descent and back-propagation. • Bayesian models of cognition Students should be able clearly to understand the potential and limits of current-generation computational models of human cognition. Students should be able to distinguish good and bad model and be able to recommend suitable methods designed to improve a model's quality. Students should develop an understanding of the current state of research in cognitive systems. With appropriate supervision, students should be able to tackle research problems in the area.
Contents	• Natural and Artificial Cognitive systems: predominant theories, models and concepts. • Cognitive Architectures: production, connectionist, probabilistic and hybrid approaches • General Models and Individual Cognitive Abilities • Applications of cognitive systems to human-computer interaction, intelligent user interfaces, user modeling, tutoring / learning systems, (multi-media) information design • Adaptive Models plus selected other topics.
Special information	The Cambridge Handbook of Computational Psychology by Ron Sun and selected additional literature.

Course Title	Image Analysis and Object Recognition
Coordinator	Volker Rodehorst
Assigned Module(s)	Intelligent Information Systems
Formal requirements for participation	(no specific requirement for this course)
Examination requirements	Successful completion of the lab classes, final written exam
Specific target qualifications	The course gives an introduction to advanced concepts of image processing, image analysis and object recognition. The goal is to understand the principles, methods and applications of computer vision from image processing to image understanding. Students should learn the following topics: • image representation and enhancement • morphological and local filter operators • corner and edge detection • Ffiltering in frequency domain • shape detection with generalized Hough transform and Fourier descriptors • object recognition with Viola-Jones, SIFT-based voting and implicit shape models • segmentation and clustering of image regions • deep learning for visual recognition • pattern recognition methods and strategies Students should be able to apply the above topics to solve computer vision problems. Furthermore, they should appreciate the limits and constraints of the above topics. Students should be able formalise and generalise their own solutions using the above concepts of image processing, image analysis and object recognition. Students should master concepts and approaches such as • application-specific feature extraction • generation, learning and application of models for object recognition • data-driven and model-driven processing strategies in order to tackle computer-vision problems and their application to digital media. They should be able to understand proposed image analysis methods, to compare different proposals for object recognition systems, to make well-informed decisions about the preferred proposal and, if necessary, to find their own solutions to given computer vision problems. Students should develop an understanding of the current state of research in image analysis and object recognition. With appropriate supervision, students should be able to tackle research problems.
Contents	• Image Processing • Feature Extraction • Shape Detection • ObjectRrecognition • Image Regions • Machine Learning
Special information	Course material: • www.uni-weimar.de/en/media/chairs/computer-vision/teaching/image-analysis-and-object-recognition/ Literature: • B. Jähne: Digital image processing, Springer, 2005. • R.C. Gonzalez and R.E. Woods: Digital image processing, Prentice Hall, 2008. • R. Szeliski: Computer vision: algorithms and applications, Springer, 2010. • D. Forsyth and J. Ponce: Computer vision: a modern approach, Pearson, 2012. • R.O. Duda, P.E. Hart and D.G. Stork: Pattern classification, Wiley, 2000. • C.M. Bishop: Pattern recognition and machine learning, Springer, 2007.

Course Title	Machine Learning for Software Engineering
Coordinator	Norbert Siegmund
Assigned Module(s)	Intelligent Software Systems, Specialist Module
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Active participation in programming tasks (minimum 50% of achievable points across all tasks). Final oral exam (max. 45 min.).
Specific target qualifications	Nature-Inspired Machine Learning (NiML) is about learning and optimising complex tasks that are computationally intractable for exact methods. The goal of this course is to understand the principles of meta-heuristics in optimisation, as well as key concepts of learning based on neural nets. Students should understand the following techniques and theories: • problem space exploration and search-based optimization • meta-heuristics for optimization • the relationship between biological learning and optimisation with algorithms • neural nets and deep learning Students should be able to apply the above theories to solve concrete learning and optimisation problems. Furthermore, they should appreciate the limits and constraints of the individual methods above. Students should be able formalise and generalise their own solutions using the above concepts and implement them in a specified language (preferable in Python). Students should master concepts and approaches such as: • simulated annealing • swarm optimization • ant colonization • evolutionary algorithms • sampling and experimental designs • dimensionality reduction • neural nets • deep learning in order to tackle the difficulty of learning and optimising huge problems inherent to digital media. They should also be able to implement the algorithms and techniques in Python and be able to understand a proposed problem, to compare different approaches and techniques regarding applicability and accuracy, to make well-informed decisions about the preferred solution and, if necessary, to find their own solutions. Students should develop an understanding of the current state of research in optimisation and learning. With appropriate supervision, students should be able to tackle new research problems, especially in the area of search-based software engineering.
Contents	• Structure of the Search Space and General Search Techniques • Simulated Annealing and Ant Colonization • Swarm Optimization • Evolutionary Algorithms • Dimensionality Reduction and Sampling • Neural Nets • Deep Learning
Special information	Python recommended Sebastian Raschka: Python Machine Learning, Packt Publishing 2015, ISBN-13: 978-1783555130. Jeff Heaton: Artificial Intelligence for Humans, Volume 2: Nature-Inspired Algorithms, CreateSpace Independent Publishing Platform 2015, ISBN-13: 978-1499720570 Jeff Heaton: Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks, CreateSpace Independent Publishing Platform 2015, ISBN-13: 978-1505714340.

