JACOBS UNIVERSITY

Study Program Handbook

Computer Science and Software Engineering

Bachelor of Science

Subject-specific Examination Regulations for Computer Science (Fachspezifische Prüfungsordnung)

The subject-specific examination regulations for Computer Science and Software Engineering are defined by this program handbook and are valid only in combination with the General Examination Regulations for Undergraduate degree programs (General Examination Regulations = Rahmenprüfungsordnung). This handbook also contains the program-specific Study and Examination Plan (Chapter 6).

Upon graduation, students in this program will receive a Bachelor of Science (BSc) degree with a scope of 180 ECTS (for specifics see Chapter 6 of this handbook).

Version	Valid as of	Decision	Details
Fall 2022 – V1	Sep 01, 2022	June 22, 2022	V1 Approved by the Academic Senate
		July 5, 2022	V1.1 Editorial Changes / Error corrections in Chapters 1 and 2, as well as the Study & Examination Plan
		Aug 18, 2022	V1.2 Changes in "Admission Requirements" and "Internship / Startup and Career Skills"
		Sep 19, 2022	V1.3 Editorial change uniform specialization module text

Contents

1 Program Overview 6

1.1 Concept 6

1.1.1 The Jacobs University Educational Concept 6

1.1.2 Program Concept 6

1.2 Specific Advantages of Computer Science and Software Engineering at Jacobs University 7

1.3 Program-specific Educational Aims 8

1.3.1 Qualification Aims 8

1.3.2 Intended Learning Outcomes 9

1.4 Career Options 10

1.5 Admission Requirements 10

1.6 More Information and Contact 11

2 The Curricular Structure 12

2.1 General 12

2.2 The Curriculum 12

2.2.1 Year 1 12

2.2.2 Year 2 13

2.2.3 Year 3 13

3 Computer Science and Software Engineering Undergraduate Program Regulations 16

3.1 Scope of these Regulations 16

3.2 Degree 16

3.3 Graduation Requirements 16

4 Schematic Study Plan for Computer Science and Software Engineering 17

5 Study and Examination Plan 18

6 Module Descriptions 20

6.1 YEAR 1 20

6.1.1 Introduction to Computer Science 20

6.1.2 Programming in C and C++ 22

6.1.3 Introduction to Data Science 24

6.1.4 Calculus and Elements of Linear Algebra I 26

6.1.5 Algorithms and Data Structures 29

6.1.6 Introduction to Cyber Physical Systems 31

6.1.7 Software Design and Prototyping 33

6.1.8 Calculus and Elements of Linear Algebra II ... 35
6.1.9 Distributed Development ... 37
6.2 YEAR 2 ... 39
6.2.1 Databases and Web Services ... 39
6.2.2 Operating Systems ... 41
6.2.3 Data Analytics and Modeling ... 43
6.2.4 Probability and Random Processes ... 45
6.2.5 Software Engineering ... 47
6.2.6 Artificial Intelligence (CSSE) ... 49
6.2.7 Machine Learning ... 51
6.2.8 Machine Learning Tools ... 53
6.3 YEAR 3 ... 55
6.3.1 Computer Graphics ... 55
6.3.2 Computer Networks ... 57
6.3.3 Web Application Development ... 59
6.3.4 Human-Computer Interaction ... 61
6.4 Internship / Startup and Career Skills ... 63
6.5 Collaborative Software Project ... 66
6.6 Bachelor Thesis ... 67
7 Appendix ... 69
7.1 Intended Learning Outcomes Assessment Matrix ... 69

1 Program Overview

1.1 Concept

1.1.1 The Jacobs University Educational Concept

Jacobs University aims to educate students for both an academic and a professional career by emphasizing four core objectives: academic quality, self-development/personal growth, internationality and the ability to succeed in the working world (employability). Hence, study programs at Jacobs University offer a comprehensive, structured approach to prepare students for graduate education as well as career success by combining disciplinary depth and interdisciplinary breadth with supplemental skills education.

