

III. Modules in the Masters Degree Course

III.1 Field: Software Technology and Information Systems

III.1.1 Model-based software development

Role in the computer science degree course

In a model-based design, models abstractly describing the system to be developed are the central parts. Models are developed in early design phases and on different levels of abstraction, which describe the system in different detail. Model transformations are used to transfer models of one level of abstraction into another. One such transformation is the final code generation. A model-based design process supports the general Software Engineering principles of abstraction and structuring. It furthermore enables an early analysis of the design and thus improves its quality.

Content structure of the module

For this module, students will select two advanced classes from the following list:

- Generating Software from Specifications
- Software Quality Assurance
- Web Engineering
- Modelchecking
- Deductive Verification
- Software Safety

The classes are structured as follows:

- Generating Software from Specifications:
see Module Languages and Programming Methods
- Software Quality Assurance:
 1. Standards (ISO 9126, CMM-I, SPICE, ISTQB, ...)
 2. Constructive Approaches (Domain-specific Languages, Meta-Modelling, Architectural Styles, Patterns, MDA, ...)
 3. Analytical Approaches (Reviews, Inspections, Black-Box, White-Box-Testing, Model-based Testing,...)
- Web Engineering
 1. Web Applications
 - Categories / Characteristics
 - Modeling Approaches (WebML, UWE, ...)
 - Web Technologies (XML, CGI, JSP, JSF, PHP, AJAX, ...)
 2. Web Services
 - Standards (WSDL, SOAP, UDDI)
 - Visual Contracts

3. Service-Oriented Architectures (SOA)

- Concepts, Notions
- Development Methods

- Modelchecking:
see Module Analytical Methods in Software Engineering
- Deductive Verification:
see Module Analytical Methods in Software Engineering
- Software Safety:
see Module Constructive Methods in Software Engineering

Usability of the content

This module teaches knowledge and skills in the area of quality assurance in a model-based design as well as the design techniques themselves. The students in particular learn to choose design as well as quality assurance techniques. The students can use the abilities gained in this module anywhere in their study and professional work where the design of high-quality software is required.

Prerequisites and prior knowledge

- basic knowledge in software design, as taught in the courses “Software design” and “model-based software design” and /or knowledge of formal modeling techniques as taught in the courses “Modelling” or “Software design with formal methods”.

Learning goals

Teaching of factual knowledge

After completing this module, the student will

- understand quality assurance techniques
- understand software design techniques
- understand design techniques for safety-critical systems

Teaching of methodological knowledge

After completing this module, the student will be able to

- select and use languages and methods for developing (large, web-based) software applications
- use industrial standards
- apply model-based software development techniques

Teaching of transfer skills

After completing this module, the student will be able to

- apply the skills and abilities gained in this module in any domain, also ones different from those of the module

Teaching of normative evaluation skills

After completing this module, the student will be able to

- evaluate and improve existing design processes
- evaluate the applicability of different techniques for a particular application domain

Key qualifications

For this module, we expect for you to

- co-operate and work in teams during the tutorial classes
- apply strategies for acquiring knowledge
- combine lectures, prepare and review the lecture material, participate in tutorial classes with supervised group work, prepare homework, and take part in class tutorials.

Module assignment

Elective module in the area of software technology and information systems

Mode

Credit points: 4+4 ECTS (2 catalog classes)

SWS (hours per week): 2+1, 2+1

Frequency: 2-3 catalog classes per year in the winter and summer semesters

Methods of implementation

- We introduce and explain methods and technologies using typical examples
- They are then applied in practice during the tutorials (all classes)
- In the tutorials, students carry out projects in supervised, small groups (GSS)

Organisational arrangements / media use / literature references

- Lectures with overhead presentation
- Tutorial classes in small groups
- Some supervised laboratory work
- Web-based lecture material

Examination modalities

Oral examination if fewer than 40 participants: otherwise, written examination in every catalog class

Person responsible for the module

Engels

III.1.2 Languages and Programming Methods

Role in the computer science degree course

Languages play various important roles in software technology: As programming languages, they are a means of expression for program development and are tailor-made for a specific programming method. As specification languages, computer scientists use them for formulating task descriptions in general, or as tailor-made description methods for specific application areas. Not only the methodically well-founded use, but also designing and implementing such languages by compilers or generators are important topics in software technology.

This module teaches knowledge and skills for the in-depth understanding, specifying, and implementing programming and specification languages. It offers the extension of this topic in a choice of two areas of language implementation or programming methods. This module enables the participants

- to acquire special methods for analysing or synthesising programs, or
- to apply programming methods for object-oriented, parallel, functional, logical or web-based paradigms, or
- to design and implement specification languages for application-specific software generators.

This course builds on the knowledge of calculi for describing language features and on methods for implementing languages.

Content structure of the module

For this module, students will select two advanced classes from the following list:

- Compilation Methods
- Generation of Software from Specifications
- Object-Oriented Programming
- Parallel Programming in Java
- Functional Programming
- Scripting Languages
- Prolog with Applications

The classes are structured as follows

- Compilation Methods
 1. Optimisation of intermediate code
 2. Code generation
 3. Register allocation
 4. Code parallelisation

- Generation of Software from Specifications:
 1. Reuse and generators
 2. Generation of structured texts
 3. Constructing trees
 4. Computations on trees
 5. Names and attributes
 6. Language design
 7. Projects

- Object-Oriented Programming (in German):
 1. Paradigms for the use of inheritance
 2. Separate design with design patterns
 3. Libraries and frameworks
 4. Design errors
 5. Beyond Java

- Parallel Programming in Java:
 1. Monitors and their systematic development
 2. Barriers: Application and implementation
 3. Loop parallelisation
 4. Programming with asynchronous messages
 5. Programming with synchronous messages

- Functional Programming (in German):
 1. Recursion paradigms
 2. Function schemes
 3. Type structures
 4. Functions as data
 5. Data streams and lazy evaluation
 6. Fixpoints, functional algebra

- Scripting Languages (in German):
 1. Static and dynamic web applications, HTML
 2. Client-side scripting with JavaScript
 3. Server-side scripting with Perl
 4. Server-side scripting with PHP

- Prolog with Applications:
See Module Databases and Information Systems

Usability of the content

Students can apply the knowledge and abilities gained in this module anywhere in their study and professional work where a deeper understanding of programming or

specification languages is required. In this context, we have oriented the class Generating Software from Specifications more toward developing language-based tools, whereas the other classes teach methods for using languages.

Prerequisites and prior knowledge

- basic knowledge in a language suitable for software development and individual practical experience with it in program development as it is taught in the classes GP1, GP2 and the software technology practical during the Bachelor course;
- understanding of general language features and of non-imperative programming paradigms as taught in GPS in the Bachelor course;
- knowledge of fundamental methods for the specification and implementation of language features as they are taught in Programming Languages and Compilers.

Learning goals

Teaching of factual knowledge

After completing this module, the student will

- understand advanced techniques for language implementation
- learn language constructs for specific programming paradigms and specification calculi

Teaching of methodological knowledge

After completing this module, the student will be able to

- apply generators and standard solutions for language implementation
- systematically apply methods of specific programming paradigms

Teaching of transfer skills

After completing this module, the student will be able to

- specify languages for new application tasks and implement them using generators
- transfer programming methods to future languages and to their implementations

Teaching of normative evaluation skills

After completing this module, the student will be able to recognise

- the clarity and problem focus of functional program and data formulations
- the value of systematic methods of program development and language implementation

Key qualifications

For this module, we expect for you to

- co-operate and work in teams during the tutorial classes

- apply strategies for acquiring knowledge
- combine lectures, prepare and review the lecture material, participate in tutorial classes with supervised group work, prepare homework, and take part in class tutorials.

Module assignment

Elective module in the area of Software Technology and Information Systems

Mode

Credit points: 4+4 ECTS (2 catalog classes)

SWS (hours per week): 2+1, 2+1

Frequency: 2-3 catalog classes per year in the winter and summer semesters

Methods of implementation

- We introduce and explain methods and technologies using typical examples
- They are then applied in practice during the tutorials (all classes)
- In the tutorials, students carry out projects in supervised, small groups (GSS)

Organisational arrangements / media use / literature references

- Lectures with overhead presentation
- Tutorial classes in small groups
- Some supervised laboratory work
- Web-based lecture material

Examination modalities

Oral examination if fewer than 60 participants: otherwise, written examination in every catalog class

Person responsible for the module

Kastens

III.1.3 Databases and Information Systems

Role in the computer science degree course

Databases play a central role in enterprises because the most important assets that a company has are its data. Therefore, data integrity, data safety, controlled data access, and efficient data processing as provided by database systems are key issues of each company - and a “must” for computer scientists to know about. Furthermore, in modern information systems and in the World Wide Web, documents and semi-structured data are exchanged largely based on modern data exchange formats such as XML. The course

DBIS2 gives the students a deep understanding of these topics and teaches the practical skills needed to manage projects involving databases.

Furthermore, knowledge management and web information management and different ways to combine data in order to infer further knowledge or to derive strategies or plans, become major challenges of today's companies. These topics are learned in the other courses offered in this module, like propositional proof systems, web engineering, Skriptsprachen, and Prolog.

Content structure of the module's classes

The module consists of the classes

- Databases and Information Systems 2 and
- Prolog mit Anwendungen (in German):

or one of these classes and a further different class from the following list:

- Propositional Proof Systems
- Scripting Languages
- Web Engineering
- Machine Learning
- a seminar related to this module

These classes are structured as follows:

- Databases and Information Systems 2
 1. XML data compression
 2. XML-relational Mapping and web-database coupling
 3. Data integration and query optimization based on XPath, XQuery, and XSLT
 4. Transactions in mobile ad-hoc networks
 5. Query optimization and chaining in distributed and mobile databases
 6. Access control and privacy
- Prolog mit Anwendungen (in German)
 1. Question and answer systems for natural language
 2. automated translation
 3. search, puzzles, and strategy games
 4. rewrite and inference systems
 5. meta-interpreters and language extensions
- Propositional Proof Systems
→ see class description in the module of Knowledge Based Systems
- Web Engineering
→ see class description for Model-Based Software Development
- Scripting Languages

→ see class description for Languages and Programming Methods

Usability of the content

Students will apply the knowledge and skills gained in practice in many companies. The knowledge and skills are also consolidated in seminars that build directly on the module's central class (DBIS2) and form an ideal basis for the master theses.

Prerequisites and prior knowledge

We assume that the students know the contents of the classes

- Foundations of Database Systems
- Concepts and Methods of System Software
- Databases and Information Systems 1
- Solid programming skills in Java, as taught in the tutorials to the class Foundations of Program Development

Learning goals of the module

Teaching of factual knowledge

After completing this module, the student will have learned

- the principle of operation of non-standard data models and database system concepts (XML, distributed databases)
- the basic concepts and structure of parsers, interpreters, and natural language processing systems

Teaching of methodological knowledge

In small groups during tutorial classes, the student will learn

- to use system components in database systems (for example, query optimization, transaction management) correctly and appropriately
- to formulate arbitrary queries and write operations in arbitrary data models
- to integrate arbitrary database accesses into a web application during practical laboratory tutorials:
- to build their individual databases, web servers, and information systems
- to read, transform, or modify XML data
- work properly with important standards used in industry, i.e. XML, XPath, XSLT, DOM, SAX, SOAP and others.

Teaching of transfer skills

After completing this module, the student will be able to

- transfer the acquired knowledge and skills to other data sources or other database systems, other web servers, and other server technologies

Teaching of normative evaluation skills

After completing this module, the student will be able to

- assess the suitability of various data models (relational, XML) for different applications
- estimate familiarization times for database and web technologies and development times for information systems

Key qualifications

In practical tutorials, students learn to work with important data and software standards used in industry, for example, XML, XPath, XSLT, DOM, SAX, SOAP. Through their individual computer exercises with these technologies, they also acquire the necessary practical experience for the independent familiarization with many very similar database and Internet technologies.

Activities expected of the students: co-operation during tutorial classes, homework, development of own software on the computer.

Module assignment

Elective module in the area of software technology and information systems

Mode

- Credit points for the complete module (workload): 8
- Credit points for each class: 4
- SWS (hours per week): 2 lectures + 1 tutorial per week for each class
- Frequency of the module offered: annually
- Duration: 1-2 semesters (depending on the catalog class selected)

Methods of implementation

- We explain core methods and technologies using typical examples
- They are then applied in practice during the tutorials,
- Partially students carry out small projects in supervised, small groups (Labs)

Organisational arrangements / media use / literature references

- Classes with problem oriented learning, i.e. lectures interchanged with exercises and discussions
- Tutorial classes in small groups
- Supervised labs
- Web-based lecture material (slides and actual research papers) and text book (if available)

Examination modalities

For DBIS 2 and Prolog, oral examination or labs if it is a small group of participants, otherwise, written examination. For the other classes, the examinations depend on the class selected.