Course Title	Introduction to Machine Learning and Data Mining
Coordinator	Benno Stein
Assigned Module(s)	Intelligent Information Systems, Specialist Module
Formal requirements for participation	(no specific requirement for this course)
Examination requirements	Active participation in lab classes. Final written exam.
Specific target qualifications	Given a task and a performance measure, a computer program (and hence a machine) is said to learn from experience if its performance at the task improves with experience. In this course, students will learn to understand machine learning as a guided search in a space of possible hypotheses. The mathematical means of formulating a particular hypothesis class determines the learning paradigm, the discriminative power of a hypothesis and the complexity of the learning process. As well as the basis of supervised learning, an introduction to unsupervised learning is also provided. Students should understand the following concepts and theories: • classifier design • hypothesis space • model bias • impurity functions • statistical learning • neural networks • cluster analysis Students should be able formalise real-world decision tasks as machine learning problems. They should be able to apply the above concepts and theories to solve concrete learning problems. In particular, they should be able to choose the appropriate learning paradigm within a concrete setting. Students should master concepts and approaches such as • classifier programming • classifier application • classifier evaluation • the selection of cluster merging principles in order to tackle learning and mining problems and their application to digital media. They should be able to analyse machine learning problems, to compare different learning algorithms, and to make well-informed decisions about the prefered learning paradigm. Students should develop an understanding of current developments in machine learning. With appropriate supervision, they should be able to tackle research problems.
Contents	• Learning Examples • Linear Regression • Concept Learning • Decision Trees • Bayesian Learning • Neural Networks • Cluster Analysis
Special information	Course material: http://www.uni-weimar.de/en/media/chairs/webis/teaching/lecturenotes/#machine-learning Tools: Weka, scikit-earn, R, SciPy, GNU Octave Literature: • C.M. Bishop. *Pattern Recognition and Machine Learning* • T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning* • T. Mitchell. *Machine Learning* • P.N. Tan, M. Steinbach, V. Kumar. *Introduction to Data Mining*

Course Title	Search Algorithms
Coordinator	Benno Stein
Assigned Module(s)	Intelligent Information Systems, Specialist Module
Formal requirements for participation	(no specific requirement for this course)
Examination requirements	Active participation in lab classes. Final written exam.
Specific target qualifications	The course will introduce search algorithms as a means of solving combinatorial problems such as constraint satisfaction and optimisation problems. Tackling such problems by machine often follows a two-step approach: (1) definition of a space of solution candidates followed by (2) intelligent exploration of this space. We will cover the modeling of search problems, basic (uninformed) search algorithms, informed search algorithms, as well as hybrid combinations. Special focus will be placed on heuristic search approaches. Students should understand the following concepts and theories: • state space versus problem reduction space • uninformed search • weight functions • cost measures • informed search • admissibility of search algorithms • search monotonicity and consistency Students should be able to model a search space by selecting the appropriate representation principle and by devising encoding for partial solution bases. They should understand and describe how different search algorithms will explore the search space differently. With regard to informed search algorithms, they should understand the principle of admissible search and be able to prove basic properties of the search algorithms (completeness, soundness, admissibility). The students will learn to analyse the nature of search problems, this way being able to • devise adequate search space representations • (heuristically) inform an uninformed strategy • develop admissible search strategies • combine informed with uninformed strategies • prove important properties such as admissibility or monotonicity. Students should eventually be able to tackle non-trivial search and constraint satisfaction problems and their application to digital media. In this regard, they should be able to make well-informed decisions and explain their approach to finding solutions, considering the theoretical background. With appropriate supervision, students should be able to tackle research problems. Students should develop an understanding of the current developments of the semantic web. With appropriate supervision, they should be able to tackle research problems.
Contents	• Search Examples • Search Space Representations • Algorithms for Uninformed Search • Hybrid Search Algorithms • Algorithms for Informed Search • Theoretical Properties of Search Algorithms
Special information	Course material: http://www.uni-weimar.de/en/media/chairs/webis/teaching/lecturenotes/#search Literature: • Edmund K. Burke, Graham Kendall. *Search Methodologies* • Nils J. Nilsson. *Artificial Intelligence: A New Synthesis* • Judea Pearl. *Heuristics* • Stuart Russel, Peter Norvig. *Artificial Intelligence: A Modern Approach*