In this context, it is Jacobs University's aim to educate talented young people from all over the world, regardless of nationality, religion, and material circumstances, to become citizens of the world who are able to take responsible roles in the democratic, peaceful, and sustainable development of the societies in which they live. This is achieved through high-quality teaching, manageable study loads, and supportive study conditions. Study programs, taught online or in presence, convey academic knowledge as well as the ability to interact positively with other individuals and groups in culturally diverse environments. The ability to succeed in the working world is a core objective for all study programs at Jacobs University, both in terms of actual disciplinary subject matter and also of social skills and intercultural competence. Study-program-specific modules and additional specializations provide the necessary depth, interdisciplinary offerings provide breadth while the collaborative and remote didactical approach of the modules as well as an extended internship period strengthen the employability of students. In addition, Jacobs University offers professional advising and counseling.

Jacobs University's educational concept is highly regarded both nationally and internationally. While the university has consistently achieved top marks over the last decade in Germany's most comprehensive and detailed university ranking by the Center for Higher Education (CHE), it has also been listed by the renowned Times Higher Education (THE) magazine as one of the top 300 universities worldwide (ranking group 251-300) in 2019, 2020 and 2021. The THE ranking is considered as one of the most widely observed university rankings. It is based on five major indicators: research, teaching, research impact, international orientation, and the volume of research income from industry.

1.1.2 Program Concept

Digitalization is a key driver of innovation and success across all industries. Computer Science and especially Software Engineering are obviously key elements in these processes. At the same time, there is a substantial change in the way daily work is organized and carried out. The share of home office and remote work increases, e.g., to collaborate with team members who are distributed around the world or to control, monitor, and maintain facilities and processes from a distance. While offering a lot of opportunities in terms of convenience for employees and reduced costs for employers, this new normal of working also requires different skills and knowledge of the related tools and methods, which are addressed by this program.

Furthermore, online education is changing the higher education landscape in profound ways. It caters for specific needs and interests of students, especially in terms of the flexibility in which

they can carry out their studies. And it is a natural option to prepare them for the new normal of remote work.

The bachelor program in Computer Science and Software Engineering uses online education with high amounts of flipped-classroom elements. This means that students participate in online courses with pre-dominantly asynchronous lectures and exercise material, which are complemented by tutorials and hands-on sessions (for the study cohort starting in 2022/23 participation in synchronous online lectures and tutorials with the possibility for synchronous communication are offered). Students are guided and supported by faculty as well as experienced tutors and lecturers to transfer the acquired knowledge into practice. The hands-on elements include high amounts of collaboration with other students, use of tools and concepts to engage in distributed work from different places in potentially different time-zones, and remote access to physical devices and set-ups.

The Computer Science core of the program is complemented with Management and Leadership modules in the final study year. Students will not only be trained in programming and software development, but will also acquire fundamental knowledge in business and learn how innovations can be transferred into a marketable product. Furthermore, they may take part in interdisciplinary courses in which problems are tackled from a wider perspective challenging them to think outside the boundaries of their discipline.

Overall, by completing their studies, students will be able to directly enter the job market or to continue their studies in a graduate program, for example the MSc in Computer Science and Software Engineering offered at Jacobs University. Apart from the solid knowledge and skills obtained in Computer Science and Software Engineering, graduates are particularly well prepared for the demands of modern work, i.e. to work remotely and as part of a diverse team.

1.2 Specific Advantages of Computer Science and Software Engineering at Jacobs University

The Computer Science and Software Engineering program at Jacobs University aims to provide an application-oriented knowledge of Computer Science and Software Engineering including a preparation for important aspects of modern professional life, namely remote work and life-long learning.

The educational approach of the faculty is to relate the theoretical contents of the discipline to their contemporary application in industry and research. The instructors aim to include recent developments of the topics covered to demonstrate how basic methods or techniques are applied today and how the material covered relates to the challenges of digitalization and the related state of the art in research and development.

- Early involvement in software development project work is an essential aspect of the study program which further extends the already positively acknowledged educational approach in Computer Science at Jacobs University.
- The Computer Science faculty's pedagogy, together with the positive teaching environment, has been acknowledged in several rankings: In the Computer Science ranking published by the Centre for Higher Education (CHE) in 2015, the support by instructors and the relationship to research were ranked 1st of 68 study programs. In the European U-Multirank ranking published in 2018, the overall learning experience in Computer Science

was ranked 10th and research-oriented teaching in Computer Science was ranked 2nd of 304 European universities offering Computer Science programs.

