Please note: The German version of this document is the legally binding version. The English translation provided here is for information purposes only.

Algorithms and Data Structures								**Abbr. ADS**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
2.2	**150 h**	**5**	2nd sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	45h		15	German

2 Learning outcomes / competences

- The students know the O-calculus and can use it to estimate the runtime behaviour of algorithms.
- They know procedures for algorithm development and strategies for solving optimisation problems and can apply them to examples.
- Students know standard data structures, as well as several types of trees, and can use them for modelling and in software development.
- They know the areas of application and the advantages and disadvantages of various search and sorting methods and can use them purposefully in programming.
- They know different methods of hashing, can evaluate them and apply them to examples.
- Students know some graph algorithms and applications of them. They can select and implement an appropriate procedure in each case.
  They know about the complexity classes P and NP and their meaning.

3 Contents

- O-calculus
- Algorithm and programme specifications
- Algorithm development with stepwise refinement, recursion, libraries and design patterns
- Algorithms for solving optimisation problems, optimal and non-optimal methods: dynamic programming, backtracking, divide-and-conquer, Greedy method
- Data structures: Sequence, linked list, stack, queue, skip list
- Trees: for example binary trees, AVL trees, 2-3-4 trees, red-black trees
- Search methods and sorting methods: for example HeapSort, SelectionSort, InsertionSort, BubbleSort, QuickSort, MergeSort
- Hashing
- Graph algorithms: breadth-first search, depth-first search, topological sorting
- Complexity classes P and NP

4 Participation requirements

None

5 Form of assessment

Performance test

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Dipl.-Inf. B.C. George

9 Other information

Operating Systems								**Abbr. BES**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
4.1	**150 h**	**5**	4th sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

Operating systems manage the resources of a computer system and thus essentially determine its performance and usability. Good knowledge of operating system concepts is therefore essential for understanding modern IT systems. Students will have the following skills after completing the course:
-Know, understand and recognise system-related abstractions as used and provided by operating systems
- Safely apply important procedures and algorithms from the field of operating systems
- Know and understand methods used in the design of operating systems
- be able to explain both in detail using example systems (primarily UNIX/Linux, partly Windows and other current operating systems)
- Students can create hardware-related software (e.g. Linux kernel drivers) in a team and draft an argument/strategy to justify design decisions.

3 Contents

- System architecture of modern computing systems
- Tasks, types and structures of operating systems
- brief presentation of current computer architectures (pipeline, cache, branch prediction), if applicable
- Concurrency, process management, scheduling
- Synchronisation and communication
- Memory management: Virtual memory, segmentation, paging, page replacement algorithms
- Input/output: character and block-oriented devices
- File systems: Examples, structure, fault tolerance
- Driver models and programming
- Security: Authentication, protection mechanisms, authorisation, trusted systems
- Changing contents of the practical courses on current topics (e.g. Bash scripting, development of a shell, development of a Linux kernel driver)

4 Participation requirements

Formally: -, Content: Knowledge from module 3.3 System Programming (SP)

5 Form of assessment

Performance test

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr.-Ing. Martin Hoffmann

9 Other information

Literature:
- Tanenbaum: "Modern Operating Systems", Pearson Studium, 2016
- Stallings: "Operating Systems: Internals and Design Principles", Prentice Hall, 2011

Computer Graphics								**Abbr. CG**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
4.3	**150 h**	**5**	4th sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical	2 SCH/30 h	45 h		15	

2 Learning outcomes / competences

Students can apply the fundamental methods and algorithms in computer graphics. Students are able to name basic terms and explain 2D as well as 3D computer graphics processes. They can summarise the mathematical basics for 2D and 3D computer graphics. Students are able to use common tools from computer graphics and are familiar with the associated technologies.

3 Contents

- History, overview, examples:
  Tools, application examples.
- Concepts and basics:
  Graphic input devices, screen technologies, 3D vision systems, raster graphics.
- Object and view transformations:
  Coordinate systems, transformations, projections, clipping.
- Representation and modelling of objects:
  Polygonal representation, spatial division methods, scene description.
- Rendering and visibility:
  Colour models, visibility techniques, lighting and shading, local lighting models, interpolative shading techniques, global lighting models, rendering pipelines.
- Modelling/rendering tools:
  Modelling and rendering of a small scene using e.g. Autodesk Maya, Blender.

4 Participation requirements

Formal: -

Content: Knowledge from module 1.1 Mathematics 1 (MA1)

Section 17 "Progress regulation" of this BPO applies

5 Form of assessment

Performance test

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Kerstin Müller

9 Other information

Literature

- Bender M., Brill, M.:
  Computer Graphics, 2nd
  edition,
  Hanser Verlag, 2005 *http://www.vislab.de*
- Hearn D., Baker M.P.:
  Computer Graphics with OpenGL,
  Pearson International Edition.
- Foley J., van Dam A., Feiner S., Hughes J.:
  Computer Graphics - Principles and Practice,
  Addison-Wesley

Database Systems I								Abbr. DB1
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
3.2	**150 h**	**5**	3rd sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical course	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

Databases are a basic technology of computer science. The aim of the course is to provide a secure understanding of the basic concepts and languages of database systems. The students apply a uniform, consistent conceptual framework with regard to database topics. They can explain the theoretical foundations of database systems using examples, in particular functional dependencies, relational algebra and normalisation, they know about the tasks and the meaning of the basic architecture of DBMS. They can search the documentation of several important DBMSs in a targeted problem-solving manner and know the basic functions of the clients of several DBMSs and use them to communicate and program databases. They model complex issues securely in even extensive data models and implement them in various DBMS. In doing so, they make reasoned decisions for the applications of constraints, domains and data types. They apply SQL confidently to solve complex information needs and create extensive non-trivial queries. They use both the current SQL standard (currently SQL:2016) and the dialects of several important DBMSs. They will understand the transaction concept, describe problems/phenomena of multi-user synchronisation and concurrency in read/write notation, and decide how to prevent them by isolating transactions - both through standard isolation levels and through specific implementations in multiple DBMSs. They access databases from their own programmes via database interfaces and process data records in programmes and databases. They programme Persistent Stored Modules in one of the DBMSs discussed.

3 Contents

The following topics are examples of possible content:

- Tasks and architecture of database systems
- Clients and interfaces to database systems
- Basics of the relational model
- E/R modelling, logical and physical data models, SQL data types, implementation in important DBMSs
- Constraints, assertions, integrity, domains, data types
- SQL:2016, in particular SQL-schema statements, SQL-data statements, SQL-data change statements, SQL-transaction statements and SQL-connection statements
- Transaction concepts, concurrency, isolation level
- Database interfaces (JDBC, ODBC)

Basics of Persistent Stored Modules, programming of PSM, triggers

4 Participation requirements

Formal: -
Content: -

5 Form of assessment

Performance exam or oral exam or term paper or project work or internship, excursion or daily log or portfolio or learning diary or parcour exam

6 Condition for the award of credit points

Practical course with test
Passed module examination

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Dominic Becking

9 Other information

	• Kleuker, S., Grundkurs Datenbankentwicklung, 4th ed. Vieweg Teubner,2016 • Kemper, A, Eickler, A, Database Systems - An Introduction, 10th ed. De Gruyter, 2015 • Elmasri, R. A., Navathe, B. N., Fundamentals of Database Systems, Hanser, 2009 • Piepmeyer, L., Grundkurs Datenbanksysteme, Hanser, 2011 • Saake, S., Sattler, K.-U., Heuer, A., Datenbanken - Konzepte und Sprachen, mitp, 2010 Current literature on database systems

Introduction to Computer Science								**Abbr.** ONE
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
1.0	**150 h**	**5**	1st sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course.	60	German
	Exercise	2 SCH/30 h	45 h		30	German

2 Learning outcomes / competences

The roots and development history of computer science should be understood. Students should recognise the reciprocal effects of society on computer science and vice versa and learn to take a position on this themselves. A special aspect is dedicated to gender equality in computer science in research, teaching and development. The students should get to know diverse job profiles of computer scientists from different industries. The subject area should be understood in its diversity and breadth and the conceptual sub-areas. The students should be enabled to make a targeted choice of subjects in their subsequent studies, taking into account their strengths and inclinations and with regard to their future intended professional field. Students should be able to be encouraged in their choice of study and motivated for successful study.

3 Contents

- History of computer science
- Achievements in computer science
- Informatics and Society
- Gender equality in computer science
- Subfields of computer science
- Importance of data protection
- Job profiles of computer scientists
- Information storage
- Propositional logic
- Formal languages
- Number representations
- Algorithms
- Coding
- Artificial intelligence
- Internet

4 Participation requirements

None

5 Form of assessment

Written examination

6 Condition for the award of credit points

Passed exam

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr.-Ing. Grit Behrens

9 Other information

Introduction to Programming with Scripting Languages								**Abbr.** **EPS**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
1.3	**150 h**	**5**	1st sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

- Students know at least one scripting language and can use it to develop their own applications.
- They know structural elements of imperative programming languages and use them in their own programmes.
- They can apply language concepts of scripting languages in their own applications.
- They can analyse data in terms of its structure in simple examples and identify it as a JSON or XML document.
- They can create and process JSON or XML documents using a scripting language.
- They understand different aspects of teamwork and can identify their team roles according to Belbin.

3 Contents

- Concept of algorithms
- Structural elements of algorithms
- Properties of scripting languages, their advantages and disadvantages
- Language concepts of scripting languages
- Structure of JSON or XML documents
- Standard libraries for processing JSON or XML files
- Team roles according to Belbin

4 Participation requirements

None

5 Form of assessment

Performance test

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Dipl.-Inf. B.C. George

9 Other information

Embedded Systems								**Abbr. ES**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
4.2	**150 h**	**5**	4th sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

The students learn the basic knowledge for the implementation of embedded systems. They will gain an understanding of the specifics of embedded systems design and the skills needed to implement embedded systems. In particular, students are given the appropriate specification and programming techniques, models of sequence planning and software and system architectures for embedded systems as "tools of the trade".
Students are taught the typical design steps for developing software for embedded systems using exemplary application scenarios.