Course Title	Software Product Line Engineering
Coordinator	Norbert Siegmund
Assigned Module(s)	Intelligent Information Systems, Specialist Module
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Active participation in programming tasks (minimum 50% of achievable points from all tasks). Final oral exam (max. 45 min.).
Specific target qualifications	Software Product Line Engineering (SPLE) is about designing, managing and implementing configurable software systems and software product lines. The goal of this course is to understand the principles of variability management in non-code and code-related software artefacts, as well as key implementation techniques. Students should understand the following concepts and techniques: • variability modelling and software configuration • compile-time, load-time, and run-time variability implementation techniques • advanced programming paradigms • the pre-planning problem and the tyranny of the dominant decomposition • the separation of concerns • feature interactions and optimisation of non-functional properties Students should be able to apply the above techniques to implement concrete configurable software systems using the tool FeatureIDE. Furthermore, they should be able to compare the strengths and weaknesses of the aforementioned approaches, tools, and techniques and select the appropriate methods for the problem at hand. Students should master concepts and approaches such as • aspect-oriented programming • feature-oriented programming • component-based software development • preprocessor-based software development • frameworks, roles, meta objects and collaborations • feature models and constraints in order to realize SPLE and its application to digital media. Students should develop an understanding of the current state of research in SPLE. With appropriate supervision, students should be able to tackle research problems in overcoming the inherent difficulties resulting from the variability.
Contents	• Software Product Lines and Variability Modeling • Run-Time and Load-Time Variability • Preprocessors • Components and Frameworks • Aspect-Oriented Programming • Feature-Oriented Programming • Aspects v. Features • Analysis of Configurable Systems • Non-Functional Properties of Configurable Systems
Special information	Sven Apel, Don S. Batory, Christian Kästner, Gunter Saake: Feature-Oriented Software Product Lines - Concepts and Implementation. Springer 2013, ISBN 978-3-642-37520-0. Krzysztof Czarnecki, Ulrich Eisenecker: Generative Programming: Methods, Tools, and Applications: Methods, Techniques and Applications. Pearson Education (Us) 2010, ISBN-13: 978-0201309775.

Course Title	Web Search and Information Retrieval
Coordinator	Matthias Hagen
Assigned Module(s)	Intelligent Information Systems, Specialist Module
Formal requirements for participation	(no specific requirements)
Examination requirements	Oral exam of 30-40 minutes
Specific target qualifications	Web search engines and information retrieval today have the world's information at the users' fingertips. Research from the last few decades now helps users to find what they want for a variety of information needs in a split second. The goal of this course is to understand how search engines and IR systems work, to acquire the necessary theoretical background that enables comprehension of practical considerations, and to develop an understanding of comparison and evaluations issues. Limits and constraints and latest trends are part of the curriculum. The students should understand the following topics: • the relationship of information retrieval and web search • indexing process (offline) and query process (online) • crawling large collections with storage issues and up-to-date checks • text-processing techniques, including subtleties of parsing, stopping, stemming and anchor texts • PageRank concept and algorithm • NLP techniques for information extraction • MapReduce technique for index creation • various types of inverted indexes • index-compression concepts for efficiency • information need vs. query formulation • techniques for transforming and improving human queries • comprehending components of result presentation • mathematical models of relevance • distinguishing probabilistic retrieval models from language modeling approaches • machine learning techniques for improving ranking • Cranfield paradigm for evaluating retrieval performance • effectiveness and efficiency metrics for retrieval systems Students should be able to apply the above theories and topics to solve concrete problems in the field of retrieval systems. Furthermore, they should appreciate the limits and constraints of the respective tools and methods that make them suitable approaches in specific scenarios. Students should be able to formalise and generalise their own solutions for retrieval problems using the above theories and methods. Atudents should master concepts and approaches such as • freshness vs. age as crawling update policies • stopping and stemming to reduce index sizes vs. store everything approaches • authority ranking approaches such as PageRank • delta encoding and skip pointes for efficient small indexes • the similarityand importance of tf-components in BM25 and query likelihood retrieval models • pooling strategies in search result evaluation • precision vs. recall in effectiveness evaluations to tackle search and retrieval problems and their application to digital media. They should be able to understand typical problems faced when developing retrieval systems, to compare different approaches suited to the different components of such systems, to make well-informed decisions about the preferred approach and, if necessary, to find their own solutions to given retrieval and search problems. Students should develop an understanding of the current state of research in web search and information retrieval. With appropriate supervision, students should also be able to tackle research problems.
Contents	• Introduction • Architecture of a Search Engine • Crawling, Parsing, Information Extraction • Inverted Indexes and Index Compression • Query Processing • Retrieval Models

	• Evaluation
Special information	Tools: Lucene, Elasticsearch, Indri, Stanford NLP toolkit Literature: • Baeza-Yates, Ribeiro-Neto. Modern Information Retrieval: The Concepts and Technology behind Search • Büttcher, Clarke, Cormack. Information Retrieval: Implementing and Evaluating Search Engines • Croft, Metzler, Strohman. Search Engines: Information Retrieval in Practice • Grossman, Frieder. Information Retrieval: Algorithms and Heuristics • Manning, Raghavan, Schütze. Introduction to Information Retrieval • Van Rijsbergen. Information Retrieval • Witten, Moffat, Bell. Managing Gigabytes: Compressing and Indexing Documents and Images

Module Title	Graphical and Interactive Systems	Module number

Semester (optional)	Frequency	Regularity and duration	ECTS credit points	Workload [hours]	Language	Module coordinator
	Every semester	During the semester, on a weekly basis	9	67.5 in-class, 160.00 self-study, 42.5 exam preparation (incl. exam). Total: 270.	English	Charles Wuethrich

Type and application of module	Formal requirements for participation	Examination requirements
M.Sc. Computer Science for Digital Media	Admission t M.Sc. programme “Computer Science for Digital Media”. See course descriptions for further requirements, if any.	The overall grade for the module is calculated as the weighted mean of the grades obtained in the component courses. See course descriptions for the examination requirements specific to the component courses.