Person responsible for the module

Böttcher

III.1.4 Knowledge-Based Systems

Role in the computer science Masters degree course

The module Knowledge-Based Systems includes classes from intelligent data processing for solving knowledge-intensive tasks. Different knowledge representations and corresponding inference algorithms will be discussed. The classes should enable the students to model problems that are difficult to structure, and to make them accessible for efficient solution methods.

Knowledge-based methods are not a "standalone technology". Rather, they intend to achieve a new quality in problem solving in combination with classic areas of computer science, or with engineering or business administration applications.

Content structure

So far, the following classes are planned for this module:

- Distributed Problem Solving
- Machine Learning
- Propositional Proof Systems
- Heuristic Search
- Prolog with Applications
- Theorem Proving

These classes are structured so that they form self-contained units and can largely be selected independently of each other.

Usability of the content

Students will apply the knowledge and skills acquired in this module in their professional practice where no standard problem solving methods exist, where they must account for aspects of uncertainty and vagueness, and where they need to emulate human problem solving behaviour, etc.

Prerequisites and prior knowledge

Students should have an interest in algorithms, abstract modeling, and a sound knowledge and practical experience in a programming language.

Learning goals

Students should have a good command of a selection of problem solving techniques, be able to analyse complex problems independently and to estimate the degree of possible automation realistically, and to develop an adequate solution based on this analysis.

Teaching of factual knowledge

After completing this module, the student will

- know a selection of task settings in the research area of knowledge-based systems,
- know and understand different representation formalisms and modeling techniques,
- know adequate inference algorithms and understand the differences and (dis)advantages of these algorithms.

Teaching of methodological knowledge

After completing this module, the student will be able to

- model systems with different representation formalisms, to apply suitable inference techniques, and to evaluate the results.

Teaching of transfer skills

After completing this module, the student will be able to

- recognise the applicability of knowledge-based techniques in new task settings and to
- acquire additional concepts and techniques independently.

Teaching of normative evaluation skills

After completing this module, the student will be able to

- assess the relevance of knowledge-based techniques in new task settings and to
- decide on the usability of problem solving techniques.

Key qualifications

For this module, we expect for you to

- co-operate and work in teams during the tutorial classes
- apply strategies for acquiring knowledge: combine lectures, prepare and review the lecture material, participate in tutorial classes with supervised group work, prepare homework, and take part in class tutorials.

Module assignment

Elective module in the area Software Technology and Information Systems

Mode

Credit points: 4+4 ECTS (2 catalog classes)

SWS (hours per week): 2+1, 2+1

Frequency: 2-3 catalog classes per year in the winter and summer semesters

Methods of implementation

- We will introduce and explain methods, techniques, and their implementation in the lectures using typical examples; students will then practice them during the tutorials.
- During some tutorials, students will prepare and evaluate prototype implementations.

Organisational arrangements / media use / literature references

- Lectures with overhead presentation
- Tutorial classes in small groups
- Homework assignments, solutions partially presented in tutorials
- Activities expected of the students:
participation in tutorial, preparation of homeworks, preparation and reviewing of lecture material
- Lecture slides, homework assignments, and links to additional literature on the Web

Examination modalities

Oral examination; written examination is possible in the case of high participant numbers

Person responsible for the module

Kleine Büning

III.1.5 Analytical Methods in Software Engineering

Role in the degree course

This module consolidates the knowledge in the foundations of software technology within the area of analytical methods. In contrast to constructive methods of software design, which aim at guaranteeing high quality by construction, analytical methods analyse software designs or code after construction. This module in particular looks at mathematical and formal methods in software design, trying to ensure correctness of software.

We will discuss in detail concepts and methods of semantics of programming languages, of semiautomatic and automatic verification techniques as well as classical software quality assurance. A command of these methods facilitates both a better understanding of the concepts of software technology, and scientific investigation, improvement, and the foundation of new software technologies.

Depending on the selected focus within this module, the students should, after successfully completing this module, be able to

- evaluate, compare and apply different analytic quality assurance techniques, and choose between different techniques based on the application domain,

- apply semiautomatic and automatic verification techniques.

This module is a mandatory elective module in Software Technology and Information Systems

Content structure of the module

To complete this module successfully, students select two classes from the following list:

- Model Checking
- Deductive Verification
- Propositional Proof Systems
- Theorem Proving
- Compilation Methods
- Software Quality Assurance

These classes are structured as follows:

- Propositional Proof systems
See Module Knowledge-based Systems
- Theorem Proving
See Module Knowledge-based Systems
- Modelchecking
 1. Modelling and property specification for reactive systems
 2. Temporal logics: LTL and CTL
 3. Fairness
 4. LTL Modelchecking via Büchi automata
 5. BDDs, Symbolic Modelchecking
 6. Reduction and abstraction techniques, Bisimulation
- Deductive verification
 1. A programming language and its semantics
 2. Proof systems
 3. Partial/total correctness
 4. Safety and liveness
 5. Soundness and completeness of proof systems
- Compilation Methods
See Module Languages and Programming Methods
- Software Quality Assurance
See Module Model-based Software Design

Usability of the content

This module teaches knowledge and skills that permit students to understand, formulate, formalize, and discuss complex relationships by using formal and mathematical models. Students will use these models specifically when developing safety critical systems and using reliable software.

Beyond these uses, the module offers a starting point for scientific work in formal methods, in particular on verification and model checking.

Prerequisites and prior knowledge

- The ability to model and formalise facts using mathematical and computer science notations as it is taught in the Modeling module is a prerequisite for the successful completion of this module.
- The module also requires command of at least one programming language as it is taught in Foundations of Program Development.
- In addition, the students should have a command of the basic techniques for formal definitions and conclusions, as they are taught in Logic and Semantics, Mathematics, and partly also in Principles of Knowledge-Based Systems.

Learning goals

Teaching of factual knowledge

After completing this module, the student will

- know the techniques and mathematical structures for formalizing the semantics of programming and modeling languages,
- know and understand different analytical techniques and methods for quality assurance, starting from static analysis over testing up to verification
- know the differences and (dis)advantages of the different techniques

Teaching of methodological knowledge

After completing this module, the student will be able to

- formally model systems and formulate their features
- evaluate the suitability of techniques and methods for different purposes
- apply mathematics and logic correctly and appropriately
- analyze software systems with respect to quality aspects

Teaching of transfer skills

After completing this module, the student will be able to

- set up mathematical models independently and discuss their features
- acquire new concepts and techniques and assess and, if necessary, adapt them

Teaching of normative evaluation skills

After completing this module, the student will be able to

- recognise the relevance of the semantic foundation of techniques
- be aware that selecting suitable verification methods requires a detailed analysis of the features of the specific application area.

Key qualifications

For this module, we expect for you to

- co-operate and work in teams during the tutorial classes
- apply strategies for acquiring knowledge
- combine lectures, prepare and review the lecture material, participate in tutorial classes with supervised group work, prepare homework, and take part in class tutorials.
- evaluate and question new concepts
- discover and establish crosslinks and relations between similar concepts

Module assignment

Mandatory elective module in the field of Software Technology and Information Systems

Mode

Credit points: 4+4 ECTS (4 for each class)

SWS (hours per week): 2 lectures + 1 tutorial, 2 lectures + 1 tutorial

Frequency: At least one course per term

Methods of implementation

- We introduce and discuss methods and techniques using typical examples
- Students try out these examples in practice during the tutorials; at times, computer tools are used.

Organisational arrangements / media use / literature references

- Lecture with overhead presentation or writing on the blackboard
- Materials complementing the lecture on the Internet
- In the tutorials, the problems are solved collectively

Examination modalities

The classes of this module are examined individually (orally or in writing, depending on the number of participants).

Person responsible for the module

Wehrheim

III.1.6 Constructive Methods in Software Engineering

Role in the computer science degree course

This module consolidates the knowledge in the foundations of software technology within the area of constructive methods. Constructive methods aim to ensure a high software quality directly through the engineering process (in contrast to analytical methods which ensure quality with an analysis after construction).

After successfully completing this module the students should be able to evaluate, compare and apply different constructive quality assurance concepts. In particular they should be able to choose and apply those methods that are appropriate for a given application domain.

Content structure of the module

For this module, students will select two advanced classes from the following list:

- Web Engineering
- Generating Software from Specifications
- Prolog mit Anwendungen
- Skriptsprachen
- Compilation Methods
- Parallel Programming in Java
- Objektorientierte Programmierung
- Funktionale Programmierung
- Software Safety
- Software Quality Assurance
- Databases and Information Systems 2 (DBIS 2)

The classes are structured as follows:

- Web Engineering:
See Module Model-based Software Design
- Generating Software from Specifications:
See Module Languages and Programming Methods
- Prolog mit Anwendungen (in German)
See Module Databases and Information Systems
- Skriptsprachen (in German)
See Module Languages and Programming Methods
- Compilation Methods
See Module Languages and Compilation Methods
- Parallel Programming in Java:
See Module Languages and Programming Methods
- Funktionale Programmierung (in German)
See Module Languages and Programming Methods
- Objektorientierte Programmierung (in German):
See Module Languages and Programming Methods

- Software Safety :
 1. Properties of safety-critical systems
 2. Model-based methods and domain-specific architectures for safety-critical systems
 3. Hazard analysis and fault tolerance
 4. Designing reliable software
- Software Quality Assurance
See Module Model-based Software Design
- Databases and Information Systems 2 (DBIS 2)
See Module Databases and Information Systems

Usability of the content

The contents of this module can be used in practice for design and implementation of complex software systems. Of particular importance is the knowledge of different design paradigms and the ability to choose a suitable method depending on a given domain and the system to be developed. Special attention will be directed to software-intensive and safety-critical systems.

Prerequisites and prior knowledge

- basic knowledge in software design, as taught in the courses “Software design” and “model-based software design” and basic knowledge about programming languages plus abilities in programming as can be learned in the courses “Programming 1 and 2” and “Foundations Programming Languages”.

Learning goals

Teaching of factual knowledge

After completing this module, the student will

- know well-established software engineering paradigms.
- understand the applicability of these paradigms to different contexts.

Teaching of methodological knowledge

After completing this module, the student will be able to

- choose and apply suitable methods for the design and maintenance of software systems.

Teaching of transfer skills

After completing this module, the student will be able to

- design complex software systems with regard to domain-specific requirements.
- adapt and apply new software engineering methods.

Teaching of normative evaluation skills

After completing this module, the student will be able to

- estimate the impact of design decisions on software systems.
- assess the applicability of design concepts for a given systems.
- grasp the importance of design decisions in the domain of safety-critical systems.

Key qualifications

For this module, we expect for you to

- co-operate and work in teams during the tutorial classes
- apply strategies for acquiring knowledge
- combine lectures, prepare and review the lecture material, participate in tutorial classes with supervised group work, prepare homework, and take part in class tutorials.

Module assignment

Elective module in the area of Software Technology and Information Systems

Mode

Credit points: 4+4 ECTS (2 catalog classes)

SWS (hours per week): 2+1, 2+1

Frequency: 2-3 catalog classes per year in the winter and summer semesters

Methods of implementation

- We introduce and explain methods and technologies using typical examples
- They are then applied in practice during the tutorials (all classes)
- In the tutorials, students carry out projects in supervised, small groups (GSS)

Organisational arrangements / media use / literature references

- Lectures with overhead presentation
- Tutorial classes in small groups
- Some supervised laboratory work
- Web-based lecture material

Examination modalities

Oral examination if fewer than 40 participants: otherwise, written examination in every catalog class

Person responsible for the module

Schäfer

III.2 Field: Models and Algorithms

III.2.1 Algorithms I

Role of the module in the degree course

Algorithms form the foundation of every hardware and software: A circuit converts an algorithm into hardware, a program makes an algorithm "understandable to the computer". Algorithms thus play a central role in computer science. An important goal of algorithm design is (resource) efficiency, for example, developing algorithms that solve a given problem as quickly as possible or with as little memory requirement as possible. In this module we discuss different methodical and application-specific algorithmic problems such as online algorithms, approximation, and randomisation, and present their applications in algorithms for graphs, coding problems, and geometric problems.

Because of the breadth and significance of the field and its important role in the University of Paderborn Computer Science Department, we offer students the opportunity to study this field in two modules, offering these two modules with an identical class listing: Algorithms I and Algorithms II.