3 Contents

- Overview and areas of application
- Special features and requirements
- Fundamentals of electronics and hardware of embedded systems
- Software development toolchain for embedded systems
- Microcontroller programming
- Interaction of software and hardware
- Design steps
- Specification and modelling languages
- Real-time operating systems
- Realisation and implementation
- Validation and evaluation

Practical implementation of embedded systems using application examples.

4 Participation requirements

Formal: Section 17 "Progress Regulation" of this BPO applies.
Content: Knowledge of technical computer science and C++ programming

5 Form of assessment

Performance test

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Dr.-Ing. Matthias König

9 Other information

Literature will be announced in the course.

Scientific Research and Writing								**Abbr. FSI**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.1	**150 h**	**5**	6th sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Tuition in seminars	4 SCH/60 h	90 h	To be announced in course	35	German and English

2 Learning outcomes / competences

The ability to work independently in a scientific manner with the development of content as well as the comprehensible presentation of technical topics are indispensable for everyday professional life.
Graduates of the module

- are able to work independently on a specialist topic using specialist literature and other sources
- can present and explain a subject in a comprehensible way
- acquire communicative competence
- deepen specialist informatics competences in the selected subject area of the seminar.

3 Contents

Self-organisation and independent work on a specialist topic

- Knowledge management and literature study (research, dealing with citations and citing specialised literature)
- Subject-specific writing for the written paper
- Presentation technique and rhetoric for the presentation of the topic
- Discussion within the framework of the seminar participants and supervising lecturers

4 Participation requirements

None

5 Form of assessment

Oral examination or term paper or project work or field trip or daily log or portfolio or learning diary or course examination.

6 Condition for the award of credit points

Certificate
Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Dominic Becking

9 Other information

Literature: Publications on the chosen topic in German and English language

Business Administration								**Abbr. BWL**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
4.5	**150 h**	**5**	4th sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	90 h	To be announced in course	60	German
	Exercise	2 SCH/30 h			30	German

2 Learning outcomes / competences

Students acquire knowledge of business contexts in the professional environment of a computer scientist. They know essential business management procedures and basic terms, have an overview of legal forms of companies, investment and financing and production planning and control. They have an overview of marketing strategies. They have become familiar with the relationship between algorithms from computer science and problems from business administration using selected quantitative examples (e.g. location planning, determination of demand).

3 Contents

- Basic concepts of business administration
- Business strategies
- Decision theory
- Costs and controlling
- Investment and financing
- Production
- Logistics
- Marketing
- Human resources and gender aspects

4 Participation requirements

Section 17 "Progress Regulation" of this BPO applies.

5 Form of assessment

Term paper or written exam or oral exam

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Dr.-Ing. Matthias König, (covered by Prof. Dr.-Ing. Martin Hoffmann)

9 Other information

Literature will be announced in the course.

IT Law								**Abbr. ITR**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
2.4	**150 h**	**5**	2nd sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Sem. lessons	4 SCH/60 h	90 h	To be announced in course	35	German

2 Learning outcomes / competences

Students know the most important legal aspects they may come into contact with while working in the field of information technology. In particular, they can assess which rights and obligations arise in contracts regarding the manufacture, distribution and use of (software/hardware) products, which intellectual property rights are associated with which products, which property rights can be used to protect intellectual property, how data protection must be observed, as well as what consequences can be expected in the event of legal violations.

3 Contents

The content relating to information technology is essentially cross-sectional:

- Civil law and contract law
- Product liability
- Data protection
- Criminal law
- Telecommunications, telemedia and internet law

Intellectual property law (inter alia copyright, patent, trademark law)

4 Participation requirements

Section 17 "Progress Regulation" of this BPO applies.

5 Form of assessment

Term paper or written exam or oral exam

6 Condition for the award of credit points

Participation in seminar classes with a test. Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Dr.-Ing. Matthias König

9 Other information

Literature will be announced in the course.

Mathematics for Computer Scientists I								**Abbr. MA1**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
1.1	**240 h**	**8**	1 sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture, exercise	4 SCH/60 h 2 SCH/30 h	90 h 60 h	To be announced in course	60 20	German

2 Learning outcomes / competences

Students understand the basic mathematical concepts as well as the basics of set theory and propositional logic. They can select suitable evidence procedures. Vector and matrix calculations as well as functions can be used by the students and the solving of linear equation systems can be applied to examples. Students have understood and can apply differential and integral calculus.

3 Contents

Basics
- Number ranges
- Set theory
- Propositional logic
- Full induction

Linear algebra
- Vectors and vector spaces
- Matrices and linear mappings
- Linear systems of equations

Analysis I
- Sequences and series
- Real functions of one variable
- Differential calculus
- Integral calculus

4 Participation requirements

None

5 Form of assessment

Written exam with preliminary examination

6 Condition for the award of credit points

Passed exam

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Kerstin Müller (covered by Dipl. Inf. Birgit Christina George, Prof. Dr. Christoph Thiel, Dr. Jan Thies)

9 Other information

Literature

Hartmann, Peter: Mathematik für Informatiker, Vieweg.
Manfred Brill: Mathematik für Informatiker, Hanser Verlag
Bronstein, Semendjajev: Taschenbuch der Mathematik

Mathematics for Computer Scientists II								**Abbr. MA2**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
2.0	**240 h**	**8**	2nd sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	4 SCH/60 h	90 h	To be announced in course	60	German
	Exercise	2 SCH/30 h	60 h		20	

2 Learning outcomes / competences

Students can apply functions with several variables and the associated differential calculus. They know linear differential equations and relevant connections from the field of numerics.
Elementary numerical procedures can be transferred to other situations.
Students are able to select suitable methods for solving elementary stochastic problems. They understand basic concepts of probability theory, important distributions and their significance as well as basic statistical methods.

3 Contents

Analysis II
- Local and global approximation
- Differential equations
- Real-valued functions with several variables
- Differential calculus for functions of several variables

Numerics
- Error and error propagation
- Elementary numerical methods

Probability calculation and statistics
- Combinatorics
- Probability calculation
- Random variables
- Distributions
- Statistics

4 Participation requirements

Formal: -

Content: Knowledge from Module 1.1 Mathematics 1 (MA1)

5 Form of assessment

Written exam with preliminary examination

6 Condition for the award of credit points

Passed exam

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Kerstin Müller (covered by Dipl. Inf. Birgit Christina George, Prof. Dr. Christoph Thiel, Dr. Jan Thies)

9 Other information

Literature

Hartmann, Peter: Mathematik für Informatiker, Vieweg.
Bronstein, Semendyajev: Taschenbuch der Mathematik

Object-Oriented Programming								**Abbr.** **OOP**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
1.2	**210 h**	**7**	1st sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	75 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	75 h		15	German

2 Learning outcomes / competences

- The students know concepts of object orientation and can use them to develop their own software applications.
- The students know the Java programming language and can use it to develop their own software.
- They know elements for documentation and apply them in their own programmes.
- They can identify and create object-oriented solutions for simple problems.
- The students know programming tools and can use them practically.
- They know methods for exception handling and apply them.
  Students learn to use standard libraries in their own implementations in a targeted manner.

3 Contents

- Introduction to object-oriented design
- Introduction of the concept of class and object
- Use of a development environment and a debugger
- Data types and control structures
- Introduction to inheritance, polymorphism
- Introduction to exception handling
- Introduction of collection classes
- Introduction to Swing
- Approaches to testing
- Approaches to documentation (for example JavaDoc)

4 Participation requirements

None

5 Form of assessment

Performance test

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr Jörg Brunsmann

9 Other information

Programming Methods and Techniques								**Abbr. PM**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
2.1	**210 h**	**7**	2nd sem.	Annual	Summer	1 Sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH / 30 h	45 h	To be announced in course	60	German
	Practical course	3 SCH / 45 h (of which 1 SCH are tutor supervised)	90 h		15	German

2 Learning outcomes / competences

After attending the course, students are able to use a comprehensive set of techniques and solution patterns for software development in the Java programming language. The successful student will be able to select and use key standard architectural patterns appropriate to the task. Students are able to develop more complex applications and their own libraries. Students master basic techniques and workflows for source code version management and are able to actively apply them in projects. The students recognise "bad smells" and are able to eliminate them with the help of refactoring through safeguarding by means of self-formulated unit tests.

3 Contents

- Modelling, Pattern, OOD:
  - Advanced object-oriented design
  - Deepening understanding of class diagrams with UML
  - Basic design patterns (Strategy, Decorator, Template, Factory-Method, Observer)
  - Inheritance vs. composition vs. generics
- Programming methods:
  - Version management using the example of Git, branching strategies and workflows
  - Systematic unit test with JUnit
  - Systematic documentation with JavaDoc
  - Coding Conventions, Bad Smells, CheckStyle
  - Refactoring with IDEs
  - Build systems (ANT, Maven, Gradle)
  - Introduction to Continuous Integration (CI)
- Advanced Java knowledge:
  - Annotations and enumerations
  - Regular Expressions in Java
  - Generic Programming (Generics)
  - Logging in Java
  - Configuration (Properties, Preferences, CLI)
  - Anonymous and inner classes
  - Reflection, Serialisation
  - Introduction to parallel programming with Java: Threads, synchronize, wait, notify
  - Java8: Default methods, method references, functional interfaces, language elements for functional programming
  - Event-oriented programming (Swing or JavaFX, Java2D), models and event hierarchies
  - Java and XML: JAXB
  - Use of software libraries (APIs): for example Apache CLI, Apache POI, JFreeChart