Target qualifications

Interactive Systems have become ubiquitous nowadays: they require deep knowledge of computer graphics, visualization and imaging methods, as well as a deep knowledge of interaction techniques and principles. The goal of the module is to develop an understanding of specific challenges and approaches in graphical and interactive systems, from both the graphical and interaction point-of-view. Students should

- learn about the specific challenges in interaction systems, mobile and ubiquitous computing and/or graphical systems and virtual reality
- master techniques required to develop graphical applications and interactive, mobile and ubiquitous devices.
- learn about current state of research in one of these fields.
- make well-informed decisions about approaches in order to solve problems or develop new devices and systems.

More specifically, students should acquire in-depth knowledge of specific fields taught in the wider fields of graphical and interactive systems. The specific fields are taught in component courses (see below). It is not permissible for students already to have studied these fields in depth in a previous Bachelor's programme. After completing the component courses, students should be able to undertake original research, or at least independent academic work at the Master's thesis level in these specific fields. For each component course, there is a more detailed list of target qualifications.

Contents

See course descriptions.

Didactic concept

Unless otherwise specified in the description of a component course: lectures and practical sessions combined with individual and group-based studies related to theoretical and practical aspects of the course contents. Practical sessions can include project-oriented and laboratory work based on concrete problems. Theoretical aspects can include reading, understanding and presenting recent publications. Classes consist of one 90-minute lecture and one 45-minute practical session per week during the semester. Postdoctoral researchers, doctoral students and teaching assistants supervise students and are available for intensive discussion and feedback.

Special information

See course descriptions

Lectures / courses included in the module (optional)	SWS / ECTS credit points (optional)
The module consists of two of the following courses to be chosen by the students:	
• Advanced HCI: Theory and Methods	• 4.5
• Advanced HCI: Ubiquitous Computing	• 4.5
• Computer Graphics II: Animation Systems	• 4.5
• Computer Graphics II: Fundamentals of Imaging	• 4.5
• Mobile Information Systems	• 4.5
• Spatial Computational Geometry	• 4.5
• Usability Engineering	• 4.5
• Virtual Reality	• 4.5

Course Title	Spatial Computational Geometry
Coordinator	Bernd Fröhlich
Assigned Module(s)	Graphical and Interactive Systems, Specialist Module By special application to the examination committee: Modeling
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Active participation in the lab class; a score of 50% of the assignments and the final project needs to be achieved for admission to the final exam. Final oral exam (max. 45 min.).
Specific target qualifications	The goal of this course is to provide students with the theoretical and applied foundations for the design and analysis of efficient algorithms for problems involving geometric input and output. The course focuses on real-time problems in 2D- and 3D-graphics and visualization applications. Students should understand the following constructs, techniques, algorithms and efficient data structures: • convex hulls • plane sweep and segment intersection computations • point localization • doubly-connected edge list data structures • range searching • window searching • Voronoi diagrams • Delaunay triangulation • ray queries Students should be able to implement the above algorithms and data structures to solve concrete problems. Furthermore, they should be able to analyse the complexity of the algorithms and data structures.
Contents	• Introduction • Fundamentals • Convex Hulls • Plane Sweep and Segment Intersections • Point Localization • Doubly-Connected Edge List • Range Searching • Window Searching • Voronoi Diagrams • Delaunay Triangulation • Ray Queries
Special information	This course is partially based on the book *Computational Geometry, Algorithms and Applications* by Mark de Berg, Otfried Cheong, Marc van Kreveld and Mark Overmars. Further references will be provided throughout the course.

Course Title	Computer Graphics: Animation Systems
Coordinator	Charles Wuethrich
Assigned Module(s)	Graphical and Interactive Systems
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Conception and submission of a computer animation. Algorithm study. Final exam.
Specific target qualifications	Computer animations and animation systems have achieved quite widespread use. This course has a double aim; to allow students to understand the algorithm and modelling techniques used in common high level animation systems, and at the same time be able to appreciate the hard work involved in the production process of a computer animation. Successful students in this course should understand and be able to programme the underlying algorithms and physics used in a 3D-animation program and to cooperate with artists and designers on a common 3D-animation project, which might involve the programming of plug-ins for the animation system. At the end of the course, they should have mastered the conception, design and implementation of 3D-animation software.
Contents	Contents: • Animation Principles. • Interpolation, Spline families. • Motion Control, Arc length in 3D-curves, Time-Velocity-Acceleration, SLERP, Quaternions. • Deformations, Topology, Genus of Surfaces, Morphing, 3D-Morphing. • Kinematics, Inverse Kinematics, Jacobians, Solving Kinematics with the Aid of the Jacobian. • Physics 101: Linear and Angular Physical Equations, Mass-Spring Systems, Navier Stokes Equations, Motion equations in Animation. Solution Methods for Differential Equations. • Collision Detection, Computing Collision Response. • L-Systems. Natural Phenomena: plants, particles, water, flocks, intelligent agents, clouds, fire. • Animal and Human animation: walking, running, jumping. • Motion Tracking, Fitting Tracked Data to 3D-Models.
Special information	