Content structure of the module

The following classes are offered:

- Approximation Algorithms
- Randomised Algorithms
- Online Algorithms
- Algorithms in Computer Graphics
- Algorithmic Game Theory
- Algorithmic Coding Theory I
- Algorithmic Coding Theory II
- Network Flow Algorithms
- Graph Algorithms
- Geometric Algorithms
- Combinatorial Optimization

Usability of the content

The ability to design not just any algorithms, but to design resource-conserving (that is, efficient) algorithms for specific problems and the ability to assess problems with regard to their inherent complexity is important for many sub-fields of computer science. Databases and information systems, computer graphics systems, and scientific computation are important examples.

Prerequisites and prior knowledge

Prerequisites are the basic concepts from algorithm and complexity theory as they are taught in Introduction to Computability, Complexity and Formal Languages, and Efficient Algorithms.

Apart from basic mathematical knowledge as it is taught during undergraduate study, students must have an interest in creative problem solving with mathematically exact methods.

Learning goals of the module

Students will acquire the significant problems, concepts, and methods of algorithm development and analysis.

At the end of the class, the students should be able to assess problems with regard to their complexity and should be able to apply the correct algorithmic techniques for their solution.

Module assignment

Elective module in the field of Models and Algorithms.

Mode

- Credit points: 4+4
- SWS 2+1,2+1
- Frequency of the class offered: The module is offered annually.

Examination modalities

Oral or written examination, depending on the number of participants

Person responsible for the module

Meyer auf der Heide

III.2.2 Algorithms II

Identical to "Algorithms I".

III.2.3 Complexity and Cryptography

Role of the module in the degree course

At the core, this module deals with the question about the limitations of computability and the classification of problems with regard to their algorithmic complexity. We will use running time and memory requirements in particular as measures of complexity, but also for example, parallelisability. The module includes the proof of both undecidability, for example, of arithmetics, and the investigation of the problem-inherent complexity, that is, the proof of lower complexity bounds and the complexity comparison of problems. We also examine formal languages as aspects of complexity theory.

The basics of algorithms and complexity are complemented by methods for the algorithmic treatment of very complex problems, for example, approximation algorithms.

Content structure of the module

The following classes are offered:

- Complexity Theory II

- Concrete Complexity Theory
- Algebraic Complexity
- Approximation Algorithms
- Cryptographic Protocols
- Provable Security
- Lattices in Computer Science
- Models of Computation

Usability of the content

This module enables the students to assess the fundamental limitations of algorithmic solvability of problems and those imposed by resource limits. It also teaches them how to apply this skill to actual problems.

These skills are of advantage in all areas in models and algorithms as well as wherever algorithms for complex problems are developed.

Prerequisites and prior knowledge

Prerequisites are the basic concepts from complexity theory, as they are taught in Introduction to Computability, Complexity, Formal Languages and Complexity Theory.

Apart from basic mathematical knowledge as it is taught during undergraduate study, an interest in creative problem solving with mathematically exact methods is required.

Learning goals of the module

Students will acquire the significant problems, concepts, and methods of computability and complexity theory. At the end of the class, students should be able to assess problems with regard to their complexity and to apply the correct algorithmic techniques for their solution.

Module assignment

Elective module in Models and Algorithms.

Mode

- Credit points: 4+4
- SWS 2+1,2+1
- Frequency of the class offered: This module is offered in year ?.

Examination modalities

Oral or written examination, depending on the number of participants

Person responsible for the module

Blömer

III.2.4 Algorithms in Computer Networks

Role of the module in the degree course

In recent years, the theory of parallel algorithms and architectures has enabled massively parallel computers with a thousand processors or more to be built and to be used efficiently. The big challenges for computer science, such as weather forecasting, ocean simulation, astro-physical simulations, and drug design but also difficult optimisation problems require using massively parallel supercomputers. In addition to supercomputers, using parallel computers in the form of multi-processor PCs or processor clusters is already standard in many scientific, commercial, or industrial applications.

The Internet, which overall is also a parallel computer, is already used as such, for example, when grid computing applications are implemented.

Theoretical computer science has contributed significantly to the efficient use of parallel computers in many areas requiring high computing performance. This includes the model development for parallel computers and the development of efficient algorithms for these models. The basic aim of the module's classes is to achieve a general understanding for parallel processes by discussing analyzable parallel algorithms and architectures.

Content structure of the module

The module includes both classes that present efficient algorithms for problem solving with computer networks, and classes that present problem solutions that permit an efficient use of computer networks.

The module consists of the following classes:

- Algorithms for Synchronous Computer Networks
- Noncooperative Networks
- Resource Management in Computer Networks
- Algorithmic Foundations of the Internet
- Peer-to-Peer Networks
- Algorithmic Problems in Wireless Networks
- Communication in Networks

Usability of the content

The basic knowledge about parallel algorithms and architectures students acquire in this module is essential for anyone working with parallel computers in research, commerce, or industry. The knowledge acquired has a promising future because of the continued heavy growth of application fields for parallel supercomputers, in particular in scientific computation, but also because of the increasing number of multi-processor PCs or PC clusters in a commercial or industrial setting.

In addition to these types of parallel computers, the Internet is also increasingly used as a parallel computer, because of the rising number of services offered. The knowledge gained in this module will also be of importance for this growth industry.

Prerequisites and prior knowledge

Students must know the basic concepts in the fields of algorithms, data structures, computability, and complexity theory, as they are taught in the first four semesters. Knowledge of algorithms and their analysis, as it is taught in Fundamental Algorithms and Methods of Algorithm Design is of advantage.

Learning goals of the module

The class aims to introduce students to important parallel algorithmic techniques and architectures. One aim is to provide a base repertoire of parallel algorithms for problems that occur ever so often in applications. The other aim is to enable students to develop efficient parallel algorithms for new problem scenarios, and to implement them on actual parallel computers-- regardless of their type-- from supercomputers to the Internet.

Module assignment

Elective module in Models and Algorithms.

Mode

- Credit points: 4+4
- SWS 2+1,2+1
- Frequency of the module offered: One basic class every winter semester, the advanced classes every summer semester.

Examination modalities

Oral or written examination, depending on the number of participants

Person responsible for the module

Scheideler

III.3 Field: Embedded Systems and System Software

III.3.1 Distributed Computer Systems

Role of the module in the degree course

In a global economy based on a division of labor, distributed computer systems form part of the essential infrastructure in computing. Their secure and fast functioning is critical for business success. Computer networks and operating and distributed systems form the basic concepts of modern information systems. Distributed computer systems are based on the basic concepts of computer networks, operating systems, and distributed systems. The operating systems link the computer hardware with the software and provide an interface to the hardware resources. Computer networks transport data between separate, physically different devices. To this end, various communication channels are used (wired, fiber optics, wireless), devices of vastly different performance characteristics are connected, and various degrees of quality assurance are given (correct, dependable, or efficient communication). Based on existing computer networks, distributed systems permit interactions beyond computer limits. This interaction allows, for example, connecting different and spatially separated departments in an enterprise or for implementing general web services. Systems for distributed processing are also used where processes are to be accelerated or where fault tolerance is to be provided. In all cases, however, it is necessary that the implementation is carried out, for the user, as transparently, reliably, and securely as possible. Security aspects play a particularly important role as the processing is carried out via insecure network structures.

In this module, students will first develop basic, general principles that they need to realize such distributed computing systems. They then transfer the general principles to actual system software, computer resources, and programming models, and demonstrate them by case studies.

We have designed this module for students who wish to specialize in the software-oriented part of Embedded Systems and System Software, but do not intend to enroll in further ESS modules. This module builds on the foundations presented in “Concepts and Methods of System Software” and “Embedded Systems and System Software“. Students must take either Introduction to Distributed Systems or Computer Networks from the second stage of the BSc degree course as a prerequisite.

Content structure of the module

The module includes operating systems, computer networks, system aspects and algorithms of distributed systems, security in computer systems, mobile communication, high-performance networks, and various aspects of cluster computing and performance-optimised programming. In addition, we will cover performance evaluation of discrete systems in general. Operating Systems explains the structure and basic components of modern operating systems and covers algorithms and strategies for efficient resource management. We present implementing important mechanisms using the example of current operating systems. A series of classes in Computer Networks teaches students the methods of computer communication, advanced concepts of computer networking and network technologies, modern types of mobile communication, integrated voice and data

networks, and exemplary added value services such as video conferences. There is also a “systems” aspect present in these classes; for example, students will cover both cellular systems for mobile communication and local, wireless LAN-type systems, as well as new system concepts like wireless sensor networks. Distributed Systems provides knowledge about basic distributed algorithms and supporting aspects from computer communication, operating systems, security, and distributed data management. Security in Computer Systems looks at threats to which information processing systems are exposed. It also looks at corresponding mechanisms to defend against such threats. This approach includes topics such as confidentiality, integrity, availability, authorisation, access and usage control, security models and security architectures, etc. Cluster Computing, Architecture of Parallel Computer Systems, and Performance-Optimised Programming introduce students to the world of high-performance computing. We present both principal structure and case studies of high-performance computers here. We also discuss corresponding system software, network structures, and in particular programming models for parallel computers. In methodically oriented classes, various approaches for performance evaluation of systems in general are discussed, for example, analysis and simulation, question of experimental design, or performance optimization.

Usability of the content

Students will apply what they learn in these classes to application development, system administration, and in designing and implementing special systems. They will apply the mechanisms presented for resource management, security, and cross-platform communication both in classic information systems and -- in an adapted versions -- in special hardware resources. Knowledge about the detailed functioning of computer networks also helps computer scientists satisfy the complex requirements of modern information systems and access new fields of application. Time-dependent processes often play an important role in commercial and technical systems. Finally, students gain useful basic knowledge for network administration.

The basic building blocks for developing distributed systems are required for Internet applications, web services, enterprise software, etc. The knowledge gained should enable students to assess, select, and adapt various solution paths and components to a specific task. Last but not least, the knowledge about high-performance computing is also required in many related sciences in which complex and computing-intensive tasks are to be solved.

Prerequisites and prior knowledge

Introduction to Distributed Systems and/or Introduction to Computer Networks taught in the second stage of the BSc degree course in computer science are prerequisites. Students should also know the content of Concepts and Methods of Systems Software. Classes such as performance-oriented programming expect students to be willing to become familiar with system-level programming languages. Students must have basic knowledge of the programming languages from SWE.

Examinable standards / learning goals of the module

The students should gain an understanding of the specific features of system software and computer networks and become familiar with the basic building blocks for developing

operating and distributed systems. Students will recognise potential threats to the computer operation arising from unauthorised access to resources, and should be able to take the corresponding countermeasures. They will learn to assess and evaluate possibilities, limitations, and risks of open distributed systems and high-performance computers. Finally, students will comprehend the core methods for efficient processing and resource management and should apply them to concrete examples.

Teaching of factual knowledge – content competency

After completing this module, students will learn the

- Relationship between hardware and system software
- Structure, management and synchronisation of processes
- Techniques for memory management and scheduling
- Techniques for securing critical areas
- Techniques for designing parallel and concurrent programs
- Techniques for efficient, problem- and requirements-adequate transmission of data in wired, wireless, or mobile communication systems

Teaching of methodological knowledge – methodological skills

After completing this module, students will know the

- Methods for efficiently managing and allocating resources
- Methods for detecting and avoiding deadlocks
- Methods for co-operation between processes in distributed systems
- Process interaction methods
- Methods and approaches for performance analysis and optimization in communication systems and similar technical systems

Teaching of transfer skills

After completing this module, students will know how to transfer global strategies to specified individual situations, for example as part of exercises

Teaching of normative evaluation skills

After completing this module, students will

- Develop techniques for applying different strategies
- Recognise the practical value of the concepts and methods of system software
- Select a solution strategy adequate to a given task, its optimization goals, and its constraints

Key qualifications

The tutorials in small groups foster the ability to co-operate and work in teams.

Activities expected of students: co-operation during class-based tutorials, homework, independent study of secondary literature.

Module assignment

Elective module in Embedded Systems and System Software

Mode

- Credit points for each module (workload) : 8
- Credit points of the class: 4 each
- SWS (2 lectures + 1 tutorial, 2 lectures + 1 tutorial per week)
- Frequency of the module offered: 2-4 classes per year during the winter and summer semesters
- Duration (2 semesters)

Methods of implementation

In addition to classical lectures and exercises, tutorials in small groups foster the practical application of the methods presented to selected examples. Students must adapt particular, parameters and strategies to a given situation, problem, or case study. The tasks are worked on in groups of three students, which fosters the ability to work as part of a team. The structure of the exercise sheets maps the structure of the system software, starting with the hardware via processes to resource management and scheduling. This approach is reinforced by offering “project groups,” where real research problems are to be solved by students in realistically sized groups working over a realistic period of time. Seminars offer the option to delve deeply into a limited topic area.