4 Participation requirements

None

5 Form of assessment

Performance exam or written exam

6	**Condition for the award of credit points** Testate and passed module examination
7	**Application of the module** (in the following study programmes)**:** Computer Science (B.Sc.)
8	**Module coordinator** Prof. Dr.-Ing. Carsten Gips
9	**Other information** • Deitel, Deitel: "Java – How to Program", Pearson Education Limited, 2012 • Bloch, J.: "Effective Java: A Programming Language Guide", Addison-Wesley, 2011 • Urma, Fusco, Mycroft: "Java 8 in Action", Manning Publications, 2014 • Chacon, Straub: "Pro Git", Apress, 2014 • Robert Martin: "Clean Code", Prentice Hall, 2008 • Martin Fowler et al.: "Refactoring", Addison Wesley, 1999 • Roy Osherove: "The Art of Unit Testing", Manning, 2013 • Kent Beck: "Test Driven Development", Addison-Wesley, 2002 • Gamma et al.: "Design Patterns", Addison-Wesley, 2011 • Ullenboom, C.: "Java ist auch eine Insel", Rheinwerk-Verlag, 2016

Software Engineering								**Abbr. SE**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
3.0	**210 h**	**7**	3rd sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	75 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	75 h		15	German

2 Learning outcomes / competences

- The students learn the systematic and structured procedure for the successful planning and implementation of a software development project and practically apply agile procedure models for this purpose.
- They will learn about relevant phases from requirements analysis to high-level design, low-level design, implementation and quality assurance.
- They will learn in detail the notation elements and diagram types of the UML standard and apply them to a software project.
- Students learn about architecture and design patterns as well as JUnit tests

3 Contents

- Introduction to Software Engineering
- UML diagrams (e.g. modelling of business processes with activity diagrams)
- Process models (waterfall model, agile process models such as Scrum and Extreme Programming)
- Requirements analysis (stakeholders, objectives, use cases, derivation of functional requirements, non-functional requirements, requirements and specifications)
- High-level design (system architecture, derivation of basic classes, methods, sequence diagram, interface development considerations)
- Programme generation (translation of classes and associations, types of object membership, software architecture)
- Low-level design (details in miniature, model view controller, GoF pattern)
- Implementations (Distributed Systems, Libraries, Components, Frameworks, Persistent Data Management)
- SW quality assurance (assurances, unit tests, test procedures, metrics)

4 Participation requirements

Formally: -, Content: Knowledge of object-oriented programming (OOP), programming methods (PM)

5 Form of assessment

Performance test

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr Jörg Brunsmann

9 Other information

Software Project								**Abbr. SWP**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
4.0	**150 h**	**5**	4th sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Practical / Seminar	4 SCH/60 h	90 h	To be announced in course	15	German

2 Learning outcomes / competences

In close dovetailing with the "Project Management" module students apply the essential basics of project management. They plan a major software project, implement it, manage it and regularly document and present project progress and results.
They make a well-founded decision on a process model in the given project context. They apply the approach, organisational forms and methods of a recognised formal project management system to their project.
They realise a project in a larger project group (approx. 8 people) with role allocation.

3 Contents

- Establishment and implementation of a project
- Drawing up a specification sheet based on the specifications of a fictitious "customer", carrying out an effort estimate on the basis of this specification sheet
- Establish a project plan and procedures for project tracking and risk management
- Implementation in the team (version management, build procedures, coordination processes, interfaces)
- Presentation of results and interim results
- Use of current technologies to implement the application
- Final presentation

4 Participation requirements

Formal: -
Content: Knowledge from Object-Oriented Programming (OOP)
Section 17 "Progress Regulation" of this BPO applies.

5 Form of assessment

Project work

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Lecturers in the computer science study programme (Becking, Behrens, Brunsmann, George, Gips, Hoffmann, Kreienkamp, König, Müller, Thiel)

9 Other information

Software Project Management								**Abbr.** **SPM**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
3.4	**150 h**	**5**	3rd sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Seminar lessons	4 SCH/60 h	90 h	To be announced in course	35	German

2 Learning outcomes / competences

Students apply the essential basics of project management confidently to exemplary projects. They are able to plan, implement and manage parts of projects as well as document and present project progress and results.
They can make a well-founded decision in favour of a process model in a project context. They know the procedure, organisational forms and methods of a recognised formal project management system.

3 Contents

- Projects as problem-solving processes
- Foundation, organisation and structuring of projects
- Project planning
- Project management
- Software project management
- Tools in project management
- Communication and documentation as a cross-sectional task
- Quality assurance
- Project management systems

4 Participation requirements

None

5 Form of assessment

Oral examination or term paper or project work or internship, excursion or daily log or portfolio or learning diary or course examination

6 Condition for the award of credit points

Certificate
Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr Dominic Becking

9 Other information

Literature:

- o.V., Managing Successful Projects with PRINCE2® 2009 Edition, Axelos 2009
- o.V., A Guide to the Project Management Body of Knowledge (PMBOK® Guide), PMI 2012

System Programming								**Abbr. SP**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
3.3	**240 h**	**8**	3rd sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	3 SCH/ 45 h	50 h	To be announced in course	60	German
	Practical course	3 SCH/ 45 h	100 h		15	German

2 Learning outcomes / competences

Students acquire comprehensive competences for the development of system-related software in the current system programming languages C and C++, taking into account common standards (e.g. ANSI-C/C11 and C++14/C++17). They are proficient in current tools in this environment, for example the Gnu compilers (gcc, g++) including debugger (gdb) and Make, and know various current standard libraries. Program development using essential parts of the UNIX/Linux programming interface (POSIX) is mastered. Students can apply this knowledge independently to more complex tasks (practical tasks).

3 Contents

- Basics C/C++
  - Types, expressions, operators, control flow
  - Structures and enumerations, typedef
  - Bit operations
  - Functions, declaration vs. definition, prototypes, call-by-value
  - Visibilities and scopes, global vs. local variables, static vs. external, cross-file
  - Storage classes
  - References in C++
  - Overloading operators and functions in C++
  - Use of a debugger, e.g. gdb
  - Unit test in C++, e.g. with cppunit or googletest
- Object-oriented programming in C++
  - Classes, Constructors, Destructors, Copy Constructor, Assignment Operator
  - Friends
  - Operators
  - Separation of interface and implementation
  - Inheritance, polymorphism, virtual, slicing, abstract classes, multiple inheritance
- Modular programming
  - Dividing code into header and implementation files
  - One Definition Rule
  - Preprocessor: Include, macros, conditional translation, constants
  - Static and dynamic libraries, linkers
  - Makefiles
- Memory management
  - Memory management under Linux, virtual memory, stack vs. heap
  - Pointers and addresses, declaration, dereferencing, assignment
  - Dynamic memory management with malloc/free and new/delete
  - Problems with memory management: Memory Leaks, Stale Pointer, Double Delete
  - Call-by-Reference in C using Pointers
  - Connection between pointers and arrays, multidimensional arrays, CMD parameters
  - Address arithmetic
  - C strings and functions from the C-Std-Lib (e.g. strcpy, strcat, strtok)
  - Function pointer
  - SmartPointer in C++
  - Reading complex declarations
- Input and output, handling directories
  - System functions under Linux, file abstraction, standard I/O channels
  - Handling files under C, scanf/printf

	- Streams in C++, error states, manipulation • Error handling – Signalling of errors in the Linux system interface (return value, errno), abort/exit/atexit – Dealing with exceptions (C++) – Assertions • Standards: ANSI C vs. C11, C++11 vs. C++14 vs. C++17 • Metaprogramming with templates (functions, classes) • Introduction to standard libraries (e.g. STL, Boost) • C++14: Move semantics, SmartPointer • Process and thread manipulation (create, terminate, states, zombies/orphans) • Inter-process communication: Signals and sockets, overview of other IPC forms • Time (calendar, time, timing and timer) • Embedding/integration of other languages (e.g. Python, Lua) • Use of libraries, e.g. SQLite3, libXML2, libCurl • System-oriented programming under Linux on the Raspberry Pi (C/C++) • Documentation with Doxygen • Safe and defensive programming • Changing contents of the internships on current topics
4	**Participation requirements** None
5	**Form of assessment** Performance exam or written exam
6	**Condition for the award of credit points** Testate and passed module examination
7	**Application of the module** (in the following study programmes)**:** Computer Science (B.Sc.)
8	**Module coordinator** Prof. Dr.-Ing. Carsten Gips
9	**Other information** • Breymann, U.: "Der C++ Programmierer", Hanser, 2011. • Scott Meyers: "Effective Modern C++", O'Reilly, 2014 • Klemens, B.: "21st Century C", O'Reilly, 2014 • Brian Kernighan, Dennis Ritchie: "The C Programming Language", Prentice Hall, 2000 • Love, O.: "Linux System Programming", O'Reilly Media, 2013 • Kerrisk, M.: "The Linux Programming Interface", no starch press, 2011

Technical English								**Abbr. TE**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.0	**150 h**	**5**	5th sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Sem. lessons	4 SCH/60 h	90 h	To be announced in course	35	English

2 Learning outcomes / competences

Participants should be able to express themselves using technical English. They should be able to use these acquired language skills in an international environment, both written and spoken. Students will learn the specific terminology used in the field of computer science as well as general technical English and will be able to describe and explain processes, solve technical problems, and use technical terminology to discuss and communicate IT solutions.

3 Contents

- Computer Hardware: Input and output media, printer, storage media
- Essential and creative software: Operating system, word processing, spreadsheets and databases, desktop publishing, multimedia
- Programming: Programming languages, Java, web design
- Technologies of the future: Communication systems, networks, video games
- Internet: E-mail, the web, videoconferencing

Employment opportunities in the ICT sector

4 Participation requirements

Section 17 "Progress Regulation" of this BPO applies.

5 Form of assessment

Written examination

6 Condition for the award of credit points

Participation in seminar lessons with a test
Passed written exam

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr.-Ing. Martin Hoffmann

9 Other information

Computer Engineering								**Abbr. TI**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
1.4	**150 h**	**5**	1st sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	90 h	To be announced in course	60 (L)	German
	Exercise	2 SCH/30 h			30 (E)	German

2 Learning outcomes / competences

After completing the course, students can

- calculate simple DC circuits in electrical engineering using Ohm's law and Kirchhoff's laws;
- assign characteristics and areas of application of different types of processors;
- describe the basic structure and functional units of a processor and describe their functions;
- name basic processor architectures, list their characteristics or advantages and disadvantages, identify existing architectures (in block diagram);
- name the most important command and addressing types, understand commands using the data sheet and convert them into assembler/machine code;
- present the principles of error detection and error correction, derive the structure and properties of a Hamming code and use it as an example;
- compare storage technologies and name their examples of use;
- apply the rules of Boolean algebra;
- create digital circuit diagrams from functional equations and vice versa;
- read normal form from truth tables and minimise using KV diagrams;
- describe the basic structure of bistable circuits, distinguish their classifications from each other; name the properties and examples of use of RS, D, JK and T flip-flops and explain their mode of operation using pulse diagrams.