Course Title	Advanced HCI: Theory and Methods
Coordinator	Hornecker
Assigned Module(s)	Graphical and Interactive Systems
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Submission of practical project- and problem-based coursework in combination with presentations and technical discussions. Final exam.
Specific target qualifications	Students should have an understanding of the difference between quantitative and qualitative methods. They should master core HCI research methods and theories for understanding and analysing human interaction with technology. They should be aware of how the role of theory in HCI has expanded from the early days of human factors and mathematical modeling of behaviour to include explanatory and generative theories, which reflect influences from fields such as design, sociology and ethnography. Students should know how to apply core HCI methods to novel (and real-world) problems and tasks. Students should be able to run studies using appropriate data gathering or evaluation techniques and methods, in particular qualitative methods (interviews, observation), to adapt and adjust these in light of the given research question and use context, and to justify research method and study design. They should understand and be able to discuss complex HCI issues from the research literature for emerging areas of human-computer interaction and be able to engage with the literature and acquire other methods independently. With appropriate supervision, students should be able to tackle research problems. In addition, social and general transferable skills are trained via group work in the classes, based on concrete problems and tasks.
Contents	Sample contents are: • Role of Theory in HCI, the History of HCI theory and Method Use • General Styles of HCI Research Methods (qualitative and quantitative) • Experimental Study Design and Statistical Analysis • Interviews, Questionnaires, Observation Methods and Approaches • Ethnography and Field Studies • Data Analysis for Qualitative Studies
Special information	Introductory Literature: • Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. Research Methods in Human-computer Interaction. Wiley Publishers • Judith S. Olson, Wendy A. Kellogg (eds.) Ways of Knowing in HCI. Springer 2014 • Yvonne Rogers. HCI Theory. Classical, Modern, and Contemporary. Morgan & Claypool Publishers 2012 • Ann Blandford, Dominic Furniss, Stephann Makri. Qualitative HCI Research. Going behind the sceneds. Morgan and Claypool Publishers2016

Course Title	Computer Graphics: Fundamentals of Imaging
Coordinator	Charles Wuethrich
Assigned Module(s)	Graphical and Interactive Systems
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Presentation, discussion, implementation and submission of imaging algorithms. Final exam.
Specific target qualifications	Modern Digital Imaging Devices are ubiquitous nowadays. The goal of this course is to understand the principles of imaging and to be able to conceive, design and implement systems relevant for imaging. Students should understand the following topics: • The physics of optics and its associated quantities, light and radiometry, geometrical optics and lenses. • Human vision, photometry, colorimentry, color spaces. • Photographic rules, composition, aperture, field of view. • Analog and digital capturing devices, light sensors. • Advanced methods and functions for assessing image quality. • Enhancing algorithms to overcome and correct capturing shortcomings. • Factors leading to imaging quality. At the end of the course, they should have mastered the conception, design and implementation of imaging software for both generic digital light sensors and digital photography.
Contents	Contents: • Light and Radiomentry. • Human Vision, Photometry, Colorimentry. Advanced Color Spaces. • Geometrical Optics and Lenses. Optical Equations for Lense Systems. • Photographic Composition, quantities used in photography. • Analog Photography. • Digital Sensors. • Image Enhancing, Debayering, Filtering, Edge Enhancement. • Image Quality Assessment. • Use of Fourier, Cosine and Wavelet Transforms in Imaging.
Special information	

Course Title	Advanced HCI: UbiComp
Coordinator	Hornecker
Assigned Module(s)	Graphical and Interactive Systems
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Submission of practical project- and problem-based coursework in combination with presentations and technical discussions. Final exam.
Specific target qualifications	Students should have an understanding of theoretical, applied and technical foundations of modern ubiquitous computing systems. They should understand how such Ubicomp systems work on a technical level and also understand their societal relevance. They should know about the technical and social-design-related challenges in developing such systems. They should be able critically to assess societal implications and discuss design trade-offs. They should be able to develop concpets for novel UbiComp applications, to determine their technical feasibility, and to reflect critically on their feasibility in an application context. Moreover, they should be able to apply a user-centered approach in the design process of UbiComp applications. Students should understand and be able to discuss complex issues from the HCI and UbiComp research literature for emerging areas of UbiComp and be able to engage with the literature. With appropriate supervision, students should be able to tackle research problems. In addition, social and general transferable skills are trained via group work in the classes based on concrete problems and tasks.
Contents	Contents: • History of Ubicomp Systems • Different Views of UbiComp • Sensing, Tracking and Monitoring Technology, Location Sensing • Interface Types: mobile, tangible, touch interfaces • Prototyping and Research Methods in the UbiComp Field • Modern User Interfaces for Ubicomp Systems • Role of the Use Context • User-Centered Design for Development of Novel Technologies, e.g. UbiComp • Societal, Ethical and User-Research Issues for Novel Technologies
Special information	Introductory Literature: • Ubiquitous Computing Fundamentals. Ed. John Krumm. ISBN: 1420093606. Chapman & Hall/CRC 2009. • Harper, Rodden, Rogers, Sellen (eds.). Being Human: Human-Computer Interaction in the Year 2020. Microsoft Research Ltd 2008