Organisational arrangements / media use / literature references

- Lectures with overhead slides, writing on the blackboard in case of examples, additional explanations and topics to be elaborated on
- Weekly tutorials in small groups. These tutorials include demonstrating the calculations for tutorial exercises and the sample solutions for the homework
- Overhead script is available on the class homepage

Examination modalities

In general, there are individual examinations for each class with subsequent calculation of the average of the two individual results. The examinations for the individual classes in the module are carried out either as written examinations or as oral discussions, depending on the number of participants. Deviations from this scheme and additional requirements are specified when the respective class is announced.

Person responsible for the module

Karl

III.3.2 System Software

Role of the module in the degree course

Operating systems form the fundamental software layer that connects the computer hardware to the software. Together with other components of the system software, it enables developing applications and provides an interface to the hardware resources.

Distributed systems, however, permit interactions beyond computer limits. This characteristic allows for connecting different and spatially separated departments in an enterprise and for implementing general web services. Systems for distributed processing are also used where processes are to be accelerated or where failure safety is to be achieved. In all cases, however, it is necessary that user implementation is carried out as transparently, reliably, and securely as possible. Security aspects play a particularly important role as the processing is carried out via insecure network structures. Current developments lead to a convergence of operating systems and distributed systems such that many links become visible.

We have designed this module for students wishing to specialise in the SW-oriented part of Embedded Systems und System Software (ESS). The specific focus on operating and distributed systems permits combining the module with all other aspects of ESS, such as computer networks or embedded and real-time systems as part of the specialisation area. This module builds on the foundations presented in Concepts and Methods of System Software and Computer Engineering and requires Distributed Systems 1 from the second stage of the BSc degree course as a prerequisite. The general principles are now transferred to actual system software, computer resources, and programming models, and are demonstrated through case studies.

Content structure of the class

The module consists of Operating Systems, System Aspects and Algorithms of Distributed Systems, Security in Computer Systems, Cluster Computing, Architecture of Parallel Computer Systems, and Performance-Optimised Programming. Operating Systems explains the structure and basic components of modern operating systems and covers algorithms and strategies for efficient resource management. We present implementing important mechanisms using the example of current operating systems. Distributed Systems provides knowledge about basic distributed algorithms and supporting aspects from computer communication, operating systems, security, and distributed data management. Security in Computer Systems looks at threats to which information processing systems are exposed. It also looks at corresponding mechanisms to defend against such threats. This approach includes topics such as confidentiality, integrity, availability, authorisation, access and use control, security models and security architectures, encryption, key management, secure protocols, firewalls, and security in networks. Cluster Computing, Architecture of Parallel Computer Systems, and Performance-Optimised Programming introduce students into the world of high-performance computing. We present both principal structure and case studies of high-performance computers here. We also discuss corresponding system software, network structures, and in particular programming models for parallel computers.

Usability of the content

Students will apply what they learn in these classes to application development, system administration, and in designing and implementing special systems. Students will apply the mechanisms presented for resource management, security, and cross-platform communication both in classic information systems and -- in an adapted versions -- in special hardware resources. Students must also know about high-performance computing for many related sciences in which they must solve complex and computing-intensive

tasks. The basic building blocks for developing distributed systems are required for Internet applications, web services, enterprise software, etc. The knowledge gained should enable students to assess, select, and adapt various solution paths and components to a specific task.

Prerequisites and prior knowledge

Foundations of Distributed Systems taught in the second stage of the BSc degree course is a prerequisite. Further prerequisites are the contents of Computer Engineering and Concepts and Methods of Systems Software. Classes such as performance-oriented programming expect students to be willing to become familiar with system-level programming languages. Students must have a basic knowledge of the programming languages from SWE.

Examinable standards / learning goals of the module

Students will understand the specific features of system software and become familiar with the basic building blocks for developing operating and distributed systems. Students will recognise potential threats to the computer operation arising from unauthorised access to resources, and will be able to take the corresponding countermeasures. They will learn to assess and evaluate possibilities, limitations and risks of open distributed systems and high-performance computers. Finally, students will comprehend the core methods for efficient processing and resource management and will apply them to concrete examples.

Teaching of factual knowledge – content competency

After completing this module, students will know the

- Relationship between hardware and system software
- Structure, management and synchronisation of processes
- Techniques for memory management and scheduling
- Techniques for securing critical areas
- Techniques for designing parallel and concurrent programs

Teaching of methodological knowledge – methodological skills

After completing this module, students will have learned the

- Methods for efficiently managing and allocating resources
- Methods for detecting and avoiding deadlocks
- Methods for co-operating between processes in distributed systems
- Process interaction methods

Teaching of transfer skills

After completing this module, students will have learned how to transfer global strategies to specified individual situations, for example as part of exercises

Teaching of normative evaluation skills

After completing this module, students will be able to

- Develop techniques to apply different strategies
- Recognise the practical value of the concepts and methods of system software

Key qualifications

The tutorials in small groups foster the ability to co-operate and work in teams.

Activities expected of students: co-operation during tutorial classes, homework, independent study of secondary literature.

Module assignment

Elective module in Embedded Systems and System Software

Mode

- Credit points for each module (workload) : 8
- Credit points of the class: 4 each
- SWS (2 lectures + 1 tutorial, 2 lectures + 1 tutorial per week)
- Frequency of the class offered: 2-4 classes per year during the winter and summer semesters
- Duration (2 semesters)

Methods of implementation

Tutorials in small groups foster the practical application of the methods to selected examples. In particular, students must adapt parameters and strategies to the actual situation. The tasks are worked on in groups of three students, which fosters the ability to work as part of a team. The structure of the exercise sheets maps the structure of the system software, starting with the hardware via processes to resource management and scheduling.

Organisational arrangements / media use / literature references

- Lectures with overhead slides, writing on the blackboard in case of examples, additional explanations and topics to be elaborated on
- Weekly tutorials in small groups including students demonstrating the calculations for tutorial exercises and the sample solutions for the homework
- Overhead script is available on the class homepage

Examination modalities

In general, there are individual examinations for each class with subsequent calculation of the average of the two individual results. The examinations for the individual classes in the module are carried out either as written examinations or as oral discussions, depending on the number of participants. Deviations from this scheme and additional requirements are specified when the respective class is announced.

Person responsible for the module

Rammig

III.3.3 Computer Networks

Role of the module in the degree course

Transmitting data between separate devices using various communication media is a basic building block for almost all modern information systems. Such communication enables distributed systems; it makes mobile communication possible by using wireless transmission; it leads to varying requirements in different system architectures and is used in different forms – from highly reliable communication in small automation networks ranging via the Internet at large to small, self-organized, wirelessly communicating ad hoc networks. These topics are treated in Computer Networks.

We have designed Computer Networks for students wanting to specialise in Embedded Systems and System Software (ESS) and wanting to combine Computer Networks with one of the three other ESS core subjects: Operating and Distributed Systems, Embedded and Real-Time Systems, or HW/SW Codesign. This module builds on Embedded Systems and System Software taught in the second stage of the Bachelors degree course and has Computer Networks as a prerequisite.

Content structure of the module

The module consists of a varying selection of classes on advanced aspects of computer networks/Internet and computer communication, introduction to and advanced aspects of mobile communication, high-performance networks, and security in computer networks (the up-to-date selection is available on the module's website). We discuss problems and architecture of communication systems, clarifying the issues of construction and classification of computer networks, addressing, routing, etc. using concrete concepts and protocols for wired and wireless communication systems. In addition, we present case studies in high-speed networks, mobile networks, integrated voice and data networks, or mobile communication: wireless transmission and switching, video conferences, multi-media systems, multi-point communication, network security, and network management.

These technologically oriented classes are complemented by classes on performance evaluation and optimization of communication systems (the methods are applicable more generally to many technical systems). We discuss methods for analysis or simulation, experimental planning and parameter optimization are discussed.

Usability of the content

Knowledge about the detailed functioning of computer networks helps computer scientists to satisfy the complex requirements of modern information systems and to access new fields of application. In addition to a well-founded theoretical discussion of communication system, we also teach practical competence in usage, planning, configuration, programming, and administration of networks, which are relevant skills to many computer scientists' professions. The detailed modeling of relevant aspects and processes in a computer network is also a basic foundation for a simulation-based performance assessment – in particular when assessing systems or protocols that do not yet exist. Students will use the formal specification of communication systems for the (semi-)automated implementation of protocols with the help of respective programming tools and for testing the systems. The implementation results in a performance

assessment in the form of laboratory measurements. Finally, the knowledge gained serves as a first step toward system and network administration.

Prerequisites and prior knowledge

We expect students to have completed Introduction to Computer Networks taught in the second stage of the Bachelors degree course. A further prerequisite is the contents of Concepts and Methods of System Software. Students must have basic knowledge of the programming languages from SWE.

Examinable standards / learning goals of the module

Based on known foundations in computer science, students should get to know and comprehend basic concepts and different functionalities of computer networks and their use. Students specialising in this area should familiarise themselves with the core concepts and protocols of communication systems and understand the reasons why certain design decisions in these systems were made the way they were. Specialists must know the methods for modeling/formal specification of communication systems and for performance assessment via simulation/measurement. They must also be able to adapt these methods to a specific problem scenario.

Teaching of factual knowledge – content competency

After completing this module, students will have learned the

- Techniques for efficient, problem- and requirements-adequate transmission of data over various transmission media, especially in mobile and wireless communication systems
- Advanced and specialized methods and techniques of the Internet

Teaching of methodological knowledge – methodological skills

After completing this module, students will know the

- Methods of performance evaluation and optimization in communication systems and similar technical systems
- Specification of communication systems and their protocols
- Approaches for systematic protocol implementation

Teaching of transfer skills

After completing this module, students will be able to transfer global strategies to specified individual situations, for example as part of exercises

Teaching of normative evaluation skills

After completing this module, students will know how to

- Develop techniques to apply different strategies
- Select a solution strategy adequate to a given task, its optimization goals, and its constraints

Key qualifications

The tutorials in small groups foster the ability to co-operate and work in teams. Students will gain practical experience via lab classes and project groups; detailed knowledge in seminars.

Activities expected of students: co-operation during class-based tutorials, homework, independent study of secondary literature.

Module assignment

Elective module in the field of Embedded Systems and System Software

Mode

- Credit points for each module (workload) : 8
- Credit points of the class: 4 each
- SWS: 2 lectures + 1 tutorial per week, project groups, seminars
- Frequency of the module offered: 2-4 classes per year during the winter and summer semesters
- Duration (2 semesters)

Methods of implementation

Tutorials in small groups foster the practical application of the methods presented to selected examples. In particular, parameters and strategies must be adapted to the actual situation. The tasks are worked on in groups of three students, which fosters the ability to work as part of a team.

Organisational arrangements / media use / literature references

- Lectures with overhead slides, writing on the blackboard in case of examples, additional explanations, and topics to be elaborated on
- Weekly tutorials in small groups including students demonstrating the calculations for tutorial exercises and the sample solutions for the homework
- Overhead script is available on the class homepage

Examination modalities

In general, there are individual examinations for each class with subsequent calculation of the average of the two individual results. The examinations for the individual classes in the module are carried out either as written examinations or as oral discussions, depending on the number of participants. Deviations from this scheme and additional requirements are specified when the respective class is announced.

Person responsible for the module

Karl

III.3.4 Embedded Systems

Role of the module in the degree course

Embedded systems play a central role because of the continuous computerisation of all technical systems. Large parts of not only mechanical, automotive, and aerospace engineering, but also of communication technology can no longer be implemented without embedded systems. Embedded systems refer to the information processing components in such systems. They usually consist of dedicated hardware and the software that builds on it. Both are designed using the fundamental methods of computer science, with the interaction between HW and SW playing a particularly significant role. An important special feature of embedded systems, however, is that the physical laws of the entire system play a dominating role and must be taken into account during the design. Apart from real-time requirements, designers must also consider resource limits (for example, power consumption). This requirement results in a specific adaptation of all phases of the general design cycle of computer systems. During specifying and modeling, real-time constraints and resource limits must be describable, which leads to specific formalisms. The designer must validate and analyse the abstract models during interaction with the surrounding system components (that may in part also be modeled or that may actually exist). In embedded systems, the partitioning into hardware and software is carried out based on the constraints that the system must fulfill. Synthesis is also dominated by the requirement to respect these constraints. As embedded systems usually contain safety-relevant parts and sometimes even take care of the system's safety, particularly stringent verification techniques must be applied here. These are especially complex, not least because of the mandatory consideration of real-time aspects.