3 Contents

Basics of the electrical direct current circuit

- Ohm's Law
- Kirchhoff's Laws

Fundamentals of computer architectures

- Structure of processors with processor types, architectures, control/computing unit and registers,
- Introduction to hardware-related programming
- Program and data storage with storage organisation, storage technologies
- Error detection and correction in data transmission

Fundamentals of digital technology

- Boolean algebra, normal form and minimisation
- Digital circuits and flip-flops

4 Participation requirements

None

5 Form of assessment

Written examination

6 Condition for the award of credit points

Passed exam

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Angela Kreienkamp (MPrComp)

9	**Other information**
	Literature (e.g.): • Elektrotechnik Grundlagen, Steffen, Bausch • Digitaltechnik - Ein Lehr- und Übungsbuch, Woitowitz, Urbanski • Mikrocontroller und Mikroprozessoren, Brinkschulte, Ungerer • Logischer Entwurf digitaler Systeme, Liebig • Mathematik sehen und verstehen, Haftendorn

Theoretical Computer Science								**Abbr. THI**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
2.3	**150 h**	**5**	2nd sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Exercise	2 SCH/30 h	45 h		30	German

2 Learning outcomes / competences

Students can

- develop automata and grammars
- give the corresponding language classes to given finite automata, pushdown automata, different types of grammars and Turing machines and vice versa
- understand the Chomsky hierarchy and assign it to the classes of languages and automata
- explain and discuss problems of computability, decidability and the halting problem
- discuss the P-NP problem with the help of examples

3 Contents

- Automata (with output, deterministic, non-deterministic)
- Regular terms
- Grammars, context-free languages
- Pushdown automaton
- Context-sensitive and type 0 languages, Turing machine
- Chomsky hierarchy
- Calculability, decidability, Church's thesis, halting problem,
- Complexity theory, P-NP problem

4 Participation requirements

None

5 Form of assessment

Written examination

6 Condition for the award of credit points

Passed exam

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Dipl.-Inf. BC George

9 Other information

Usability and Data Visualization								**Abbr. UDV**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.0	**150 h**	**5**	6th sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Sem. lessons	4 SCH/60 h	90 h	To be announced in course	35	German

2 Learning outcomes / competences

After completing the course, students can

- describe the possibilities of perception and the processing of information in humans;
- name and describe common techniques for entering and presenting information;
- present various interaction models and apply them by example;
- explain different principles and techniques of data visualisation;
- describe and compare individual usability concepts and principles;
- describe, compare and apply different methods for usability efficiency measurement;
- describe and compare essential methods of the design process;
- name relevant methods of evaluation, explain their characteristics and apply them by way of example

3 Contents

- Psychological, physical and theoretical aspects of the interaction of humans in their environment (HCI-Human Computer Interaction)
- Design principles and techniques for visualising data
- Models and methods of usability engineering
- Methods for the evaluation of usability

4 Participation requirements

None

5 Form of assessment

Written exam and/or term paper

6 Condition for the award of credit points

Participation in seminar classes with a test Passed
written exam/term paper

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Kerstin Müller

9 Other information

Literature (e.g.):

- The Laws of Simplicity, J. Maeda
- Human-Computer Interaction, A. Dix, J. Finlay, G. Abowd, R. Beale
- Human-Centered Visualization Environments, A. Kerren, A. Ebert, J. Meyer
- Interaction Design, H. Sharp, Y. Rogers, J. Preece

Distributed Systems and Communication Networks								**Abbr. VES**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
3.1	**150 h**	**5**	3rd sem.	Annual	Winter	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

Communication via networks and the internet form a fundamental part of the modern professional world. Furthermore, Ethernet technologies and TCP/IP-based communication are a fundamental part of most distributed information technology systems. Students learn basic knowledge of digital communication networks with a focus on computer communication. They will learn about the architecture and important methods and tools for distributed systems. They will develop the ability to develop stand-alone distributed systems and network applications. They will learn to evaluate the strengths and weaknesses of different approaches to distributed applications. Students will be able to create distributed software in a team and justify design decisions.

3 Contents

- Fundamentals of communication networks
- Communication models/protocols and standards
- Mediation principles
- Technologies for local area networks (LAN), transmission media
- Stochastic and deterministic media access methods
- Ethernet technologies and protocols
- Protocols of the TCP/IP protocol family (IP, ICMP, UDP, TCP)
- Application layer protocols (HTTP, SMTP etc).
- Basics of IP-based routing protocols
- Router and router configuration
- Routing protocols
- Object-oriented, distributed systems (Java RMI etc.)
- Times and logical clocks

4 Participation requirements

Formally: -, Content: Knowledge of object-oriented programming, data structures and algorithms

5 Form of assessment

Performance test

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr.-Ing. Martin Hoffmann

9 Other information

Web-Based Applications								**Abbr. WBA**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
4.4	**150 h**	**5**	4th sem.	Annual	Summer	1 sem.	Compulsory	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**planned Group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in COURSE	60	German
	Practical course	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

Web-based systems are one of the most widespread forms of distributed information systems and it is impossible to imagine today's private or professional life without them. Students learn to plan, realise and evaluate these professionally.
They will get to know the basic technologies standardised by the W3C and acquire the ability to use them appropriately for the problem. They will get an overview of current open source frameworks for professional web development and use a selection of them for the Design and implementation of own application systems.

3 Contents

- Introduction, classification of web applications, architectures
- Basics (HTTP, session management, standardisation, W3C)
- Designing websites (cascading stylesheets, HTML5)
- Client-side technologies: JavaScript, Ajax, DOM, current libraries and frameworks
- Server-side multi-layer architectures, frameworks for their implementation: e.g. JSF,
- Application server (tasks, services, examples): e.g. Glassfish
- Web services e.g. REST Data exchange formats e.g. JSON

4 Formal participation requirements: -

Content: Knowledge of programming in Java, software engineering, introduction to programming with scripting languages, databases

5 Form of assessment

Performance test

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr.-Ing. Grit Behrens

9 Other information

Literature:

Kurz, Marinschek: „JavaSever Faces 2.2: Grundlagen und erweiterte Konzepte“, dpunkt 2013
Schießer, Schmollinger „Workshop in JavaEE: Ein praktischer Einstieg in die Java Enterprise Edition mit dem Web Profile“, dpunkt 2014
Dean Cemron „HTML5, JavaScript und jQuery“, dpunkt 2015
Somin Timms „Mastering JavaScript Design Patterns“, Packt Publishing 2016

Compulsory Elective Module from List 1 "Methods in Computer Science" Database Systems II: Architectures and Implementation Techniques								Abbr. DB2
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.10	**150 h**	**5**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical course	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

Database systems are highly complex software systems whose safe application and use require in-depth knowledge of implementation and administration. Students can confidently analyse and explain problems in database systems using their acquired theoretical knowledge in advanced database techniques and derive solution approaches from theory. They make reasoned decisions for the application of these techniques. They can install and administer databases, analysing and implementing the requirements of the application software. They will be able to analyse the performance of databases and, by taking appropriate technical measures, also improve this. They can formulate advanced SQL and apply it meaningfully. They can analyse and optimise SQL to increase performance (SQL tuning).

3 Contents

The following topics are examples of possible content:

- Architecture of database systems
- Managing the background memory
- Buffer management
- File organisation and access structures
- Special index structures
- Basic algorithms for database operations
- Advanced SQL
- Request optimisation
- Transaction models
- Transaction management
- Recovery and data backup
- Modern database paradigm

Current problems in database technology

4 Participation requirements

Formal: -
Content: Contents of Module 3.2 "Database Systems I" (DB1)

5 Form of assessment

Oral examination or term paper or project work or internship, excursion or daily log or portfolio or learning diary or course examination

6 Condition for the award of credit points

Practical course with test
Passed module examination

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr Dominic Becking

9 Other information

	Literature: • Saake, G., Sattler, K.-U., Datenbanken: Implementierungstechniken, Heidelberg 2011 • Härder, Th., Rahm, E., Datenbanksysteme: Konzepte und Techniken der Implementierung, Berlin 2001 • Current literature on newer database technologies

Compulsory Elective Module from List 1 "Methods in Computer Science" Introduction to Computer Music								Abbr. MI1
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.11	**300 h**	**10**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	Course type	Contact hours	Self-study	Forms of teaching (forms of learning)	Planned group size	Language
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical course	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

With a turnover of about 2 billion euros in Germany alone, the music industry is one of the most important target industries for the field of media informatics. Music informatics deals with all computer-based techniques and the development of applications for the composition, production, distribution, billing/licensing and enjoyment of music and other audio products. In addition, special aspects of music management, the music business and the technical support of creative processes of music creators are the subject of the field. The students develop a scientific approach to this important sub-field of media informatics by means of a complex project from the field of music informatics.
Students experience and describe music as a complex cultural and technical phenomenon. They analyse aspects of music generation, production and distribution in relation to the role of IT. The students include findings about music as a universal cultural phenomenon in their considerations and familiarise themselves with scientific literature from anthropology, psychology and cultural studies. They use standard music informatics programmes and produce their own music. In consultation with the lecturer, the students select project topics and work on these over a semester as a project group. They research the state of the art and science, formulate a development goal and work out the required skillset. They use current project management methods and tools. They implement selected parts of the modelling into functioning software. They present results in both academic and musical formats.