Course Title	Usability Engineering and Usability testing
Coordinator	Sven Bertel
Assigned Module(s)	Graphical and Interactive Systems
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Active participation in labs (minimum 50% of achievable points across all lab sections). Final oral exam (max. 45 min.).
Specific target qualifications	Participants should learn about the various factors that determine a system's usability, as well as how to test for them, how to formulate recommendations towards improving a system's usability and how successfully to accompany processes of implementing such recommendations. Students should understand the following topics: • proactive, descriptive and remedial aspects of system usability • formative and summative evaluation • the ISO9241 norm • design methods, including interaction, scenario-based, user-centred design • stakeholders and requirement engineering • reading and formulating case studies, user stories, scenarios • quantitative and qualitative methods of data gathering • internal and external validity, reliability • descriptive, relational and experimental methods of behavioural research • low-, mid-, and high-fidelity prototype testing • formulating and testing experimental hypotheses • sampling strategies and confidence intervals • descriptive and inferential statistics (variance analysis, correlation, regression, general linear models) • significance testing • modeling error, cumulated alpha-error, omnibus testing • parametric and non-parametric (e.g. frequency- and rank-based) methods • power, effect size, sensitivity • experimental design, main effects, interactions • model fitting, overfitting, model validation • graphic data presentation, reporting statistics Students should master the design of behavioural experiments with users to test hypotheses and be able to use appropriate methods for data analysis. They should be able clearly to understand the limits of statistical inferences that can be drawn from an experiment. Students should be able to distinguish good and bad usability and to recommend suitable methods for increasing a system's usability. Students should develop an understanding of the current state of research in usability. With appropriate supervision, students should be able to tackle research problems in the area.
Contents	• Factors that Determine a System's Usability • Usability Engineering Lifecycles • Testing for Usability: goals, theories, methods, techniques • Parametric and Non-Parametric Methods of Experimental Statistics • Formulating Requirements • Usability Heuristics • Designing and Running an Experiment • Usability Engineering for Specific Systems and Specific User Groups • Issues of Standardization • Designing for Usability plus further selected topics.
Special information	Lazar et al. (2009): Research Methods in Human-Computer Interaction, Wiley. Rosson & Carroll (2002): Usability Engineering. Morgan Kaufmann. Rubin & Chisnell (2008): Handbook of Usability Testing, 2nd edition. Wiley. Field (2013): Discovering Statistics Using IBM SPSS Statistics. Sage.

Course Title	Virtual Reality
Coordinator	Bernd Fröhlich
Assigned Module(s)	Graphical and Interactive Systems
Formal requirements for participation	(no specific requirements for this course)
Examination requirements	Active participation in the lab class; a score of 50% of the assignments and the final project needs to be achieved for admittance to the final exam. Final oral exam (max. 45 min.)
Specific target qualifications	The goal of this course is to provide students with the theoretical, technical and applied foundations of modern virtual reality systems, 3D-cinema, stereoscopic gaming and 3D-user interfaces. Students should understand the following concepts, techniques and technical systems: • scenegraph technology • viewing in 3D • 3D-perception • stereoscopic single- and multi-viewer display technology • three-dimensional user interfaces and interaction techniques Students should be able to apply the above concepts, techniques and their knowledge of technical solutions to solve concrete problems. Furthermore, they should be able identify and discuss the main usability factors of 3D-interaction techniques, 3D-interfaces and 3D-display technology. Students should master concepts and approaches such as • computing stereoscopic projection parameters for various technical setups • designing a scenegraph-based interactive virtual reality application that supports multiple users • selecting navigation, selection and manipulation techniques for specific use cases • using Fitts's law and the steering law to evaluate the performance of design decisions in selection and navigation tasks • the design and parametrization of transfer functions for the different types of sensors and tasks • assessing the optical efficiency of various projection technologies in order to tackle problems from the virtual reality domain. Students should develop an understanding of the basics and the current state of research in virtual reality and make well-informed decision in this context. They should be able to discuss research problems, implement current approaches and understand the limitations of the solutions.
Contents	• Introduction • Stereoscopic Viewing • Graphics and Scenegraph Basics • Viewing Setups in Scenegraphs • 3D-User Interface Basics • Navigation • 3D-Selection • 3D-Manipulation • 3D-Input Devices • 3D-Display Technology Basics • Stereoscopic Multi-User Display Technology • Interaction and Collaboration in Multi-User Virtual Reality • Introduction to Augmented/Mixed Reality
Special information	This course is mostly based on recent research publications. References will be provided throughout the course.

Module Title	Specialization	Module number

Semester (optional)	Frequency	Regularity and duration	ECTS credit points	Workload [hours]	Language	Module coordinator
	Every semester	During the semester, on a weekly basis	9	67.5 in-class, 160.00 self-study, 42.5 exam preparation (incl. exam). Total: 270.	English	Stefan Lucks

Type and application of module	Formal requirements for participation	Examination requirements
M.Sc. Computer Science for Digital Media	Admission to M.Sc. programme "Computer Science for Digital Media". See course descriptions for further requirements, if any.	The overall grade for the module is calculated as the weighted mean of the grades obtained in the component courses. See course descriptions for the examination requirements specific to the component courses.

Target qualifications

Students shall acquire in-depth knowledge of specialist topics from Computer Science and its application to Digital Media.

Most of courses offered for the modules Modelling, Graphical and Interactive Systems, Intelligent Information Systems and Distributed and Secure Systems are also open to the Specialization module. The student can pick two such courses, which have not been taken for any of the above modules.

Contents

See course descriptions.

Didactic concept

Unless otherwise specified in the description of a component course: lectures and practical sessions combined with individual and group-based studies related to theoretical and practical aspects of the course contents. Practical sessions can include project-oriented and laboratory work based on concrete problems. Theoretical aspects can include reading, understanding and presenting recent publications. Classes consist of one 90-minute lecture and one 45-minute practical session per week during the semester. Postdoctoral researchers, doctoral students and teaching assistants supervise students and are available for intensive discussion and feedback.