- The involvement of students and alumni in the program development process using a direct and open dialogue is going to ensure that the program will be constantly fine-tuned to the specific needs of students, such as covering certain topics at a certain time with respect to the preparation of internship or job applications.
- Computer Science student teams participate regularly in international programming competitions. Jacobs University hosted the Northwestern European Regional Contest (NWERC) of the ACM International Collegiate Programming Contest on campus in 2010 and 2011. Student teams participate in NWERC competitions since then on an annual basis. In 2014, students organized the first JacobsHack! hackathon on campus, which was sponsored, among others, by Google, Microsoft, and SAP. The 2018 edition of JacobsHack!, sponsored, among others, by Facebook, Skyscanner, GitHub and Bloomberg, attracted participants from all over Europe. As the program features important elements remote collaborative software development, there is also the option for online students to participate in according activities if they are interested in them.

1.3 Program-specific Educational Aims

1.3.1 Qualification Aims

The program is an online program with optional blended elements, e.g., in summer (for the study cohort starting in 2022/23 participation in synchronous online lectures and tutorials with the possibility for synchronous communication are offered). Lectures incorporate asynchronous material and primarily follow a flipped classroom model, i.e., including application components in the spirit of problem-based- as well as project-based-learning (PBL & PrBL). Practical components, particularly labs, projects, and thesis are based on remote access, distributed development. Tutoring includes virtual study groups, peer evaluation and mentoring by faculty. Performance evaluation are conducted as online e-exams.

The remote work aspects include collaborative software development and remote access to physical devices for, e.g., control, monitoring and maintenance. Due to the aspects of independent, self-governed knowledge acquisition, the students are prepared for life-long learning, where additional knowledge and skills need to be acquired or updated in a regular fashion, especially in fast moving areas like Computer Science and Software Engineering.

The main subject-specific qualification aim is to enable students to take up qualified employment in modern industries involving digitalization and information technology or to enter graduate programs related to Computer Science and Software Engineering. Graduates of the Computer Science and Software Engineering program have obtained the following competencies:

- Computer Science and Software Engineering competence
  Graduates are familiar with the foundations of Computer Science and they are able to design and develop software addressing a given application scenario. They are able to analyze and structure complex problems and they are able to address them using methods of Computer Science and Software Engineering. Graduates are able to construct and maintain complex computer systems using a structured, analytic, and creative approach. They are trained in developing software in collaborative teams in a remote fashion, i.e., independent of the location they live and work at.

- Communication competence
  Graduates are able to communicate subject-specific topics convincingly in both spoken and written form to fellow computer scientists or to customers.

- Teamwork and project management competence
  Graduates are able to work effectively in a remote team and they are able to organize workflows in complex development efforts. They are familiar with tools that support the development, testing, and maintenance of large software systems and they are able to take design decisions in a constructive way.

- Learning competence
  Graduates have acquired a solid foundation enabling them to assess their own knowledge and skills, learn effectively, and remain up-to-date with the latest developments in the rapidly evolving field of Computer Science and Software Engineering.

- Personal and professional competence
  Graduates are able to develop a professional profile, justify professional decisions based on theoretical and methodical knowledge, and critically reflect on their behavior with respect to their consequences for society.

- Management competence
  Graduates have obtained fundamental business and management knowledge supporting them to pursue a successful career in a corporate environment or with an own start-up idea.