We have designed this module for students wanting to specialise in those aspects of Embedded Systems and System Software (ESS) that examine the interaction with the physical systems. The specific focus on embedded systems permits combining the module with all other aspects of ESS, such as computer networks or operating systems and distributed systems as part of the specialisation area. This module builds on the foundations presented in Concepts and Methods of System Software and Fundamentals of Computer Engineering. Students will transfer the general principles to real-time capable system software, mapped to hardware resources and application-specific programming models. They will demonstrate their knowledge through case studies.

Content structure of the module

The module consists of these classes: Embedded Systems, Real Time Operating Systems, Distributed and Parallel Embedded Systems, Intelligence in Embedded Systems, HW/SW Codesign, Sensor Technology, Design Methods for Embedded Systems, Advanced Concepts in Computer Architecture, and Advanced Concepts in HW/SW Codesign. Embedded Systems provides an overview of the task and basic solution approaches using a case-based approach. In embedded systems, the real-time operating systems play a central role. Therefore, this area is explored in depth in two classes that build on each other. First, we derive the special features of real-time operating systems from the concepts of general operating systems and then present the basic concepts that we then, second, refine and reinforce in a mathematically precise manner. Embedded systems are increasingly implemented in the form of distributed systems. We discuss the specific

aspects that students must consider during this development in Distributed and Parallel Embedded Systems. Intelligence in Embedded Systems establishes a link to the recognisable trend toward more autonomous, self-organising systems. HW/SW Codesign and Sensor Technology discuss problems that are of special importance and have special characteristics in embedded systems. The important aspect of design methods for embedded systems is covered in an additional class. Embedded systems are often based on special and advanced concepts in computer architecture, for example, in combination with reconfigurable hardware. We examine this relationship in Advanced Concepts in Computer Architecture. Advanced Concepts in HW/SW Codesign also refines and extends the basic principles of this technology.

Usability of the content

Students will apply what they learn in these classes to application development, technical systems, and in designing and implementing special systems. The methods presented for the specification, modeling, analysis, synthesis, and verification are required in all application areas of embedded systems, that is, in the entire field of technical systems. However, real-time applications are also applied in non-technical environments, for example, for weather forecasts or for the strategic planning of financial services. Beyond the application reference, the investigation of embedded systems also provides non-negligible insight because one is forced to abandon the fiction of idealism in Plato's sense and to deal with physical boundary conditions.

Prerequisites and prior knowledge

We expect students to have completed Embedded Systems or HW/SW Codesign taught in the second stage of the BSc degree course. Further prerequisites are the contents of the modules Computer Engineering and Concepts and Methods of Systems Software. In addition, students must have a basic knowledge of modeling principles from the module Modeling and of programming languages from SWE. Students should also be willing to become familiar with system-level programming languages. In some classes, in particular in HW/SW Codesign, we expect students to familiarise themselves with hardware description languages.

Examinable standards / learning goals of the module

Students should understand the specific features of embedded systems and become familiar with the basic concepts for the design of such systems. Students will also recognise potential dangers in the case of faulty design of embedded systems. They should also have a command of the instruments used to avoid such errors. Further, they should be able to assess the specific restrictions that result from the physical laws of the surrounding system and learn to include them systematically into the design process. Finally, students should comprehend the core methods for the precise, predictable use of scarce resources and should apply them to concrete examples.

Teaching of factual knowledge – content competency

After completing this module, students will have learned the

- Relationship between computer and physical system components

- Architecture variants for embedded systems
- Techniques of real-time management
- Techniques for validation and verification
- Methods for designing embedded systems

Teaching of methodological knowledge – methodological skills

After completing this module, students will know the

- Methods for predictable resource planning
- Methods for interacting with physical systems
- Methods for verifying time-dependent systems
- Methods for the targeted partitioning of tasks in HW and SW

Teaching of transfer skills

After completing this module, students will be able to transfer global strategies to specified individual situations, for example as part of exercises

Teaching of normative evaluation skills

After completing this module, students will

- Develop techniques to apply different strategies
- Recognise the practical value of the concepts and methods of embedded systems

Key qualifications

The tutorials in small groups foster the ability to co-operate and work in teams.

Activities expected of students: co-operation during tutorial classes, homework, independent study of secondary literature.

Module assignment

Elective module in Embedded Systems and System Software

Mode

- Credit points for each module (workload) : 8
- Credit points of the classes: 4 each
- SWS (2 lectures + 1 tutorial, 2 lectures + 1 tutorial per week)
- Frequency of the module offered: 2-4 classes per year during the winter and summer semesters
- Duration (2 semesters)

Methods of implementation

Tutorials in small groups foster the practical application of the presented methods to selected examples. In particular, students must adapt parameters and strategies to the actual situation. The tasks are worked on in groups of three students, which fosters the ability to work as part of a team. The structure of the exercise sheets maps the structure of

the system software, starting with the hardware via processes to resource management and scheduling.

Organisational arrangements / media use / literature references

- Lectures with overhead slides, writing on the blackboard in case of examples, additional explanations and topics to be elaborated on
- Weekly tutorials in small groups in which students will demonstrate the calculations for tutorial exercises and the sample solutions for the homework
- Overhead script is available on the class homepage

Examination modalities

In general, there are individual examinations for each class with subsequent calculation of the average of the two individual results. The examinations for the individual classes in the module are carried out either as written examinations or as oral discussions, depending on the number of participants. Deviations from this scheme and additional requirements are specified when the respective class is announced.

Person responsible for the module

Platzner

III.3.5 HW/SW Codesign

Role of the module in the degree course

Embedded systems play a central role because of the continuous computerisation of all technical systems. Embedded systems refer to the information processing components in such systems. They usually consist of dedicated hardware and software that builds on it. Both are designed using the fundamental methods of computer science, with the interaction between HW and SW playing a particularly significant role. An important special feature of embedded systems is, however, that the physical laws of the entire system play a dominating role and designers must account for them during the design. Apart from real-time requirements, designers must also consider resource limits (for example, relating to power consumption or available chip space) in this context. These requirements result in specific adaptation of all phases of designing computer systems. During specifying and modeling, real-time constraints and resource limits must be describable, which leads to specific formalisms. Designers must validate and analyse the abstract models in interaction with the surrounding system components (that may in part also be modeled or that may actually exist). In embedded systems, hardware and software is partitioned primarily with regard to the specific constraints rather than toward general optimisation. The process of hardware and software synthesis is also dominated by the specification of having to respect these restrictions. As embedded systems usually contain safety-relevant parts and sometimes even take care of the system's safety, designers must apply particularly stringent verification techniques here. These techniques are especially complex, not least because of the mandatory consideration of real-time aspects. However, because the systems under consideration are usually predefined and finite, designers may use methods from hardware verification as a basis for their work.

We designed this module for students wanting to specialise in those aspects of Embedded Systems and System Software (ESS) that deal not only with the interaction between hardware and software components, but also with the interaction with physical systems. The specific focus on HW/SW codesign permits combining the module with all other aspects of ESS, such as computer networks, operating systems and distributed systems, or embedded and real-time systems as part of the specialisation area. This module builds on the foundations presented in the modules Concepts and Methods of System Software and Computer Engineering. Students will transfer the general principles to the overall design of mixed HW/SW systems and demonstrate them in case studies. The special consideration paid to the physical laws of the surrounding non-computer system components poses specific challenges in this context.

Content structure of the module

The module consists of HW/SW Codesign, Sensor Technology, Design Methods for Embedded Systems, Advanced Concepts in Computer Architecture, and Advanced Concepts in HW/SW Codesign. In HW/SW Codesign, we discuss the design cycle of an integrated HW/SW design, starting with the specification and modeling, via analysis and validation, HW/SW partitioning, and HW/SW synthesis to system integration and verification. The partitioning plays a particularly significant role in this context. Furthermore, suitable interfaces have to be designed. In Sensor Technology, we discuss problems that are of special importance and have special characteristics in embedded systems. The task is to capture the state of the surrounding system as completely as possible and to make it available to be processed accordingly. This processing requires the designer to consider all aspects, ranging from acquiring elementary measurement data to complex filter algorithms. Sensor technology is naturally mirrored by its counterpart, actuator technology, which we also discuss. A further class covers the important aspect of design methods for embedded systems. In addition to the special problems in HW/SW Codesign, this class examines the entire design cycle. Embedded systems are often based on special and advanced concepts in computer architecture, for example, in combination with reconfigurable hardware. We examine this situation in Advanced Concepts in Computer Architecture. Advanced Concepts in HW/SW Codesign also refines and extends the basic principles of this technology. Reconfigurable hardware components play an increasingly important role here as well.

Usability of the content

Students will apply what they learn in these classes to developing applications and technical systems, and in designing and implementing special systems. The methods presented for the specification, modeling, analysis, HW/SW partitioning, synthesis, and verification are required in all application areas of embedded systems, that is, in the entire field of technical systems. Solutions in the traditional environment of information processing can also be optimised in a task-specific way by clever partitioning into HW and SW components. In general, an algorithm can not only be implemented in SW, but can also be implemented by a dedicated HW solution. This approach represents a non-negligible insight for students.

Prerequisites and prior knowledge

We expect students to have completed HW/SW Codesign taught in the second stage of the BSc degree course. Further prerequisites are the contents of the modules on Computer Engineering. In addition, students need a basic knowledge of modeling principles from Modeling and of programming languages in SWE. Students are also assumed to be willing to become familiar with system-level programming languages and hardware description languages.

Examinable standards / learning goals of the module

The students will understand the specific features of embedded systems and become familiar with the basic concepts for the design of such systems as mixed HW/SW implementations. Students will get to know partitioning criteria for HW/SW and will be able to carry out that partitioning. They will assess the specific restrictions that result from the physical laws of the surrounding system and learn to include them systematically into the design process. Finally, students will learn how to combine specific methods from both software technology and hardware design to achieve a powerful design methodology.

Teaching of factual knowledge – content competency

After completing this module, students will have learned the

- Relationship between computer and physical system components
- HW/SW architecture variants for embedded and real-time systems
- Techniques of HW/SW partitioning
- Techniques for validation and verification
- Techniques for the integrated design of mixed HW/SW systems

Teaching of methodological knowledge – methodological skills

After completing this module, students will know the methods for

- characterising algorithms with respect to the implementation technique
- the technical interaction with physical systems
- the verification of time-dependent HW/SW systems
- the targeted design of dedicated HW architectures

Teaching of transfer skills

After completing this module, students will be able to transfer global strategies to specified individual situations, for example as part of exercises

Teaching of normative evaluation skills

After completing this module, students will

- Develop techniques to apply different strategies
- Recognise the practical value of the concepts and methods of embedded systems

Key qualifications

The tutorials in small groups foster the ability to co-operate and work in teams.

Activities expected of students: co-operation during tutorial classes, homework, independent study of secondary literature.

Module assignment

Elective module in Embedded Systems and System Software

Mode

- Credit points for each module (workload) : 8
- Credit points of the classes: 4 each
- SWS (2 lectures + 1 tutorial, 2 lectures + 1 tutorial per week)
- Frequency of the module offered: 2-4 classes per year during the winter and summer semesters
- Duration (2 semesters)

Methods of implementation

Tutorials in small groups foster the practical application of the methods presented to selected examples. In particular, students must adapt parameters and strategies to the actual situation. The tasks are worked on in groups of three students, which fosters the ability to work as part of a team. The structure of the exercise sheets maps the structure of the system software, starting with the hardware via processes to resource management and scheduling.

Organisational arrangements / media use / literature references

- Lectures with overhead slides, writing on the blackboard in case of examples, additional explanations and topics to be elaborated on
- Weekly tutorials in small groups. This includes the demonstration of the calculations for tutorial exercises and the sample solutions for the homework
- Overhead script is available on the class homepage

Examination modalities

In general, there are individual examinations for each class with subsequent calculation of the average of the two individual results. The examinations for the individual classes in the module are either written examinations or oral discussions, depending on the number of participants. Deviations from this scheme and additional requirements are specified when the respective class is announced.