3 Contents

Music informatics uses methods and findings from various fields of computer science, physics, mathematics and cultural studies. The application of such methods is the main content of the course.
The following topics are examples of possible content:

- Mathematical Foundations of Music
- Physical Fundamentals of Music
- Analogue and digital sound generators
- Audio digitisation and audio formats
- MIDI
- Virtual instruments and VST
- Digital sound processing and alteration
- Special audio programming languages
- Audio libraries for all-purpose programming languages, esp. C/C++
- Agogics and the human factor
- Music as a universal human phenomenon
- Psychoacoustics and musical enjoyment
- DAW programming

4 Participation requirements

None

5 Form of assessment

	Performance exam, oral exam or term paper or project work or internship, excursion or daily log or portfolio or learning diary or parcour exam
6	**Condition for the award of credit points**
	Practical course with test Passed module examination
7	**Application of the module** (in the following study programmes)**:**
	Computer Science (B.Sc.)
8	**Module coordinator**
	Prof. Dr Dominic Becking
9	**Other information**
	References: Current journals and proceedings on the topic. Steppat, M.: Audioprogrammierung. Hanser, München, 2014. Boulanger, R., Lazzarini, V. (Hgg.): The Audio Programming Book. MIT Press, Cambridge USA, 2011. Mazzola, G.: Elemente der Musikinformatik. Birkhäuser, Basel, 2006. Loy, G.: Musimathics – the mathematical foundations of music, Vol. 1 u. 2. MIT Press, Cambridge USA, 2007. Gouveia, D.: Getting Started with C++ Audio Programming for Game Development. Packt Publishing, Birmingham, 2013. Brown, A. R.: Making Music with Java. o.O., 2005

Compulsory Elective Module from List 1 "Methods in Computer Science" **Functional Programming**								**Abbr.** **FP**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.12	**150 h**	**5**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical course	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

Functional programming is an important programming paradigm alongside object-oriented programming and distributed/parallel programming. Concepts from functional programming such as lambda expressions are gradually finding their way into modern languages such as Java9 and C#.
The course introduces the concepts of functional programming and shows the implementation in the example languages Haskell and Scala.
The students master important concepts of functional programming and can apply them using the Haskell and Scala programming languages. They recognise functional concepts in other modern programming languages and can apply them in a results-oriented manner.

3 Contents

Selection of topics for lecture:

- Functions and operators
- Lambda notation
- Higher order functions: map, filter, reduce, zip
- Functional composition and currying
- Data structures
- (Algebraic) types and type classes, polymorphism, pattern matching
- Functors and monads, combinatorial libraries
- Treatment of optional values
- Evaluation strategies, laziness
- Modularisation and interfaces
- Calculability and lambda calculus
- Type inference systems
- Introduction to the programming languages Haskell and Scala

4 Participation requirements

None

5 Form of assessment

Oral examination or written exam

6 Condition for the award of credit points

Testate and passed module examination

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr.-Ing. Carsten Gips

9 Other information

- Pepper, Hofstedt: "Funktionale Programmierung", Springer, 2006
- Jeuring, Peyton-Jones: "Advanced Functional Programming", Springer, 2009
- Block, Neumann: "Haskell Intensivkurs", Springer, 2011
- Lipovaca, M.: "Learn You a Haskell", No Starch Press, 2011
- Horstmann, C.: "Scala for the Impatient", Addison Wesley, 2012
- Odersky, M.: "Programming in Scala", Artima, 2011

Module catalogue for Computer Science (B.Sc.) of the Faculty of Minden Campus

Compulsory Elective Module from List 1: Cloud Computing								**Abbr. CC**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.13	**150 h**	**5**	5th sem.	Annual	Winter	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Seminar lessons	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

- The students know commercial and open source solutions for the development of cloud computing applications and can apply them.
- Students learn to describe established algorithms for the scaling provision of distributed system resources, taking into account various performance criteria.
- Students learn to design, implement and use a complex, distributed system for parallel processing of large amounts of data.

3 Contents

- Commercial and open source cloud computing technologies in the field of IaaS (e.g. AWS, Azure, Eucalyptus), PaaS (e.g. Heroku, EC2), SaaS (e.g. Cloudgene.
- System architectures, web and data service topologies.
- Protocols, patterns and standards.
- Big Data Analytics and Parallelisation (e.g. Hadoop, BigQuery, Storm).
- Fog / Edge Computing applications.
- Resource management under flexible performance criteria.
- Stochastic methods in the context of load balancing and elastic computing.
- Monitoring and anomaly detection.

4 Participation requirements

Formally: -, Content: Knowledge of databases, software engineering, web-based applications, distributed systems

5 Form of assessment

Project/seminar paper, oral exam or written exam.

6 Condition for the award of credit points

Practical course with test and passed module examination

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr Jörg Brunsmann

9 Other information

Compulsory Elective Module from List 1 "Methods in Computer Science" Artificial Intelligence								Abbr. KI
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.14	**150 h**	**5**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical course	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

The course introduces the principles and basic methods and algorithms of modern artificial intelligence and their application in intelligent systems. With the successful participation in the lecture and the practical courses, competences for the selection and application of basic methods of artificial intelligence for a concrete problem are acquired. Participants are able to apply the methods they have learnt to other areas and problems.

3 Contents

Selection of topics for lecture:

- Intelligent agents
- Problem solving:
  - Informed and uninformed search (depth first search, breadth first search, branch-and-bound search, best first search, A* search)
  - Local search (gradient search, simulated annealing)
  - Genetic and Evolutionary Algorithms
  - Constraint satisfaction problems, backtracking search with heuristics, constraint propagation and AC3 (edge consistency)
  - Games (minimax algorithm, alpha-beta pruning, heuristics)
- Knowledge representation and reasoning:
  - Propositional logic
  - Predicate logic, syntax and semantics, models
  - Unification, normal form, resolution calculus
  - Logical Programming (Prolog)
  - Closing with uncertainty, Bayes' rule, Bayesian networks
  - Semantic networks and ontologies
  - Planning, Situation Calculus, STRIPS
- Machine learning:
  - Decision tree procedures (CAL2, CAL3, ID3, C4.5)
  - Neural networks: Perceptron, MLP, Delta Rule, Backpropagation
  - Support Vector Machines
  - (Text) Classification with Naive Bayes, Fundamentals of Probability Theory
  - Generalisation, overfitting, cross-validation, regularisation, boosting
  - Reinforcement Learning
  - Unsupervised learning: RBF, kNN
- Fuzzy text search, similarities between texts
- Recommendation systems
- Autonomous mobile systems
- Changing contents of the homework sheets/assignments on current topics

4 Participation requirements

None

5 Form of assessment

	Oral examination or written exam

6	**Condition for the award of credit points** Testate and passed module examination
7	**Application of the module** (in the following study programmes)**:** Computer Science (B.Sc.)
8	**Module coordinator** Prof. Dr.-Ing. Carsten Gips
9	**Other information** • Russel, S., Norvig, P: "Artificial Intelligence. A Modern Approach", Prentice Hall, 2014 • Ertel, W.: "Grundkurs Künstliche Intelligenz", Springer Vieweg, 2016 • Bishop, C.: "Pattern Recognition and Machine Learning", Springer, 2007 • Witten et al.: "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, 2011 • Mitchell: "Machine Learning", Mcgraw-Hill Education, 1997

Compulsory Elective Module from List 1 "Methods in Computer Science" **Computer Graphics: Computational Geometry and Modelling**								**Abbr.** **GL1**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.15	**150 h**	**5**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	45 h		15	

2 Learning outcomes / competences

Building on the acquired basics of the lecture Computer Graphics, the students can apply methods from parts of the fields of computational geometry and modelling. Students can analyse the problems posed and select the appropriate method from the field of computer graphics

3 Contents

- Algorithms and data structures: Methods from computational geometry.
- Geometric modelling.

4 Participation requirements

Formal: -

Content: Knowledge from module Computer Graphics (CG)

5 Form of assessment

Oral examination or written exam

6 Condition for the award of credit points

Participation in the practical course with a test,
passed module examination

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Kerstin Müller

9 Other information

- Gerald Farin:
  Curves and Surfaces for CAGD: A Practical Guide,
  Morgan Kaufmann
- de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.:
  Computational Geometry -Algorithms and Applications, Springer

Compulsory Elective Module from List 1 "Methods in Computer Science" **Computer Graphics: Image Synthesis**								**Abbr.** **VL1**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.16	**150 h**	**5**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	
	Practical / Seminar	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

Building on the acquired basics of the lecture Computer Graphics, students can apply methods from parts of the fields of visualisation and global illumination. The students can analyse the problems posed and choose the appropriate procedures from the field of computer graphics.

3 Contents

- Global lighting methods: Ray tracing, radiosity.
- Visualisation.

4 Participation requirements

Formal: -

Content: Knowledge from module Computer Graphics (CG)

5 Form of assessment

Oral examination or written exam

6 Condition for the award of credit points

Participation in the practical course with a test, passed module examination

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Kerstin Müller

9 Other information

- Hearn D., Baker M.P.:
  Computer Graphics with OpenGL,
  Pearson International Edition.
- Foley J., van Dam A., Feiner S., Hughes J.:
  Computer Graphics - Principles and Practice,
  Addison-Wesley

Compulsory Elective Module from List 1 "Methods in Computer Science" Pattern Recognition & Image Processing								**Abbr. MMS**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.17	**150 h**	**5**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	90 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h			15	German

2 Learning outcomes / competences

The students know the essential methods of pattern recognition and image processing. In particular, students understand the interplay of image processing methods, feature extraction, pattern recognition, classification, machine learning to extract information from digital images. They can implement solutions for computer-based analysis of images in the field of machine vision, for example in the areas of computer vision and robot vision.
In the practical course, students learn how to develop their own image processing programmes as well as how to work with image processing software.

3 Contents

- Overview and areas of application
- Image acquisition
- Image representation and compression
- Image enhancement and filters
- Segmentation
- Registration
- Feature extraction and reduction
- Classification methods / machine learning
- Clustering

4 Participation requirements

Formal: -
Content: Knowledge of analysis, linear algebra and C++ programming

5 Form of assessment

Oral examination or written exam

6 Condition for the award of credit points

Practical course with test
Passed module examination

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Dr.-Ing. Matthias König

9 Other information

Literature will be announced in the course.