1.3.2 Intended Learning Outcomes

By the end of the program, students will be able to:

1. acquire Computer Science and Software Engineering knowledge in an independent, self-governed way;
2. work in teams distributed around the globe to analyze complex problems, to evaluate them, and to derive solutions;
3. comprehend the processes and tools of Software Engineering for collaborative, remote software and systems development;
4. program software in C/C++ and understand algorithms;
5. be able to use libraries and to generate software in core Computer Science areas;
6. apply suited mathematical methods;
7. understand operating systems, databases, and web services;
8. comprehend methods from Artificial Intelligence and Machine Learning;
9. understand the relation between software and its links to the physical world;
10. analyze data and to extract insights from it;
11. apply the acquired Software Engineering skills and Computer Science knowledge in collaborative, remote projects;
12. use academic or scientific methods as appropriate in the field of Computer Science and Software Engineering such as defining research questions, justifying methods, collecting, assessing and interpreting relevant information, and drawing scientifically-founded conclusions that consider social, scientific and ethical insights;

13. develop and advance solutions to problems and arguments in their subject area and defend these in discussions with specialists and non-specialists;
14. engage ethically with academic, professional and wider communities and to actively contribute to a sustainable future, reflecting and respecting different views;
15. take responsibility for their own learning, personal and professional development and role in society, evaluating critical feedback and self-analysis;
16. apply their knowledge and understanding to a professional context;
17. take on responsibility in a diverse team;
18. adhere to and defend ethical, scientific and professional standards.

1.4 Career Options

Digitalization is affecting all areas of business, industry, daily life, and society. There is accordingly a very high demand for graduates with a background in Computer Science and Software Engineering in general. In addition, students have been trained to be able to work in a remote, collaborative fashion and being able to engage in life-long learning, i.e., to acquire or update knowledge and skills in the fast-moving areas of Computer Science and Software Engineering in an independent and self-governed way. This offers not only increased flexibility for graduates to engage in professional opportunities worldwide, it is also a substantial benefit for potential employers as they may select from an increased pool of talented candidates, whom they do not need to relocate to work on their job.

The areas of employment are almost unlimited as digitalization is important in business, industry, daily life, and society. Within these areas, research & development or management tracks can be taken. The job market includes jobs such as software engineer, information systems manager, data analyst, computer systems engineer, application developer, IT consultant, remote maintenance manager, and system analyst.

Jacobs University's Career Services Center and Alumni Association will help students in their career development. The Career Services Center provides students with high-quality training and coaching in application and interview preparation, effective presenting, business etiquette, and employer research as well as many other career aspects. It helps students select and achieve rewarding careers after their graduation from Jacobs University. In addition, the Alumni Association helps students establish a long-lasting worldwide network they can use to explore career opportunities in industry and academia.

1.5 Admission Requirements

Admission to Jacobs University is selective and based on a candidate's school and/or university achievements, recommendations, self-presentation, and performance on required standardized tests. Students admitted to Jacobs University demonstrate exceptional academic achievements, intellectual creativity, and the desire and motivation to make a difference in the world.

The following documents need to be submitted with the application:

- Recommendation Letter (optional)
- Official or certified copies of high school/university transcripts
- Educational History Form
- Standardized test results (SAT/ACT) if applicable
- Motivation statement

- ZeeMee electronic resume (optional)
- Language proficiency test results (TOEFL, IELTS or equivalent)

Formal admission requirements are subject to higher education law and are outlined in the Admission and Enrollment Policy of Jacobs University.

For more detailed information about the admission visit: https://www.jacobs-university.de/study/undergraduate/application-information

1.6 More Information and Contact

For more information, please contact the study program chair:

Prof. Dr. Andreas Birk, Study and Program Chair

Professor of Electrical Engineering & Computer Science

Email: a.birk@jacobs-university.de

Telephone: +49 421 200-3113

or visit our program website: https://www.jacobs-university.de/study/undergraduate/programs/computer-science-and-software-engineering-bsc

2 The Curricular Structure

2.1 General

The curricular structure provides multiple elements for enhancing employability, interdisciplinarity, and internationality. Additionally, a mandatory internship (or work in a start-up) of at least two months after the second year of study gives students opportunities to gain insight into the professional world, apply their intercultural competences and reflect on their roles and ambitions for employment and in a globalized society.

All undergraduate programs at Jacobs University are based on a coherently modularized structure, which provides students with a certain degree of flexibility regarding their individual study path and which ensures that they can complete their studies within the regular period.

The framework policies and procedures regulating undergraduate study programs at Jacobs University can be found on the website (https://www.jacobs-university.de/academic-policies).