Person responsible for the module

Platzner

III.3.6 Embedded and Real-Time Systems

Role of the module in the degree course

Embedded systems play a central role because of the continuous computerisation of all technical systems. Large parts of not only mechanical, automotive, and aerospace engineering, but also communication technology can no longer be implemented without embedded systems. Embedded systems refer to the information processing components in such systems. They usually consist of dedicated hardware and software that builds on it. Both are designed using the basic methods of computer science. An important special feature of embedded systems, however, is that the physical laws of the entire system play a dominating role and designers must consider them during the design. This consideration is especially true for the real-time requirements. The real-time aspect can also play an important role in non-technical applications. This aspect results in a specific adaptation of all phases of the general design cycle of computer systems. During specifying and modeling, real-time constraints and resource limits must be describable, which leads to specific formalisms. Designers must validate and analyse the abstract models in interaction with the surrounding system components (that may in part also be modeled or that may actually exist). The synthesis is dominated by the requirement to respect these restrictions. As embedded systems usually contain safety-relevant parts and sometimes even take care of the system's safety, designers must apply particularly stringent verification techniques here. These techniques are especially complex, not least because of the mandatory consideration of real-time aspects.

We have designed this module for students wanting to specialise in those aspects of Embedded Systems and System Software (ESS) that deal with the interaction with physical systems. In addition, we intend to teach general issues of real-time processing. The specific focus on embedded and real-time systems permits combining the module with all other aspects of ESS, such as computer networks, operating systems and distributed systems, or HW/SW Codesign as part of the specialisation area. This module builds on the foundations presented in Concepts and Methods of System Software and Computer Engineering. The students will transfer the general principles to real-time capable system software and application-specific programming models, and demonstrate their knowledge with case studies. The special consideration paid to the physical laws of the surrounding non-computer system components poses specific challenges in this context.

Content structure of the module

The module encompasses Embedded Systems, Real Time Operating Systems, Distributed and Parallel Embedded Systems and Intelligence in Embedded Systems. Embedded Systems provides students an overview of the problem scenario and basic solution approaches with a case-based approach. In embedded systems, real-time operating systems play a particularly important role. For this reason, we present that topic in detail in classes that build on each other. First, we derive the special features of real-time operating systems from the concepts of general operating systems and, second, present the basic concepts. In this module the focus is on real-time scheduling and deterministic resource management. We then refine these concepts in a mathematically precise manner. In particular, we introduce students to analysis techniques that ensure a precise

predictability for the different scheduling methods. Embedded systems are increasingly implemented in the form of distributed systems. We discuss the specific aspects that designers must consider during this development in Distributed and Parallel Embedded Systems. The discussion topics here are real-time capable communication protocols and the distributed implementation of strictly deterministic algorithms. Intelligence in Embedded Systems ranges from the noticeable trend toward more autonomous, self-organising systems to evolving or organic systems. A further class covers the important aspect of design methods for embedded systems.

Usability of the content

Students will apply what they learn in these classes to developing applications and technical systems, and in designing and implementing special systems. Students must know the methods presented for the specification, modeling, analysis, synthesis, and verification in all application areas of embedded systems, that is, in the entire field of technical systems. However, real-time applications are also applied in a non-technical environment, for example, for weather forecasts or for the strategic planning of financial services. Beyond the application reference, investigating embedded and real-time systems also provides non-negligible insight because one is forced to abandon the fiction of idealism in Plato's sense and to deal with physical boundary conditions, particularly that of a temporal development predetermined by the environment.

Prerequisites and prior knowledge

Prerequisites for Embedded and Real-Time Systems are Embedded Systems or HW/SW Codesign taught in the Bachelors module Embedded Systems and System Software. A further prerequisite is the content of Computer Engineering and Concepts and Methods of Systems Software. In addition, students must possess basic knowledge of modeling principles from Modeling and of programming languages in SWE. We also expect students to be willing to become familiar with system-level programming languages.

Examinable standards / learning goals of the module

The students will acquire an understanding of the specific features of embedded systems and become familiar with the basic concepts for the design of such systems. The students will recognise potential dangers in the case of faulty design of embedded systems. They will also have a command of the instruments used to avoid such errors. They will be able to assess the specific restrictions that result from the physical laws of the surrounding system and learn to include them systematically into the design process. Finally, students will comprehend the core methods for ensuring a precise and predictable system behaviour and should apply these methods to concrete examples.

Teaching of factual knowledge – content competency

After completing this module, students will have learned the

- Relationship between computer and physical system components
- Implementation variants for embedded and real-time systems
- Techniques of real-time management
- Techniques for validation and verification

- Methods for designing embedded and real-time systems

Teaching of methodological knowledge – methodological skills

After completing this module, students will know the methods for

- predictable resource planning
- the logical interaction with physical systems
- verifying time-dependent systems
- designing systems with inherent intelligence

Teaching of transfer skills

After completing this module, students will transfer global strategies to specified individual situations, for example as part of exercises

Teaching of normative evaluation skills

After completing this module, students will be able to

- Develop techniques for applying different strategies
- Recognise the practical value of the concepts and methods of embedded systems

Key qualifications

The tutorials in small groups foster the ability to co-operate and work in teams.

Activities expected of students: co-operation during tutorial classes, homework, independent study of secondary literature.

Module assignment

Elective module in Embedded Systems and System Software

Mode

- Credit points for each module (workload) : 8
- Credit points of the classes: 4 each
- SWS (2 lectures + 1 tutorial, 2 lectures + 1 tutorial per week)
- Frequency of the module offered: 2-4 classes per year during the winter and summer semesters
- Duration (2 semesters)

Methods of implementation

Tutorials in small groups foster the practical application of the methods presented to selected examples. In particular, students must adapt parameters and strategies to the actual situation. The tasks are worked on in groups of three students, which fosters the ability to work as part of a team. The structure of the exercise sheets maps the structure of the system software, starting with the hardware via processes to resource management and scheduling.

Organisational arrangements / media use / literature references

- Lectures with overhead slides, writing on the blackboard in case of examples, additional explanations and topics to be elaborated on
- Weekly tutorials in small groups including demonstrating the calculations for tutorial exercises and the sample solutions for the homework
- Overhead script is available on the class homepage

Examination modalities

In general, there are individual examinations for each class with subsequent calculation of the average of the two individual results. The examinations for the individual classes in the module are either written examinations or oral discussions, depending on the number of participants. Deviations from this scheme and additional requirements are specified when the respective class is announced.

Person responsible for the module

Rammig

III.4 Field: Human-Machine Interaction

III.4.1 Graphics and Visual Computing

Role in the degree course

Graphics and Visual Computing focuses on generating images on the computer via scene descriptions, simulated, measured or empirical data, and the capture, analysis, interaction and exchange of image data. It belongs to the modules in Human-Machine Interaction (MMWW)

Content structure of the module:

The module consists of

- one basic class Computer Graphics II with the following content:
 - ray tracing
 - radiosity
 - volume rendering
 - advanced modeling
 - texture mapping
 - image-based rendering
 - non-photorealistic rendering
 - animation
- a range of additional classes, where students select one of the following:
 - Digital Image Processing
 - Computer-Generated Visualisation
 - AR/VR (Augmented Reality/Virtual Reality) (planned)
 - Seminar: Selected Topics in Computer Graphics
 - Seminar: Selected Topics in Digital Image Processing
 - Seminar: Selected Topics in Visualisation

Topics for additional classes are as follows::

- Computer-Generated Visualisation
 - basics, definitions
 - data and data models
 - observers and tasks
 - mapping (mapping of data onto images)

- representation
- interaction flow
- Digital Image Processing
 - characterisation of digital images (rastering, quantisation, etc.)
 - point operations and filter operations in the spatial domain
 - transformations
 - operations in the frequency domain
 - image compression
 - image segmentation
 - image restoration.
- The contents of the other classes are determined as required.

Usability of the content

The methods of photorealistic rendering are a recent and dynamic area of computer science. Computer Graphics II provides knowledge in state-of-the-art photorealistic rendering, and builds the necessary foundations in order for students to understand future developments in computer graphics. Because the quantity of data continuously increases (for example, medical data, data from space missions, statistical data, scientific computations, etc.) and because it is to be interpreted quickly and correctly by humans (for example, surgeons, geologists, environmentalists, social scientists, etc.), systematic strategies for converting data into expressive and effective images (or image series collections) are required. Computer Generated Visualisation concerns itself with these issues. In order to characterise the image data generated in this way with respect to quality and quantity and to use transformations and operations for image improvement, image manipulation, and image transfer, Digital Image Processing provides the necessary basics for understanding the respective algorithms.

Prerequisites and prior knowledge

Participation in Computer Graphics I.

Learning goals of the module

Teaching of factual knowledge

- see Content structure of the module (above)

Teaching of methodological knowledge

After completing this module, students will have learned

- the methodical foundations of the algorithms
- efficient algorithms vs. photorealistic algorithms
- the practical application of the methods on the computer

- strategic procedures for converting data into images with respect to human interpretation
- transformation into different image spaces
- compression algorithms
- practical implementation of the algorithms on the computer: a fundamental step in order to understand the problem of applying the theory to practice.

Teaching of transfer skills

Knowledge in computer graphics and image processing permits generating effective visualisations for application areas such as medicine, biology, chemistry, and many more. Image segmentation is a preprocessing step in robotics.

Teaching of normative evaluation skills

After completing this module, students will understand the

- efficiency assessment of computer graphics algorithms
- quality assessment of a graphics card
- image quality assessment for a specific target group and a specific visualisation goal
- assessment of quality loss in image compression

Key qualifications

- Ability to use modern information and communication technologies
- Subject-specific foreign language skills because accompanying literature is in English (for non-native speakers of English)
- Ability to co-operate and work in teams is fostered in group projects
- Activities expected of students: Willingness to reactivate mathematical knowledge from the past; independent programming; co-operation during tutorial classes

Module assignment:

Elective module in Human-Machine Interaction (MMWW)

Mode

- Credit points: 4 + 4 ECTS (2 catalog classes)
- SWS: 2+1, 2+1 (or 2+1, 2 if a seminar is selected)
- Frequency of the module offered: 2-3 catalog classes per year in the winter and summer semesters

Methods of implementation

Students will expand their knowledge of the theoretical concepts in small groups during tutorial classes and test the methods in practical tutorials.

Organisational arrangements / media use / literature references

- For classes: one two-hour lecture per week and one two-hour tutorial class every second week, or solving of programming tasks in one's own time to a similar extent
- Software used: currently OpenGL, IDL (by Research Systems, Inc.) for image processing
- For seminars: either during the semester or as a block seminar, as announced
- Materials used: PowerPoint overhead slides for downloading and exercise sheets
- Literature references for the class Computer Graphics: Angel, Interactive Computer Graphics, Addison-Wesley; or Watt, Three Dimensional Computer Graphics, Addison-Wesley; or Foley et al., Computer Graphics, Addison Wesley Publishing.
- Literature references for the class Digital Image Processing: Gonzalez /Woods, Digital Image Processing, Addison Wesley Publishing.
- Literature references for the class Computer-Generated Visualisation: Web-based class script

Examination modalities

- Written examination
- Independent programming of (parts of) the rendering pipeline or tasks adapted to the additional topics
- Information on grading policies: The weighting of written examination and programming assignments/projects is announced at the beginning of the semester.

Person responsible for the module

Domik

III.4.2 Computer Science and Society

Role in the degree course

Computer scientists develop products that are based on characters (programs, specifications, documentations, etc.). In contrast to other engineering products that are manufactured from materials such as steel, plastics, or glass, software maps social reality into a wide range of shapes. Through its use, this reality changes. This change leads to many different interactions between computer systems and their application environment with respect to understanding, usage potential, and application risks. Starting from these special aspects of computer science, the class compares automatic data processing (product) with human information processing (process) and discusses the resulting consequences for designing computer systems on all levels of development and usage. Topics discussed in depth are as follows:

- Cultural history of data processing
- Design as adaptation
- Computer science and military
- Socially oriented system design

- Multimedia and society

The module belongs to the modules in Human-Machine Interaction (MMWW)

Content structure of the module:

The module consists of

- a basic class Computer Science and Society with the following content:
 - features of software as an engineering product
 - human-machine interactions
 - biological information processing
 - faults in technical and natural systems
 - artifacts as external memory
 - non-self-aware and self-aware design processes
 - product-process complementarity
 - computer science and military
 - nuclear war by accident
 - software development as a learning process
 - responsibility of the computer scientist
- a range of additional classes selected from the following list:
 - Concepts of Digital Media
 - Seminar: Between Science and Fiction
 - Seminar: Cultural History of Data Processing
 - Seminar: Copyright and Digital Media

Topics for additional classes are as follows:

- Concepts of Digital Media
 - instrument, machine, automaton, interactive system
 - hypertext
 - groupware
 - virtual communities
 - virtual reality, augmented reality
 - ambient computing, mediatronics
- The seminar topics are determined as required.