Compulsory Elective Module from List 1 "Methods in Computer Science" **Practical Aspects of Computer Security**								**Abbr.** **PIS**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.18	**150 h**	**5**	6th sem.	Annual according to demand	Summer	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

Expertise:
Students have an in-depth understanding of the modus operandi of attackers against IT systems and networks and of concrete attacks and dangers on the Internet. They are able to assess protective measures as well as to participate in the implementation of such protective measures.

Methodological competence:
Students can recognise attacks, describe, structure and classify the phases of an attack and outline and apply suitable protective measures. In addition, students can evaluate the suitability of (protective) measures and apply the measures.

Social competence: Due to the teamwork, among other things in the practical tasks, the students are able to develop solutions in the group and solve tasks cooperatively.

3 Contents

- Attackers' procedure models
- Attack method (local/remote)
- Protective measures
- Typical attacks on systems
  - Attacks on weaknesses in protocols
  - Attacks on the configuration of systems
  - Attacks on web applications
  - Special monitoring or attack programmes
- Attack detection
- Intrusion detection systems

4 Participation requirements

Formal: -
Content: Basic knowledge from Mathematics for Computer Scientists I and II, programming skills, computer engineering, operating systems, databases, web technologies

5 Form of assessment

Oral examination or project work

6 Condition for the award of credit points

Passing the module examination.
Section 17 "Progress Regulation" of the Examination Regulations for Computer Science (B.Sc.) applies.

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr Christoph Thiel

9 Other information

Compulsory Elective Module from List 1 "Methods in Computer Science" Security and Dependability								Abbr. BSZ
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.19	**150 h**	**5**	5th sem.	Annual according to demand	Winter	1 sem.	Compulsory elective	B.Sc.

1	Course type	Contact hours	Self-study	Forms of teaching (forms of learning)	Planned group size	Language
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical / Seminar	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

Expertise:
The students understand the connections and differences between the safety terms Dependability, Safety and Security. They are familiar with typical vulnerabilities and threats and know suitable countermeasures and mechanisms to increase reliability and security.

Methodological competence:
Students can make initial assessments of the safety and reliability of systems and software, evaluate the possibilities and limits of solutions and propose possible improvements.

Social competence: Due to the teamwork, among other things in the practical tasks, the students are able to develop solutions in the group and solve tasks cooperatively.

3 Contents

- Basic concepts and problems of safety and reliability;
- Reliability and security objectives (confidentiality, integrity, availability, safety, maintainability, ...);
- Vulnerability, threat, impact and risk analyses;
- Basic forms of addressing risks
- Design and structure of risk treatment plans
- Measures and mechanisms to increase the reliability and security of software and systems (cryptography, authentication, access control, protocols, Firewalls, etc.)

4 Participation requirements

Formal: -
Content: Knowledge from Mathematics for Computer Scientists I and II, programming skills, technical computer science, operating systems, databases, web technologies

5 Form of assessment

Oral examination or written exam or project work

6 Condition for the award of credit points

Passing the module examination.
Section 17 "Progress Regulation" of the Examination Regulations for Computer Science (B.Sc.) applies.

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr Christoph Thiel

9 Other information

Compulsory Elective Module from List 1 "Methods of Computer Science" **Advanced Programming Techniques**								**Abbr.** **SM**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
5.20	**150 h**	**5**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical course	2 SCH/30 h	45 h		15	German

2 Learning outcomes / competences

Participants will get an overview of different programming paradigms and language concepts as well as their fields of application. They get to know selected concepts and their implementation in modern programming languages in more detail and understand the structure and principles of programming languages. Participants are enabled to learn and classify new language constructs independently. After successful completion of the module, they will be able to choose a suitable paradigm and select and use the appropriate programming language in a solution-oriented way.

3 Contents

Selection of topics for lecture:

- Overview of programming paradigms and concepts
- Deepening understanding of selected paradigms
  - Object-oriented programming (Ruby: duck typing, open classes, mixins, metaprogramming)
  - Functional programming (Haskell: lambda notation, currying, higher order functions, data types/type classes, monads)
  - Hybrid object-oriented and functional programming (Scala: list comprehensions, traits, companion objects, pattern matching, option data type)
  - Logical programming (Prolog: unification, resolution, recursion, lists: head, tail, accumulators; cuts)
  - Concurrent/parallel programming (for example Erlang)
- Consolidation of selected concepts
  - Calculability and lambda calculus
  - Evaluation strategies, lazyness
  - Algebraic types with pattern matching
  - Combinatorial libraries, functors and monads
  - Concurrency on the JVM using actuators in Akka
  - Handling of optional or null/nil values
- DSL and tools (Xtext/Xtend, Eclipse plugins, Antlr)
- Model-based development (UML; Eclipse EMF/GMF, ...)
- Model-based development (Matlab/Simulink)

4 Participation requirements

None

5 Form of assessment

Oral examination or written exam

6 Condition for the award of credit points

Testate and passed module examination

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8	**Module coordinator** Prof. Dr.-Ing. Carsten Gips
9	**Other information**

- Tate, B.A.: "Seven Languages in Seven Weeks", Pragmatic Bookshelf Inc., 2010
- Scott, M.L.: "Programming Language Pragmatics", Morgan Kaufmann, 2009
- Lipovaca, M.: "Learn You a Haskell", No Starch Press, 2011
- Block, Neumann: "Haskell Intensivkurs", Springer, 2011
- Horstmann, C.: "Scala for the Impatient", Addison Wesley, 2012
- Odersky, M.: "Programming in Scala", Artima, 2011
- Subramaniam, V.: "Programming Groovy 2", O'Reilly, 2013
- Thomas, Hunt: "Programming Ruby", O'Reilly, 2013
- Voelter, M.: "DSL Engineering: Designing, Implementing and Using Domain-Specific Languages", CreateSpace Independent Publishing Platform, 2013
- Bettini, L.: "Implementing Domain-Specific Languages with Xtext and Xtend", PACKT Publishing, 2013
- Pepper, Hofstedt: "Funktionale Programmierung", Springer, 2006
- Johan Jeuring, Simon Peyton Jones: "Advanced Functional Programming", Springer, 2009

Compulsory Elective Module from List 2 "Applications of Computer Science" **Computer Graphics: CAGD and Efficient Data Structures**								**Abbr.** **CG2**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.20	**450 h**	**15**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	
	Practical / Seminar	4 SCH/ 30 h	315 h		15	German

2 Learning outcomes / competences

Building on the basics acquired from the lecture in Computer Graphics, students work on current problems in computer graphics and are able to evaluate suitable methods and select current tools from the field of computer graphics.

3 Contents

The topics come from current research areas and research cooperations with industrial partners in the following areas:

- Efficient data structures of computer graphics.
- Computer Aided Geometric Design

The tasks are worked on in a team; interdisciplinary projects are also possible within the framework of this compulsory elective module.

4 Participation requirements

Formal: -

Content: Knowledge from module Computer Graphics (CG)

5 Form of assessment

Project work

6 Condition for the award of credit points

Successful completion of the project work

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Kerstin Müller

9 Other information

- Gerald Farin:
  Curves and Surfaces for CAGD: A Practical Guide,
  Morgan Kaufmann
- de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry - Algorithms and Applications,
  Springer

Compulsory Elective Module from List 2 "Applications of Computer Science" Computer Graphics: Virtual Reality and Visualisation								**Abbr. VR2**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.21	**450 h**	**15**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	
	Practical / Seminar	4 SCH/ 30 h	315 h		15	German

2	**Learning outcomes / competences** Building on the basics acquired from the lecture in Computer Graphics, students work on current problems in computer graphics and are able to evaluate appropriate methods and select current tools from computer graphics
3	**Contents** The topics come from current research areas and research cooperations with industrial partners in the following areas: • Visualisation techniques: Methods for optimised presentation. • Usability and information visualisation. • (Serious) games. • Virtual reality applications. The tasks are worked on in a team; interdisciplinary projects are also possible within the framework of this compulsory elective module.
4	**Participation requirements** Formal: - Content: Knowledge from module Computer Graphics (CG)
5	**Form of assessment** project work
6	**Condition for the award of credit points** Successful completion of the project work
7	**Application of the module** (in the following study programmes)**:** Computer Science (B.Sc.)
8	**Module coordinator** Prof. Dr. Kerstin Müller
9	**Other information** • Hearn D., Baker M.P.: Computer Graphics with OpenGL, Pearson International Edition. • Foley J., van Dam A., Feiner S., Hughes J.: Computer Graphics - Principles and Practice, Addison-Wesley

Compulsory Elective Module from List 2 "Applications of Computer Science" Computer Art, Social Computing and New Communications Algorithms								**Abbr. CSK**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.22	**450 h**	**15**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	
	Practical / Seminar	4 SCH/ 30 h	315 h		15	German

2	**Learning outcomes / competences** Students have an overview of current techniques and applications of computer art. They design and implement and/or integrate interaction techniques and hardware of the interaction between social groups and software systems and become familiar with design principles and realisation methods for software, even for those with large user groups. They construct new forms of communication and are familiar with corresponding new algorithms.
3	**Contents** The software applications to be created come from the application fields of media informatics, visualisation, computer vision, music informatics, AI, game theory, robotics, art, psychology and sociology. An interdisciplinary orientation is desirable. Students will design and implement creative, new forms of communication, action scenarios and interaction mechanisms between humans and machines or between humans and machines as communication partners. Aspects of large scale and mass interaction can be included.
4	**Participation requirements** None
5	**Form of assessment** project work
6	**Condition for the award of credit points** Successful completion of the project work
7	**Application of the module** (in the following study programmes)**:** Computer Science (B.Sc.)
8	**Module coordinator** Prof. Dr. Dominic Becking, Prof. Dr. Kerstin Müller
9	**Other information**

Compulsory Elective Module from List 2 "Applications of Computer Science" **Database Applications**								**Abbr.** **DBA**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.23	**450 h**	**15**	5th/6th sem.	Bi-annual according to demand	according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	360 h	To be announced in course	60	German
	Practical course	4 SCH/60 h			15	German

2 Learning outcomes / competences

Databases are the foundation of every large software system. The interaction of programming and database is of utmost importance. The students survey and formulate the requirements of different software system classes for databases. They design special data models for different system classes and implement them. They design and implement a software system consisting of an application programme and a coordinated database. They implement programme logic in the database with Persistent Stored Modules. The participants use various APIs to connect to User programmes on databases...