2.2 The Curriculum

2.2.1 Year 1

The first study year is characterized by a university-specific offering of disciplinary education that builds on and expands upon the students' entrance qualifications. Students take introductory modules for a total of 60 CP from the Year 1 area. The Academic Advising Coordinator offers curriculum counseling to all Bachelor students independently of their major, while Academic Advisors, in their capacity as contact persons from the faculty, support students individually in deciding on their major study program.

Computer Science and Software Engineering students take the following mandatory modules in the first semester (30 CP)

- Module: Introduction to Computer Science (7.5 CP)
- Module: Programming in C/C++ (7.5 CP)
- Module: Introduction to Data Science (7.5 CP)
- Module: Calculus and Linear Algebra I (5 CP)
- Module: Distributed Development (Part 1) (2.5 CP)

and the following modules in the second semester (30 CP):

- Module: Algorithms and Data Structures (7.5 CP)
- Module: Introduction to Cyber Physical Systems (7.5 CP)
- Module: Software Design and Prototyping (7.5 CP)
- Module: Calculus and Linear Algebra II (5 CP)
- Module: Distributed Development (Part 2) (2.5 CP)

The modules Programming in C and C++ and Algorithms and Data Structures introduce students to imperative and object-oriented programming and basic algorithms and data structures. The Introduction to Computer Science module discusses abstract and concrete notions of computing machines and algorithms, and the representation of information. Students are also exposed to a pure functional programming language. The Software Design and Prototyping module deals with prototyping software, also known as mockup systems. It is complemented by the

Distributed Development module that deals with practical aspects of remotely developing software in teams distributed at different physical locations. The module Introduction to Cyber Physical Systems deals with the relations and interfaces of software to computer hardware, embedded systems, sensors and actuators, and networking. Relevant mathematical content is covered in the Calculus and Linear Algebra modules and in the Introduction to Data Science module.

2.2.2 Year 2

In their second year, students take a total of 50 CP from a selection of in-depth, discipline-specific modules. Building on the introductory Year 1 modules and applying the methods and skills students have already acquired so far, these modules aim to expand the students' critical understanding of the key theories, principles, and methods in their major for the current state of knowledge and best practice.

In Year 2, Computer Science and Software Engineering students acquire the following mandatory modules (50 CP in total):

- Module: Databases and Web Services (7.5 CP)
- Module: Operating Systems (7.5 CP)
- Module: Data Analytics and Modeling (7.5 CP)
- Module: Probability and Random Processes (5 CP)
- Module: Software Engineering (7.5 CP)
- Module: Artificial Intelligence (CSSE) (7.5 CP)
- Module: Machine Learning (5 CP)
- Module: Machine Learning Tools (2.5 CP)

In the second year, core areas of Computer Science with a high relevance to modern software development are covered in the modules Databases and Web Services, Operating Systems, Artificial Intelligence (CSSE), and Machine Learning. Knowledge in Software Engineering itself is deepened in the according module. Relevant mathematical aspects are covered in the modules Probability and Random Processes and Data Analytics, where the latter – together with Artificial Intelligence (CSSE) and Machine Learning – also deepens the knowledge related to Data Science. Multiple modules include practical software development aspects, namely Software Engineering, Databases and Web Services, Artificial Intelligence (CSSE), Machine Learning, Machine Learning Tools and Data Analytics and Modeling.

The remaining 10 CP can be chosen from the Elective area, which includes selected German language, mathematics and skills, and problem-solving oriented modules that tackle global challenges beyond disciplinary boundaries. An updated list of all modules in the Elective area will be available in the online course catalogue at the start of the second academic year.

2.2.3 Year 3

During their third year, students prepare for and make decisions about their career after graduation. To explore available choices fitting individual interests, and to gain professional experience, students take a mandatory summer internship (see 2.2.3.1). The third year of studies allows Computer Science and Software Engineering students to take CSSE Specialization modules, mandatory Management modules and one further Elective module (as described in Chapter 2.2.2).