Usability of the content

Students gain basic knowledge about the possibilities and limitations of designing computer systems and of formalisation. These insights are necessary both for assessing technical potentials and for managerial positions when handling software projects. The study of interactions also provides an advanced understanding of problems and potentials of IT in different application contexts. Considerations regarding the history of data processing integrate current concepts of computer science into a wider cultural and historic framework.

Prerequisites and prior knowledge

None.

Learning goals of the module

Teaching of factual knowledge

After completing this module, students will understand

- the history of data processing
- human-machine interactions
- biological information processing
- cultural and social design processes
- the relationship between computer science and the military
- process-oriented software design

Teaching of methodical skills

After completing this module, students will be able to

- consider different concepts of technical and biological information processing
- develop interdisciplinary avenues in the human-machine field
- discuss ethical problems
- apply the product-process complementarity to different problems

Teaching of transfer skills

After completing this module, students will be able to

- assess formal and informal methods
- recognise and solve design conflicts
- design error-tolerant development environments
- apply project management skills for the process design
- explore interdisciplinary literature

Teaching of normative evaluation skills

After completing this module, students will be able to assess

- the limitations of formalisation
- relevant legislative texts

Key qualifications

- Basic design and presentation skills
- Acquisition of concepts foreign to the subject
- Ability to co-operate and work in teams as fostered through group work
- Activities expected of students: Willingness to look beyond the familiar horizon of computer science and to acquire concepts and approaches from other disciplines

Module assignment:

Elective module in Human-Machine Interaction (MMWW)

Mode

- Credit points: 4 + 4 ECTS (2 catalog classes)
- SWS: 2+1, 2+1 (or 2+1, 2 if a seminar is selected)
- Frequency of the module offered: 2-3 catalog classes per year in the winter and summer semesters

Methods of implementation

Students will explore and work out topics in group work. The presentation is carried out as design of multi-media knowledge spaces, where presentation skills are learned over a longer period by working on a coherent problem. The research and exploration of literature from other disciplines plays an important role in this context.

Organisational arrangements / media use / literature references

- For classes: one two-hour lecture per week and one two-hour tutorial class every second week,
- Studying topics by researching interdisciplinary literature in one's own time with a similar workload
- For seminars: (either during the semester or as a block seminar) exploring a specific topic area for presentation during the class
- Materials used: PowerPoint overhead slides for download and exercise sheets, texts of legislation and third-party materials, and audio annotations to the lecture.

Examination modalities

- Written examination and oral examinations
- Information on grading policies: The weighting of written examination and tutorial performance is announced at the beginning of the semester.

Person responsible for the module

Keil

III.4.3 Concepts of Digital Media

Role in the degree course

The classic media theories consist mainly of mass media reception analyses with special emphasis on film and television. Mathematical formulas, technical drawings, or administrative forms are not considered in media theory. However, the computer erodes these boundaries. Digital media potentially connect all previously known media types, even if this does not happen with the same quality and under the same conditions of production and reception.

The concept of a character and its processing by (digital) automata yields an extended media concept. It permits discussion of the multitude of digital media under a common technical point of reference. In analogy to the concepts of programming languages, we may compare different expressions of digital media and determine the associated added value of the media. In today's world, this determination is essential for all application areas that carry the e-prefix (e-learning, e-government, e-business, etc.).

The basic understanding of the computer as a digital medium is of decisive importance for the future development of information technologies. At the same time, it allows the positioning of computer science and research and development areas relevant to computer science within newly developing interdisciplinary degree courses and research topics. It also provides connective knowledge for interdisciplinary work, in particular with communication sciences, pedagogy, and psychology, and creates an advanced understanding of one's own technical foundations.

Content structure of the module:

The module includes classes relating to both designing interactive systems and cooperative media, and to basic design and media concepts that computer scientists can use to discuss cognitive and media-related added value as well as deficits. The class, which also carries the title Concepts of Digital Media, provides the theoretical and conceptual foundations for this purpose. The other classes consolidate and differentiate these concepts with respect to different problem areas. The module consists of the following classes:

- Concepts of Digital Media (mandatory)
- Software Ergonomics
- Architectures of CSCW Systems
- Design of Interactive Systems

Usability of the content

Basic knowledge of the functionality of digital media with regard to cognitive and cultural-social processes is essential for the productive use of computers in networked workplace and living environments. The ability to distinguish technical problems and concepts from non-technical ones is an important prerequisite for analysing requirements and for developing transparent application systems. Advanced knowledge about special

architectures and about the fundamental comparability of functional, interactive and co-operative system concepts is a necessity for computer scientists in different application contexts.

Prerequisites and prior knowledge

The basic knowledge from the first four semesters is a prerequisite.

Learning goals of the module

Students should learn to distinguish between technical and non-technical problem scenarios and to relate them adequately to each other. They should also be able to determine requirements from a media-related application environment, using basic application-related but not application-specific concepts, to evaluate and compare possible system architectures, and to assess new innovation potentials in the media field. Teaching and applying cognition-scientific foundations should enable students to link technical and non-technical concepts constructively.

Module assignment

Human-Machine Interaction.

Mode

- Credit points: 4 + 4
- SWS 2 + 1, 2 + 1
- Frequency of the module offered: One basic class every winter semester, the advanced classes every summer semester.

Examination modalities

Oral or written examination, depending on the number of participants

Person responsible for the module

Keil

III.4.4 Computer-Supported Co-operative Collaboration and Learning

Role in the degree course

Co-operation support systems play an increasing role in large areas of human collaboration and learning. Accordingly, the research fields of Computer Supported Co-operative Work (CSCW) and Computer Supported Co-operative Learning (CSCL) encompass both tools and systems, but also theories and approaches of co-operative media use. The class enables computer scientists to assess the state-of-the-art of the CSCW/L research field, and to categorise and distinguish the foundations of classification, architectural development lines, and different types of support for human collaboration. The module belongs to the modules in Human-Machine Interaction (MMWW)

Content structure of the module:

The module consists of

- an introductory class CSCW with the following content:
 - conceptual foundations: CSCW/CSCL/groupware
 - media between production and communication
 - from interaction to co-operation: overview of the research fields CSCW/CSCL/groupware
 - foundations of co-operative media use, media functions
 - non-sequential writing: hypertext
 - reciprocal perception, media spaces, awareness
 - virtual communities MUD/MOO
 - introduction to different systems: BSCW, Notes, Netmeeting, Groove, etc.
 - virtual knowledge spaces/CSCL
 - technical-architectural foundations of co-operation support systems
 - basic architectural concepts of replicated and centralised systems
 - mobile computing, dynamic configuration
- a range of additional classes, one of which must be selected
 - additional classes from the module Human-Computer Interaction
 - Concepts of Digital Media
 - Architectures CSCW
 - Seminar: Mobile Ad-Hoc Networking of Co-Operative Knowledge Spaces
 - Seminar: Virtual Knowledge Spaces - CSCL

Topics for additional classes are as follows:

- Concepts of Digital Media (Reinhard Keil-Slawik)
 - instrument, machine, automaton, interactive system
 - hypertext
 - groupware
 - virtual communities
 - virtual reality, augmented reality
 - ambient computing, mediatronics
- see additional classes listed in Human-Computer Interaction
- Architectures of co-operation support systems - CSCW/CSCL
 - from interaction to co-operation: overview of the research fields CSCW/CSCL/groupware
 - historical development of different CSCW architecture concepts
 - from screen sharing to common information spaces
 - architectural design patterns of co-operation support systems
 - basic protocols and standards
 - replicated architectures
 - centralised architectures – co-operative MVC concept
 - architecture concepts MUDs and MOOs
 - communication between applications – different standards

- relevant architecture concepts of the web for the area of co-operation support systems
 - Semantic Web
 - web services
- basic technological concepts and metaphors of co-operation support systems
 - virtual spaces
 - groups and access rights
 - floor control
 - collaboration unaware/aware - collaboration transparent
 - concurrency check
 - versioning
 - persistence layer
- frameworks of co-operation support systems
 - centralised and replicated architecture approaches
 - object-oriented frameworks
 - frameworks of the CSCL
 - exemplary introduction of different innovative architecture concepts, for example, Fraunhofer DyCE, University of Paderborn sTeam
- mobility aspects of co-operation support systems
 - protocols and standards (peer-to-peer, IPV6)
 - protocols for ad-hoc networking
 - frameworks for ad-hoc networking
- The seminar topics are determined as required.

Usability of the content

The knowledge taught in computer-supported group work forms an important foundation for many application scenarios. These scenarios include, for example, groupware and workflow, which are motivated by office administration economy, and providing network-based group work environments. Further application fields include research group support in scientific work and the wide field of e-learning.

Prerequisites and prior knowledge

None.

Learning goals of the module

Teaching of factual knowledge

After completing this module, students will have learned about

- different groupware systems
- the architectural foundations of these systems (for example, group structures, user privileges)
- the sample architectures of synchronous and asynchronous applications
- the basics of media-supported human collaboration

Teaching of design competency

After completing this module, students will know

- how to develop groupware applications
- the architecture design of synchronous and asynchronous applications
- how to create different models of user privileges and group structures

Teaching of transfer skills

After completing this module, students will know how

- to compare individual and co-operative types of collaboration
- differentiate between different types and levels of support for human collaboration.
- to detect and resolve conflicts in the design of co-operation support systems.

Teaching of normative evaluation skills

After completing this module, students will be able to assess

- and evaluate different support types for human collaboration
- and evaluate different systems
- the usability of co-operation support systems

Key qualifications

- Basic assessment competencies in different approaches and systems
- Design competency in different architecture concepts
- Acquisition of concepts foreign to the subject
- Ability to co-operate and work in teams as fostered through group work

Module assignment:

Elective module in the area of Human-Machine Interaction (MMWW)

Mode

- Credit points: 4 + 4 ECTS (2 catalog classes)
- SWS: 2+1, 2+1 (or 2+1, 2 if a seminar is selected)
- Frequency of the module offered: 2 catalog classes per year in the winter and summer semesters

Methods of implementation

Students will apply and consolidate the design concepts in small groups and acquire and apply presentation skills during tutorial classes.

Organisational arrangements / media use / literature references

- For classes: one two-hour lecture and one tutorial class per week, or solving of design tasks in one's own time to a similar extent
- Software used: presentation systems and different development systems, introduction of different co-operation support systems,

- Use of the sTeam co-operation support system for work in small groups, provision of overhead slides, and teaching materials.
- For seminars (either during the semester or as a block seminar): exploring a specific topic area and presenting it during the class
- References regarding activities expected of students: co-operation during the tutorial classes, active willingness to develop and design independent solution approaches and architecture concepts, and willingness to analyse and assess different systems.
- Materials used: PowerPoint slides, sTeam co-operation support system.
- Literature references for Introduction to CSCW: Borghoff, U.M., Schlichter, J.H.: *Rechnergestützte Gruppenarbeit – Eine Einführung in verteilte Anwendungen*. Berlin: Springer 1995. Schwabe, G., Streitz, N., Unland, R.: *CSCW Kompendium – Lehr- und Handbuch zum computergestützten kooperativen Arbeiten*, Springer 2001, Teufel, S., Sauter, C., Mühlherr, T., Bauknecht, K.: *Computerunterstützung für die Gruppenarbeit*. Bonn: Addison-Wesley 1995. Greif, I.: *Computer Supported Co-operative Work: A Book of Readings*. San Mateo: Morgan Kaufmann Publishers, 1988, Hofte, G.H.: *Working Apart Together – Foundations for Component Groupware*, Telematica Instituut Fundamental Research Series, Vol. 1, Enschede, the Netherlands, 1998.

Examination modalities

- Oral examinations
- Information on grading policies: The weighting of oral examination and tutorial performance is announced at the beginning of the semester.

Person responsible for the module

Hampel

III.4.5 User Interface Development

Role in the degree course

Software and software use are continuously increasing. As a result, developing user interfaces is a major task for software companies. It represents a multilevel problem that considers diverse aspects such as software development, logical and graphical design, workflow integration, perception psychology problems, etc. User Interface Development belongs to the modules in the sub-area of Human-Machine Interaction (MMWW). This module provides significant concepts and methods for this task, for example, modeling concepts and techniques, usage paradigms, and design guidelines. Additional classes offer a specialisation in sub-areas such as programming techniques and tools, usability engineering, web-based user interfaces, and more.