3 Contents

The following topics are examples of possible content:

- Software systems and requirements for databases
- Requirements analysis for database applications
- Object-oriented and ER modelling
- Persistent Stored Modules (SQL/PSM)
- Trigger
- Impedance Mismatch
- Cursors
- Object Relational Mappings - Object Language Bindings (SQL/OLB)
- APIs and frameworks for database access

4 Participation requirements

Formal: -
Content: Contents of Module 3.2 "Database Systems I" (DB1)

5 Form of assessment

Performance examination or oral examination or term paper or project work or Internship, excursion or daily log or portfolio or learning diary or course examination

6 Condition for the award of credit points

Practical course with test
Passed module examination

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Dominic Becking

9 Other information

Compulsory Elective Module from List 2 "Applications of Computer Science" Embedded Software								**Abbr. ESW**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.24	**450 h**	**15**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	45 h	To be announced in course	60	German
	Practical course	4 SCH/60 h	315 h		15	German

2 Learning outcomes / competences

The students know how to apply the methods of software engineering for embedded systems, taking into account the boundary conditions of these systems. They have the ability to carry out the development of software for embedded systems from planning to implementation.
To deepen their knowledge, students work independently on projects in which they develop "embedded software" for an application. In the team-oriented project work, project management and self-competences are deepened.

3 Contents

- Requirements
- Software Architecture and Design Patterns
- UML/SysML for Embedded Systems and Model-Based Development
- Embedded operating systems
- Quality assurance and standards
- Changing content of the projects on current topics, e.g.
  - Internet of Things technologies

For example, an application is developed as an in-depth project (e.g. for the Internet of Things, computer vision, robotics, measurement and control technology). In the process, hardware abstraction layers or operating systems for embedded systems (e.g. embedded Linux, RTOS) can be used as needed.

4 Participation requirements

Formal: -
Content: Knowledge of embedded systems, software engineering and C++ programming

5 Form of assessment

Project work

6 Condition for the award of credit points

Successful completion of the project work

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Dr.-Ing. Matthias König

9 Other information

Literature will be announced in the course.

Compulsory Elective Module from List 2 "Applications of Computer Science" Internet Security								Abbr. BIS
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.25	**450 h**	**15**	5th sem.	Annual according to demand	Winter	1 sem.	Compulsory elective	B.Sc.

1	Course type	Contact hours	Self-study	Forms of teaching (forms of learning)	Planned group size	Language
	Lecture	2 SCH/30 h	60 h	To be announced in course	60	German
	Practical / Seminar	4 SCH/60 h	300 h		15	German

2 Learning outcomes / competences

Expertise:
The students are familiar with the most important basic technologies for securing networks. They demonstrate an in-depth understanding of security mechanisms at the different protocol layers (application layer, transport layer, network layer, link layer, physical layer) and know the structure, principles, architecture and functioning of security components and systems in the field of internet security.
They are able to explain in detail the characteristics and basic principles of the problem space of internet security and demonstrate a sound knowledge of practice and theory in this field.

Furthermore, they are familiar with current developments in the field of internet security and can explain them (e.g. security in peer-to-peer systems, security in mobile networks, security in cloud computing, block chains, etc.).

Methodological competence:
Students will be able to apply the basics of IT security and cryptography to the field of communication networks and thus develop and evaluate solutions for internet security.

Social competence: Due to the teamwork, among other things in the practical tasks, the students are able to develop solutions in the group and solve tasks cooperatively.

3 Contents

- Internet security: Introduction, motivation and challenges
- Basic feature: Reference model for network security, security standards for networks and the Internet, threats, attacks, security services and mechanisms
- Cryptographic basics for securing networks: symmetric cryptography and asymmetric cryptography and their application in networks, supporting mechanisms for the implementation of security solutions, public key infrastructures
- Security at the different protocol layers (application layer, transport layer, network layer, link layer, physical layer)
- Applied Internet Security: Firewalls, Intrusion Detection Systems, Identity Management
- Selected topics in internet security: Security for Distributed Systems, Security for Web Applications and Web Services, Security for Cloud Computing
- Changing content of the projects on current topics
- Replication of attack scenarios and countermeasures in the laboratory

4 Participation requirements

Formal: -
Content: Knowledge of Java or C++ programming, basic knowledge of technical computer science, distributed systems and communication networks

5 Form of assessment

Oral examination or written exam or project work

6	**Condition for the award of credit points** Passing the module examination. Section 17 "Progress Regulation" of the Examination Regulations for Computer Science (B.Sc.) applies.
7	**Application of the module** (in the following study programmes)**:** Computer Science (B.Sc.)
8	**Module coordinator** Prof. Dr Christoph Thiel
9	**Other information** Literature: • Eckert, C.: IT security: Konzepte - Verfahren, Oldenbourg Wissenschaftlicher Verlag; ISBN: 978-3-486-72138-6, 8th edition 2013. • Schwenk, Jörg: Security and cryptography on the internet: Von Sicherer E-Mail bis zu IP- Verschlüsselung (German Edition), Vieweg+Teubner Verlag ISBN: 978-3834808141 3rd ed. 2010 • Stallings ,William; Network Security Essentials, 4th Edition, Prentice Hall, ISBN: 978-0-136-10805-9, 2010, • Current professional articles

Compulsory Elective Module from List 2 "Applications of Computer Science" **Mobile Applications**								**Abbr.** MOB
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.26	**450 h**	**15**	5th/6th sem.	Bi-annual	Summer/ winter	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	60 h	To be announced in course	60	German
	Practical / Seminar	4 SCH/60 h	300 h		15	German

2 Learning outcomes / competences

Software development for mobile devices is the further development of traditional (application) software development towards an open, distributed and dynamic technology that combines mobile, powerful devices with traditional structures of the internet and distributed software systems.

The aim of this module is to enable students to

- understand the special features and boundary conditions as well as concepts and solution approaches of mobile application development and also be able to estimate them quantitatively,
- get to know system architecture and application development solutions tailored to this,
- and be able to apply this knowledge practically to solve concrete tasks.

Students receive an overview of current hardware platforms and mobile operating systems and learn to master a current system (for example Android). After attending the course, they will be able to develop their own mobile applications, taking usability, energy and security aspects into account.
To deepen their knowledge, students work independently on projects and deepen their project management and personal skills during the team-oriented project work.

3 Contents

- Mobile application technology, hardware, operating systems
- Applications & Application Development
- Current software frameworks for mobile applications
- Use of hardware components of mobile devices
- Integration of existing sensors and interfaces
- Changing content of the projects on current topics, e.g.
  - Resource management in mobile systems and energy aspects
  - Security for mobile systems and applications

4 Participation requirements

Formally: -, Content: Knowledge of programming, software engineering

5 Form of assessment

project work

6 Condition for the award of credit points

Successful completion of the project work

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr. Martin Hoffmann (covered by Prof. Dr. Dr.-Ing. Matthias König)

9 Other information

References: Thomas Künneth: Android 7 - Das Praxisbuch für Entwickler, Rheinwerk Verlag 2017

Compulsory Elective Module from List 2: Full Stack Development								**Abbr. FSD**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.27	**450 h**	**15**	6th sem.	Annual	Summer	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Seminar lessons Practical / Seminar	2 SCH/30 h 4 SCH/60 h	60h 300h	To be announced in course	60 15	German German

2 Learning outcomes / competences

- Students learn about current architectures, technologies and tools for the development of full stack applications for different end devices and can apply them.
- Students learn to independently find suitable solutions for practical problems and to implement and test them in a targeted manner using software tools.
- Students learn to make use-case specific technology decisions for front-end, back-end and database components of distributed applications.
- They will learn about architecture patterns for the appropriate division of large software projects and be able to apply them to software projects.

3 Contents

- Basics, motivation, overview, requirements.
- Design patterns, architectures, frameworks.
- Microservices and monolithic server architectures.
- Planning and development of scalable and secure software components.
- Responsive single-page applications.
- Modern programming languages (e.g. ECMAScript, Go, Rust).
- Front-end technologies (e.g. Angular, React, Vue).
- Backend technologies (e.g. Node.js, Django).
- Relational and No-SQL databases (e.g. PostgreSQL, MongoDB).

4 Participation requirements

Formally: -, Content: Knowledge of object-oriented programming, databases, software engineering, web-based applications, cloud computing

5 Form of assessment

project work

6 Condition for the award of credit points

Successful completion of the project work

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr Jörg Brunsmann

9 Other information

Compulsory Elective Module from List 2 "Applications of Computer Science" **Software Quality**								**Abbr. SQ**
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.28	**450 h**	**15**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	60 h	To be announced in course	60	German
	Practical course	4 SCH/60 h	300 h		15	German

2 Learning outcomes / competences

The course introduces techniques and methods for software quality assurance in the development and maintenance of software systems. Successful participation in the course will provide an understanding of software quality and the importance of systematic software testing. The participants know the general test process and the roles involved. They know different test levels and types and are able to select and use different static and dynamic testing techniques and tools according to requirements. Participants will be able to develop higher quality software using the methods learned.

The lecture serves to convey basic theoretical knowledge and skills, with practical application being learned and deepened in the accompanying project. Through the team-oriented project work, the participants' project management and self-competences are deepened.

After successful participation in this module, it is optionally possible to take an examination for "Certified Tester - Foundation Level" according to ISTQB at an examination institute certified by the German Testing Board.