2.2.3.1 Internship/Startup and Career Skills Module

As a core element of Jacobs University's employability approach students are required to engage in a mandatory two-month internship of 15 CP that will usually be completed during the summer between the second and third years of study. This gives students the opportunity to gain first-hand practical experience in a professional environment, apply their knowledge and understanding in a professional context, reflect on the relevance of their major to employment and society, reflect on their own personal role in employment and society, and develop a professional orientation. The internship can also establish valuable contacts for the students' bachelor's thesis project, for the selection of a master program graduate school or further employment after graduation. This module is complemented by career advising and several career skills workshops throughout all six semesters that prepare students for the transition from student life to professional life. As an alternative to the full-time internship, students interested in setting up their own company can apply for a start-up option to focus on developing their business plans.

For further information, please contact the Career Services Center https://www.jacobs-university.de/career-services)

2.2.3.2 CSSE Specialization Modules

In the third year of their studies, students take 15 CP of advanced CSSE Specialization modules to consolidate their knowledge and to be exposed to state-of-the-art research in the areas of their interest. This curricular component is offered as a portfolio of modules, from among which students can select freely during their fifth and sixth semester. The default module size is 5 CP, with smaller 2.5 CP modules being possible as justified exceptions.

Computer Science and Software Engineering students take at least 15 CP from the following abridged list of CSSE Specialization Modules:

- CSSE Specialization Module: Computer Graphics (5 CP)
- CSSE Specialization Module: Computer Networks (5 CP)
- CSSE Specialization Module: Web Application Development (5 CP)
- CSSE Specialization Module: Human Computer Interaction (5 CP)

An updated list of all modules in the CSSE Specialization area will be available in the online course catalogue at the start of the third academic year.

2.2.3.3 Management Modules

Students take 10 CP from the Management area to acquire valuable knowledge in the field of business and management. Modules in this area aim to bridge the gap from software development to marketable software products and to prepare students interested in a management-oriented career track.

A broad spectrum of topics is tackled, such as product development, innovation, marketing, leadership, general business, and change management.

An updated list of all modules in the Management area will be available in the online course catalogue at the start of the third academic year.

2.2.3.4 Collaborative Software Project

In the collaborative software project, the students deepen their knowledge and skills in one or multiple areas of the first and especially second year. They are exposed to state-of-the-art research with the goal to derive ideas and strategies to address application-oriented problems and to develop software for them. Students learn how to organize and execute an application-oriented research and development (R&D) project. Students are expected to organize themselves in group work under the guidance of the instructor.

2.2.3.5 Bachelor Thesis

This module is a mandatory graduation requirement for all undergraduate students. The title of the thesis will appear on the students' transcripts.

Within this module, students apply the knowledge skills, and methods they have acquired in their major discipline to become acquainted with actual research topics, ranging from the identification of suitable (short-term) research projects, preparatory literature searches, the realization of discipline-specific research, and the documentation, discussion, and interpretation of the results.

With their Bachelor Thesis students demonstrate mastery of the contents and methods of their major-specific research field. Furthermore, students show the ability to analyze and solve a well-defined problem with scientific approaches, a critical reflection of the status quo in scientific literature, and the original development of their own ideas. With the permission of a Jacobs Faculty Supervisor, the Bachelor Thesis can also have an interdisciplinary nature.

3 Computer Science and Software Engineering Undergraduate Program Regulations

3.1 Scope of these Regulations

The regulations in this handbook are valid for all students who entered the Computer Science and Software Engineering undergraduate program at Jacobs University in Fall 2022. In case of a conflict between the regulations in this handbook and the general Policies for Bachelor Studies, the latter apply (see https://www.jacobs-university.de/academic-policies).

In exceptional cases, certain necessary deviations from the regulations of this study handbook might occur during the course of study (e.g., change of the semester sequence, assessment type, or the teaching mode of courses). Jacobs University Bremen reserves therefore the right to modify the regulations of the program handbook.

3.2 Degree

Upon successful completion of this study program, students are awarded a Bachelor of Science degree in Computer Science and Software Engineering.