Content structure of the module:

The module consists of

- a basic class User Interface Modeling with the following content:
 - basics of modeling
 - model-based development process
 - task modeling
 - user interaction modeling
 - control modeling
- Topics for additional classes are as follows (select at least one):
 - Design of Interactive Systems
 - Practical Usability Engineering
 - Programming Interactive Systems (in English)
 - Web Modeling (planned)
 - Programming Interactive Web Sites (planned, in English)
 - Seminar: "Current Topics in User Interfaces"

Usability of the content

The knowledge gained is required for the adequate structuring of the user interface. The modeling concepts taught in the basic class are particularly useful in this context. They provide basic abstraction concepts that permit the complex development process to be divided into steps that are to be carried out systematically. The additional classes provide the opportunity to expand the acquired knowledge in different directions: One possible direction, for example, is the implementation steps that follow the modeling, or applying the concepts and techniques to web-based user interfaces, or advanced techniques in usability engineering. The additional classes are updated to reflect the progress in the area.

Prerequisites and prior knowledge

Programming skills (currently, Java) are useful for understanding the entire module, but these are not explicitly expected. However, the programming-oriented additional classes (Programming Interactive Systems and Programming for the Web) require proof of successful participation in the software technology practical. Practice in Usability Engineering requires basic knowledge in usability engineering techniques. These may be obtained, for example, in the BSc class Usability Engineering.

Learning goals of the module

Teaching of factual knowledge

After completing this module, students will have learned

- the meaning of different modeling techniques in the development process
- the structure of a model-based development process
- task modeling
- user interaction modeling
- control modeling
- design guidelines from a psychological point of view
- basic facts and laws of perception psychology
- standards and style guides

- concepts and techniques for developing web-based user interfaces (additional class)
- concepts and techniques for the programming of user interfaces (additional class)

Teaching of methodological knowledge

After completing this module, students will be able to

- adequately structure the user interface development process
- design different aspects of user interfaces using suitable models without actual programming
- assess and create user interfaces on the basis of fundamental design principles
- use style guides in developing user interfaces
- check user interfaces with respect to standards
- assess the usability of web-based user interfaces (additional class)
- apply concepts of classic user interface development to web-based problems (additional class)
- apply concepts, techniques, and tools of user interface programming (additional class)
- plan and implement usability tests to a realistic extent

Teaching of transfer skills

Students can apply the modeling concepts taught as central contents in the module in other areas of computer science (any type of behaviour descriptions). To a large extent, they are also applicable in the advanced development of user interfaces, as abstractions are made, for example from graphics techniques or hardware properties. The extension to web-based systems taught in additional classes creates the foundations for acquiring new and current techniques, although this content needs to be updated regularly to reflect new developments. Because the human user is subject to a relatively small degree of change, the usability-oriented classes remain highly current and continue to be relevant for all problem areas that exist between human and machine.

Teaching of normative evaluation skills

After completing this module, students will have learned to

- check the adherence to standards, style guides, and design laws in classic user interfaces and in web sites
- assess and test the basic performance of tools for user interface development
- assess the difficulty of developing a user interface in relation to given development and runtime environments (additional class)
- assess the feasibility of interactive systems on the web (additional class)
- assess the quality and significance of usability tests (additional class)

Key qualifications

- Ability to use modern user interface technologies
- Subject-specific foreign language skills because accompanying literature is in English (for non-native speakers of English)

- Ability to co-operate and work in teams as fostered in group projects

Module assignment:

Elective module in the area of Human-Machine Interaction (MMWW)

Mode

- Credit points: 4 + 4 ECTS (2 catalog classes)
- SWS: 2+1, 2+1 (or 2, if a seminar is selected)
- Frequency of the module offered: 2-3 catalog classes per year in the winter and summer semesters

Methods of implementation

The theoretical concepts are expanded in small groups during tutorial classes. The methods are tested in practical tutorials on user interface development problems, web interfaces, or usability tests, depending on the additional class selected.

Organisational arrangements / media use / literature references

- For classes: one two-hour lecture and a one hour tutorial class per week (or one two-hour tutorial class every second week), and/or solving of programming tasks, or development of modeling documents or usability tests in one's own time and to a similar extent
- Typically, the seminar is a block seminar, but it may also take place regularly during the semester, as announced
- References regarding activities expected of students: independent programming; co-operation during tutorial classes; contact with human subjects during user tests, where applicable.
- Materials used: exercise sheets

Examination modalities

- Written examination
- Solution of practical problems via programming tasks and/or independent projects
- Information on grading policies: The weighting of written examination and programming assignments/projects is announced at the beginning of the semester.

Person responsible for the module

Szwillus

III.4.6 Human-Machine Interaction

Role in the degree course

The module provides an expanded and sometimes broader access for students interested in the topics of Human-Machine Interaction. The contents of the module are not connected to each other. Rather, students have a broad choice either to study some areas in more detail or to explore several directions in parallel.

Content structure of the module:

The module consists of

- a range of additional classes, two of which must be selected:
 1. Architectures CSCW
 2. Augmented Reality/Virtual Reality (planned)
 3. Computers and Disabled Persons - Access to Information Technology for People with Special Needs
 4. Computer-Generated Visualisation
 5. Digital Image Processing
 6. Design of Interactive Systems
 7. Concepts of Digital Media
 8. Practical Usability Engineering
 9. Programming Interactive Systems (in English)
 10. Programming Interactive Web Sites (planned, in English)
 11. Software Ergonomics
 12. Web Modeling (planned)

The individual classes are structured as follows:

1. Architectures CSCW
 - from interaction to co-operation: overview of the research fields CSCW/CSCL/groupware
 - historical development of different CSCW architecture concepts
 - architectural design patterns of co-operation support systems
 - relevant architecture concepts of the web for co-operation support systems
 - basic technological concepts and metaphors of co-operation support systems
 - frameworks of co-operation support systems
 - mobility aspects of co-operation support systems
2. Augmented Reality/Virtual Reality (planned)
3. Computers and Disabled Persons - Access to Information Technology for People with Special Needs
4. Computer-Generated Visualisation
 - definition, history, aims
 - raw data
 - the viewer and the visualisation aims
 - "mapping" of data to images
 - visual representations
 - systems and tools
5. Digital Image Processing
 - visual perception, sampling and quantisation
 - Fourier transformation
 - Fast Fourier transform, sampling, other transformations
 - point processes
 - neighbourhood processes
 - filtering in frequency domain, image restoration
 - image compression
6. Design of Interactive Systems
 - design principles
 - user-centred design
 - Xerox star

- direct manipulation
 - rooms system
 - fisheye views
 - see-through tools, click-through tools, magic lenses
 - information visualizer, 3D rooms
7. Concepts of Digital Media
- instrument, machine, automaton, interactive system
 - hypertext
 - groupware
 - virtual communities
 - virtual reality, augmented reality
 - ambient computing, mediatronics
8. Practical Usability Engineering
- This class is project-oriented – design projects in classic user interfaces and/or web presences are carried out during the semester. This course is a practical challenge for students, in particular in their role as user test developers.
9. Programming Interactive Systems (in English)
- introduction
 - the implementation problem
 - basics in computer graphics
 - event-oriented programming
 - implementation of fundamental interaction
 - user interfaces from widgets
 - implementation of web sites
10. Programming Interactive Web Sites (planned, in English)
- introduction
 - basic principles of interactive web sites
 - client-side technology
 - server-side technology
11. Software Ergonomics
- occupational health and safety laws and other regulations
 - international standards and standards of software ergonomics
 - theoretical basics of design
 - a) perception
 - b) memory
 - c) iconicity and textuality
 - guiding principle: "reduction of forced sequentiality"
 - presentation criteria
 - interaction criteria
 - embedding criteria (conventions)
 - specific aspects of web design
12. Web Modeling (planned)
- basic principles of model-based approaches
 - WebML
 - OOHDM
 - task-model-based web modeling
 - implementation questions

Prerequisites and prior knowledge

The programming-oriented classes (Programming Interactive Systems and Programming Interactive Web Sites) require proof of successful participation in the software technology practical. The prerequisite for Practical Usability Engineering is Usability Engineering or, alternatively, User Interface Modeling. The latter is a prerequisite for Web Modeling. The classes in the area of computer graphics require Computer Graphics I.

Learning goals of the module

In this module's classes, students can expand their knowledge in different areas of human-machine interaction. With regard to computer graphics, we offer in-depth study of data visualisation and image processing based on the basic concepts in computer graphics already learned. In user interface development, the model concepts learned before are consolidated and complemented by implementation concepts and techniques from web programming. In usability engineering, we teach practical user test development skills, including interacting with subjects and practical experimental set-up. Applying web usability techniques is another possible area of specialisation. In design, students should get to know and apply the possibilities for carrying out user- and task-specific user interface designs. These interfaces can be classic software or web sites that may also have to meet general design rules, standards, or regulations. We will use relevant examples and independent development projects to aid learning. The necessary knowledge and skills for the design activities are, among others, taught in Design of Interactive Systems and in Software Ergonomics.

Key qualifications

- Ability to use modern technologies in human-computer interaction
- Subject-specific foreign language skills because the accompanying literature is in English (for non-native speakers of English)
- Ability to co-operate and work in teams is fostered in group projects

Module assignment:

Elective module in the area of Human-Machine Interaction (MMWW)

Mode

- Credit points: 4 + 4 ECTS (2 catalog classes)
- SWS: 2+1, 2+1 (or 2, if a seminar is selected)
- Frequency of the module offered: 2-3 catalog classes per year in the winter and summer semesters

Methods of implementation

The theoretical concepts are expanded in small groups during tutorial classes. Students will test the methods in practical tutorials using development tasks with respective contents that depend on the additional class selected.

Organisational arrangements / media use / literature references

- For classes: one two-hour lecture and a one hour tutorial class per week (or one two-hour tutorial class every second week), and/or solving of programming tasks, or development of modeling documents or usability tests in one's own time and to a similar extent
- Typically, the seminar is a block seminar, but it may also take place regularly during the semester, as announced
- References regarding activities expected of students: independent programming; co-operation during tutorial classes; contact with human subjects during user tests, where applicable.
- Materials used: exercise sheets

Examination modalities

- Written examination
- Solution of practical problems via programming tasks and/or independent projects
- Information on grading policies: The weighting of written examination and programming assignments/projects is announced at the beginning of the semester.

Person responsible for the module

Keil

III.4.7 Human-Computer Interaction

Role in the degree course

The productivity of computer systems depends to a large degree on their usability. In addition, health effects as a result of using software are now known and internationally accepted as a hazard. Because of respective legislation, health protection for VDU workers now places certain requirements on designing user interfaces. Productivity and health protection are thus two important and complementary factors that faculty must consider when educating computer scientists. Another factor is the accessibility, now defined through international standards, that aims to provide equal access to digital information offerings for all citizens.

Here, it is essential to link constructive design approaches and development tools and methods with analytic concepts on the basis of cognitive psychology. Computer scientists should learn to detect the most important requirements and to convert them into corresponding technical solutions. Apart from the knowledge of relevant laws and standards, this requirement includes theoretical, design-oriented, and methodical foundations and the respective specification techniques. This module belongs to the modules in Human-Machine Interaction (MMWW)

Content structure of the module:

The module consists of classes focusing on the different facets of analysing, creating, and designing interactive systems and co-operative media. We will consider the general regulatory framework in this class, as well as its cognitive-psychological foundations. Accordingly, the module consists of the following classes:

- Introduction to Human-Computer Interaction
- Software Ergonomics
- Design of Interactive Systems
- Practical Usability

Usability of the content

The knowledge students will gain in this module provides a broad foundation for developing interactive systems. In today's world, such knowledge is essential as the effective and reliable interaction with the computer constitutes a decisive component in using modern data processing systems. At the same time, this field provides many areas that tie in with special application fields such as e-learning, web-based work, mobile devices, visualisation, virtual reality, information design, etc., and with other scientific disciplines such as psychology, pedagogy, and communication sciences.

Prerequisites and prior knowledge

The significant concepts from the first four semesters in computer science are prerequisites. In particular, these prerequisites include modeling and the respective specification languages, and software tools for developing interactive systems. Knowledge in graphical data processing and scientific visualisation provides a helpful addition but is not necessarily required.

Learning goals of the module

The module should enable students to analyse, assess, and constructively improve interactive systems at all levels and of all types. They should become familiar with the regulatory requirements and the relevant international standards. In some areas, the aim is to become familiar with special solutions and exemplary approaches for implementing high-quality user interfaces and to assess the potential for innovation.

Module assignment:

Elective module in the field of Human-Machine Interaction.

Mode

Credit points: 4+4

SWS 2+1, 2+1

Frequency of the module offered: One basic class every winter semester, the advanced classes every summer semester.

Examination modalities

Oral or written examination, depending on the number of participants

Person responsible for the module

Keil