3 Contents

Selection of topics for lecture:

- Quality aspects of software systems
- Basics of software testing, test principles, fundamental test process
- Testing in the software life cycle, test levels and types
- Static testing techniques: Reviews, static analysis
- Dynamic test techniques, test design
  - Specification-based (black box): Equivalence partitioning and boundary value analysis, decision tables, state-based test, other black-box design methods
  - Structure-based (white-box): Coverage (C0, C1, C2, C3), control flow anomalies and data flow anomalies
  - Experience-based test: Error Guessing, Exploratory Testing
- Test concept, test strategy, test management
- Tools
- Test automation
- Test case generation
- Changing content of the projects on current topics

4 Participation requirements

None

5 Form of assessment

Project work

6	Condition for the award of credit points

	Module examination pass
7	**Application of the module** (in the following study programmes)**:** Computer Science (B.Sc.)
8	**Module coordinator** Prof. Dr.-Ing. Carsten Gips
9	**Other information** • Spillner, A., Linz, T.: "Basiswissen Softwaretest", dpunkt-Verlag, 2012 • Kleuker, S.: "Qualitätssicherung durch Softwaretests", Springer Vieweg, 2013 • Liggesmeyer, P.: "Software-Qualität", Springer Spektrum, 2009 • Klaus Franz: "Handbuch zum Testen von Web- und Mobile-Apps", Springer Vieweg, 2014 • Robert Martin: "Clean Code", Prentice Hall, 2008 • Michael Feathers: "Working Effectively with Legacy Code", Prentice Hall, 2013 • Roy Osherove: "The Art of Unit Testing", Manning, 2013 • Gerard Meszaros: "xUnit Test Patterns", Addison Wesley, 2007 • Kent Beck: "Test Driven Development", Addison-Wesley, 2002 • Graham et al.: "Foundations of Software Testing", Cengage Learning, 2012 • Myers, G.J.: "The Art of Software Testing", John Wiley, 2011

Compulsory Elective Module from List 2 "Applications of Computer Science" Webengineering								**Abbr.** WE
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.29	**450 h**	**15**	5th/6th sem.	Bi-annual according to demand	Summer/ winter at demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	60 h	To be announced in course	60	German
	Practical course	4 SCH/60 h	300 h		15	German

2 Learning outcomes / competences

Understanding of current concepts, methods, techniques, tools and experiences for the engineering development of web applications as well as their practical application in own project work in web development teams.
• Evaluate potential risks of web applications
• Ability to follow and assess future developments in the field of web engineering
The skills acquired thus contribute in particular to the development of specific analysis, design and realisation competences and to the expansion of specific technological competences. Team-oriented project work also increases
Project management and self-competences.

3 Contents

Introduction to web engineering (motivation, definition, basic principles)
• Product development
• Requirements engineering for web applications
• Modelling of web applications
• Web application architecture
• Testing web applications
• Web project management
• Quality aspects (usability, performance, security)
• Java-based web frameworks e.g. JSF with Primefaces, Richfaces and JPA
• Web applications with JavaScript and HTML5
  - Webpattern
  - Frameworks for Javascript e.g. Knockout JS, Angular JS, Node JS

4 Formal participation requirements: -

Content: Web-based applications, object-oriented programming, programming methodics, software engineering, databases

5 Form of assessment

project work

6 Condition for the award of credit points

Successful completion of the project work

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Prof. Dr.-Ing. Grit Behrens

9 Other information

- Kurz, Marinschek: „JavaSever Faces 2.2: Grundlagen und erweiterte Konzepte“, dpunkt 2013
- Kappel, Pröll, Reich, Teschitzegger: Web-Engineering, dpunkt 2004
- Backschat, Martin : „Enterprise JavaBeans und JPA“ Springer Spektrum 2016
- Tarasiewicz: „Angular JS - Framework“, dpunkt 2014

Compulsory Elective Module from List 2 "Applications of Computer Science" Applications in Artificial Intelligence								**Abbr.** AKI
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
6.30	**450 h**	**15**	5th/6th sem.	Bi-annual according to demand	Summer/ winter according to demand	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Lecture	2 SCH/30 h	60 h	L	60	German
	Practical course	4 SCH/60 h	300 h	P	15	German

2 Learning outcomes / competences

Understanding of current concepts, methods, techniques, tools and experiences for the engineering development of artificial intelligence applications as well as their practical application in own project work in development teams.
• Assess potential risks of artificial intelligence applications
• Ability to follow and assess future developments in the field of AI
The skills acquired thus contribute in particular to the development of specific analysis, design and realisation competences and to the expansion of specific technological competences. Team-oriented project work also increases project management and self-competences.

3 Contents

• AI application projects are created in teamwork

• Application of machine learning methods (e.g. neural networks, deep learning, support vector machine, decision trees, clustering methods)

• Sensors (video, audio, infrared camera, electroluminescence camera, weather data, indoor air parameters, characteristic measuring devices, robots, copters)

• Processing large amounts of data from research and application projects of the lecturers in teamwork

• Feature extraction with elements of image processing and language processing

• Application of libraries of modern tools for data analysis and machine learning (e.g Python, NumPy, Pandas, SciPy, Jupyter, IPython or WEKA or KNIME)

The focus is largely on the independent processing of a complex task within the framework of a development project in a team, which can also be processed in cooperation with research and development departments of companies. As a rule, the project groups consist of 2–4 students who are to come together freely, choose a project leader from among themselves and develop according to procedural models agreed with the lecturers. The lecturer defines the objective and conducts a regular discourse on the progress of the AI project

4 Formal participation requirements: -

Content: Artificial Intelligence (Subject List 1- Methods in Computer Science), Object-Oriented Programming, Databases

5 Form of assessment

project work

6 Condition for the award of credit points

	Successful completion of the project work

7	**Application of the module** (in the following study programmes)**:** Computer Science (B.Sc.)
8	**Module coordinator** Prof. Dr.-Ing. Grit Behrens
9	**Other information** Ian H. Witten „Data Mining: Practical Machine Learning Tools and Techniques“, ELSEWVIER 2017, ISBN 978-0128042915 Thomas Haslwanter “An Introduction to Statistics with Python”, Springer Nature 2016, ISBN 978-3-319-28316-6 Miroslav Kubat “An Introdruction to Machine Learning”, Springer Nature 2017, ISBN 978-3-319-63912-3

Project in Industry								**Abbr.** PRA
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
7.0	**450 h**	**18**	7th sem.	Annual	Winter	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	Work at an individual practice site	450 h	-	Working activity	1	German

2 Learning outcomes / competences

The work term offers students the opportunity to try out in practice the professional skills they have acquired in several semesters at the university and to acquire additional important competences in the extracurricular area. It therefore plays a central role within the framework of a practice-oriented and labour market-oriented education as well as for personality development. The learning outcomes include:

- Orientation in the desired occupational field
- Acquisition of practical knowledge and familiarisation with typical professional working methods
- Getting to know technical and organisational contexts that are typical for the occupational field.
- Participation in the work process according to the level of training
- Practical training on clearly defined, concrete projects

3 Contents

Practical activity with a clear computer science focus, e.g.

- Carrying out requirements analyses, creating software design
- Programming
- Database design and implementation
- Realisation of web applications
- Network planning, security analyses
- Processing of graphic data, visualisation

Framework conditions:

- Shorter daily working hours than half a day are not permitted.
- The entire internship must be completed within 9 months.
- Sick leave and other periods of absence do not count.
- The professional internship must be completed at an internship institution.
- A report of 13 to 20 pages is to be prepared on the professional internship.
- The internship is supervised by a university faculty tutor.

4 Participation requirements

Formal: 110 cps (s. BPO)

Content:

5 Form of assessment

Evaluated practical report as a certificate and report from the training institution (see BPO)

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Lecturers in the computer science study programme (Becking, Behrens, Brunsmann, George, Gips, Hoffmann, König, Kreienkamp, Müller, Thiel)

9 Other information

Working materials and literature correspond to the individual assignment

Bachelor Thesis								**Abbr.** BA
No.	**Workload**	**Credit points**	**Study semester**	**Frequency**	**Sem.**	**Duration**	**Type**	**Q level**
7.1	**360 h**	**12**	7th sem.	Annual	Winter	1 sem.	Compulsory elective	B.Sc.

1	**Course type**	**Contact hours**	**Self-study**	**Forms of teaching (forms of learning)**	**Planned group size**	**Language**
	0.3 SCH Individual lecturer-based faculty tutoring	10 h	350 h	Individual lecturer-based faculty tutoring Independent preparation of the bachelor thesis	1	German

2 Learning outcomes / competences

The independent solution of a practice-related topic according to scientific principles is one of the professional skills of a computer scientist. The systematic processing and practice-related implementation of a task as well as the coherent presentation of reports and publications serves as communication between experts and ensures that acquired knowledge and skills are retained. Students learn how to methodically process a task and present it with a clearly structured result in a given time frame by quickly familiarising themselves with a new task and independently deepening their knowledge in a specific area. Students learn to use common tools and methods for work support, apply a range of subject-specific skills, abilities and techniques in order to solve tasks independently, to analyse and evaluate them and to present them in an overall context.

3 Contents

The bachelor thesis is intended to prove that students are able to solve a complex problem using scientific methods within a limited period of time and to document the theoretical and practical knowledge acquired in a comprehensible manner.

1. Define the concrete details of a task
2. Preparation of a timetable
3. Evaluation and listing of the techniques and methods to be used
4. Creation of a software concept
5. Implementation and documentation of the software solution
6. Overall view, test and evaluation of the solution
7. Presentation of the solution in the form of the bachelor thesis

4 Participation requirements

Formal: Passed module examinations according to the course schedule except for four module examinations (see also BPO).
Content: Knowledge in the breadth of the subject studied

5 Form of assessment

Bachelor thesis assessed by two examiners

6 Condition for the award of credit points

Module examination pass

7 Application of the module (in the following study programmes)**:**

Computer Science (B.Sc.)

8 Module coordinator

Lecturers in the computer science study programme (Becking, Behrens, Brunsmann, George, Gips, Hoffmann, König, Kreienkamp, Müller, Thiel)

9 Other information

Working materials and literature correspond to the individual assignment