Special information

See course descriptions

Lectures / courses included in the module (optional)	SWS / ECTS credit points (optional)

Module Title	Electives	Module number

Semester (optional)	Frequency	Regularity, duration	ECTS credit points	Workload [hours]	Language	Module coordinator
	Every semester	On a weekly basis during the semester	12	360	English (students may also choose courses in German)	examination committee

Type and application of module	Formal requirements for participation	Examination requirements

Qualification Goals

The module enables students to:

1. acquire in-depth knowledge of specialist topics in Computer Science
2. broaden their academic knowledge in other fields
3. improve their English, or, in the case of non-native speakers, German.

Course Contents

(depends on course(s) chosen)

Didactic Concept

(depends on course(s) chosen)

Special information

Students may choose from the following course types:

1. Master's courses in Computer Science offered by professors from the Computer Science department.

   In the case of a research project offered by a professor from the Computer Science department, students need the examination committee's permission to count this project towards their electives module.

2. Non-Computer Science Bachelor's and Master's courses offered by professors from other Faculties or other departments of the Faculty of Media.

   A course does NOT fall into this category if parts of the content or some of the qualification goals are typical of Computer Science, such as programming skills, writing scripts in an executable language, or dealing with the internals of binary communication protocols. The examination committee can restrict or reject the validity of credit points from a course if it does not fall into this category.

3. English or German Language courses offered at Bauhaus-University for non-native speakers.

   The validity of language courses is limited to a maximum of 6 credit points for the whole electives module.

Upon application, the examination committee can adjudicate on the validity of other courses for this module.

Lectures / Courses included in the module (optional)	SWS / ECTS credit points (optional)

Course Title	Spatial information systems (GIS)
Coordinator	Volker Rodehorst
Assigned Module(s)	Electives
Formal requirements for participation	(no specific requirement for this course)
Examination requirements	Successful completion of the lab classes, final written exam.
Specific target qualifications	The course covers advanced basics of spatial information systems (GIS), such as acquisition, organization, analysis and presentation of data with spatial reference. The practical classes and the individual project lead to a deeper understanding of GIS workflows, tools and extensions and should turn knowledge into practice. Students should understand the following topics: • primary and secondary spatial reference • data types and dimensions of geo-objects • coordinate reference systems and map projections • acquisition of geospatial base data and available online resources • object-relational database management systems • efficient tree-structures for spatial data • graphical GIS modeling in UML • 3D city models • spatial interpolation and analysis of vector-based geo-objects • route planning and traveling salesman problem • cartographic visualization and generalization Students should be able to apply the above topics to solve problems with spatial reference. Furthermore, they should appreciate the limits and constraints of the above topics. Students should be able formalise and generalise their own solutions using the above concepts of acquisition, organization, analysis and presentation of geospatial data. Students should master concepts and approaches such as • conceptual design and realization of a GIS • collection of subject-specific geospatial data • application for location-based services, geo-marketing and strategic site planning in order to tackle problems of spatial information systems and its application to digital media. They should be able to understand the proposed concepts, to compare different proposals for GIS systems, to make well-informed decisions about the preferred proposal and, if necessary, to find their own solutions to given problems with spatial reference. Students should develop an understanding of the current state of research in spatial information systems. With appropriate supervision, students should be able to tackle research problems.
Contents	• Acquisition of spatial data • Spatial data management • Object-oriented data modeling • Spatial data analysis • Presentation of spatial data • GIS applications
Special information	Course material: • www.uni-weimar.de/en/media/chairs/computer-vision/teaching/spatial-information-systems-gis/ Literature: • R. Bill: Grundlagen der Geo-Informationssysteme, 6. Ed., Wichmann, 2016. • M. de Smith, M. Goodchild and D. Longley: Geospatial Analysis, 2009. • N. Bartelme: Geoinformatik – Modelle, Strukturen, Funktionen, 4. Ed., Springer, 2005. • N. de Lange: Geoinformatik in Theorie und Praxis, 2. Ed., Springer, 2006.

Course Title	Photogrammetric Computer Vision
Coordinator	Volker Rodehorst
Assigned Module(s)	Electives
Formal requirements for participation	(no specific requirement for this course)
Examination requirements	Successful completion of the lab classes. Final written exam.
Specific target qualifications	The course gives an introduction to the basic concepts of sensor orientation and 3D reconstruction. The goal is an understanding of the principles, methods and applications of image-based measurement. Students should learn about the following topics: • homogeneous representation of points, lines and planes • planar and spatial transformations • stimation of relations using a direct linear transformation (DLT) • modeling and interpretation of a camera • optical imaging with lenses • epipolar geometry and multi-view tensors • global bundle adjustment • robust parameter estimation • image-matching strategies Students should be able to apply knowledge of the above topics to solve photogrammetric problems. Furthermore, they should appreciate the limits and constraints of the above topics. Students should be able formalise and generalise their own solutions using the above concepts of sensor orientation and 3D reconstruction. Students should master concepts and approaches such as • algebraic projective geometry • reconstruction and inversion of the imaging geometry • the correspondence problem in order to tackle problems in photogrammetry and its application to digital media. They should be able to understand proposed sensor orientation problems, to compare different proposals for image-based 3D reconstruction systems, to make well-informed decisions about the preferred proposal and, if necessary, to find their own solutions to given problems in photogrammetry. Students should develop an understanding of the current state of research in photogrammetric computer vision. With appropriate supervision, students should be able to tackle research problems.
Contents	• Image-based 3D Reconstruction • Homogeneous Coordinates • Algebraic Projective 2D and 3D Geometry • Camera Calibration • Sensor Orientation Using Multi-View Geometry • Stereo Image Matching
Special information	Course material: • www.uni-weimar.de/en/media/chairs/computer-vision/teaching/photogrammetric-computer-vision/ Literature: • W. Förstner and B.P. Wrobel: Photogrammetric Computer Vision – Statistics, Geometry, Orientation and Reconstruction, Springer, 2016. • R. Hartley and A. Zisserman: Multiple View Geometry in Computer Vision, 2. Edition, Cambridge University Press, 2003. • O. Faugeras and Q.-T. Luong: The Geometry from Multiple Images, MIT Press, 2004. • Y. Ma, S. Soatto, J. Kosecka and S. Sastry: An Invitation to 3D-Vision – From Images to Geometric Models, 2. Edition, Springer, 2005. • R. Szeliski: Computer vision: algorithms and applications, Springer, 2010.

Module Title	Research Project I and II	Module number

Semester (optional)	Frequency	Regularity and duration	ECTS credit points	Workload [hours]	Language	Module coordinator
	Every semester	Over course of one semester	15	45h in organized meetings/ classes and 405h self-study. Total: 450h	English	Respective Professorship

Type and application of module	Formal requirements for participation	Examination requirements
M.Sc. Computer Science for Digital Media Can be open to M.Sc HCI	Admission to M.Sc programme "Computer Science for Digital Media". See course descriptions for further requirements, if any.	Completion of a body of work and its documentation, usually in the form of a scientific report. Specific criteria for evaluation will be announced in the course catalogue and at the beginning of the individual project. Quality of the presentation, results achieved, autonomy in work and creativity are important factors.

Target qualifications

Depending on the type of project, students should have gained practical experience with the design, implementation and evaluation of advanced software systems and their user interfaces. Students may also have gained practical experience in designing, planning and running user studies related to specific user interface technologies.

Participants refine their presentation skills via independent literature research based on current publications and presentations on the various aspects and milestones of the project. An evaluation and documentation of the results in the form of a scientific report completes the project. As a result of various types of activities involving presentations, participants have experience in presenting and explaining their work in oral and written form. They understand the importance of project management and organisation for complex projects and are accustomed to acquiring new skills and knowledge in self-study.

Projects require considerable autonomy from students and develop social and general transferable skills via group work and independent research (team work, self-organisation, project management).

Contents

Depends on individual topic

Within the project, students work on research topics in close collaboration with the supervising professors and their research assistants. In many cases, the projects focus on the design, implementation and evaluation of software systems and their user interfaces with a particular emphasis on teamwork. Projects may also focus on designing, planning and running user studies related to specific user interface technologies.

Projects will often produce a body of practical work or a working system, and a scientific report, or may predominantly result in a more in-depth scientific report.

Didactic concept

Projects confront students with complex problems of scientific relevance and require, as well as develop autonomy and creativity, problem-solving skills and team work. They are at the core of the Bauhaus tradition of teaching.
Typically, project teams meet once a week with the supervising professors and their research assistants. The majority of effort consists of autonomous self-study.

Special information

Projects on offer are announced in the teaching catalogue for each semester and presented at the project fair at the start of the semester.

Lectures / courses included in the module (optional)	SWS / ECTS credit points (optional)
(none)	

Module Title	Master's Thesis Module	Module number

Semester (optional)	Frequency	Regularity and duration	ECTS credit points	Workload [hours]	Language	Module coordinator
	Every semester	Any time, 4 months	33	790h in self-study, 20h in meetings with the supervisor, and 180h for the defence and its preparation (1h for the defence itself). Total: 990.	English	Respective professorship

Type and application of module	Formal requirements for participation	Examination requirements
M.Sc. Computer Science for Digital Media	At least 60 ECTS of the Master's programme have to be successfully completed. English proficiency at C1 level (CERT).	Written thesis in the style of an academic publication (weight 80%) and a related defence (weight 20%)

Target qualifications

In the thesis, the students prove their ability to perform independent academic work in Computer Science on an adequately challenging topic within a given time frame. They use established methods or adapt existing approaches while adhering to standards of academic work. They are given the opportunity to develop, refine and formulate their own ideas and work critically with the literature.

Contents

Depends on individual topic

Didactic concept

The module conists of three phases

1. Initial research for the thesis. At the end of the initial research, the topic and time frame are fixed. Duration: about 2-3 months, in parallel with courses/projects.
2. Core research for the thesis. At the end of this phase, the written thesis has to be finished. Duration: 4 months.
3. Defence of the thesis. With a few weeks of preparation, the student will present the motivation and the main results from the thesis.

The students work largely independently, with regular intermediate reporting and consultation with the supervisor.

Special information

The final thesis is the most important part of the module. It describes the results as well as the path that led to these results. The thesis should be written in the style of an academic publication, whereby the student's own contribution to the results should be clearly evident. The evaluation of the thesis comprises a grade for the written thesis (weighted at 80%) and a combined grade for the presentation and the related defence (weighted at 20%).

Lectures / courses included in the module (optional)	SWS / ECTS credit points (optional)