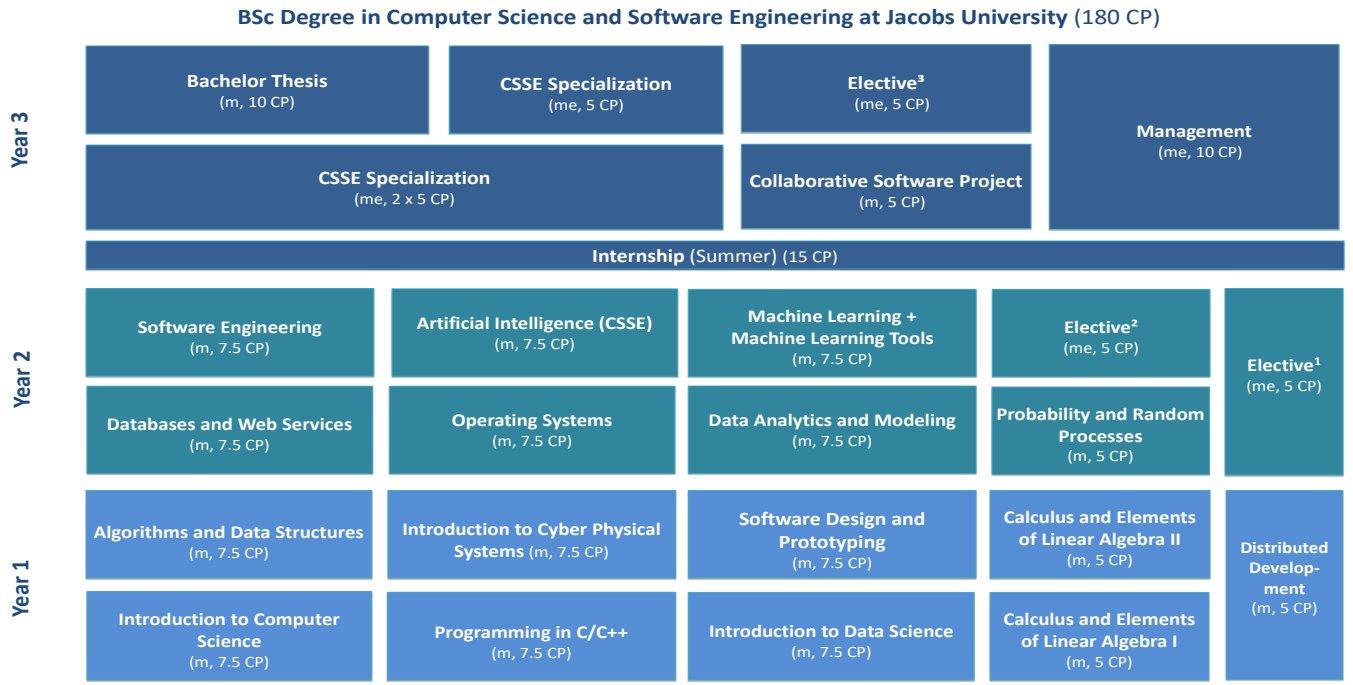
3.3 Graduation Requirements

In order to graduate, students need to obtain 180 CP. In addition, the following graduation requirements apply:

Students need to complete all mandatory components of the program as indicated in the Study and Examination Plan in Chapter 5 of this handbook.

4 Schematic Study Plan for Computer Science and Software Engineering

Figure 1 shows schematically the sequence and types of modules required for the study program. A more detailed description, including the assessment types, is given in the Study and Examination Plans in the following section.



m = mandatory
me = mandatory elective
1 = Options, e.g. (a) German; (b) Academic Skills in CS & Legal and Ethical Aspects of Computer Science
2,3 = Options, e.g. (a) Numerical Methods; (b) Ethics and Science in Technology; (c) Global Existential Risks

Figure 1: Schematic Study Plan for Computer Science and Software Engineering

5 Study and Examination Plan

Computer Science and Software Engineering (CSSE) BSc

Matriculation Fall 2022

	Program-Specific Modules	Type	Assessment	Period	Status[1]	Sem.	CP
Year 1							**60**
Take all the mandatory YEAR 1 modules listed below, as this is a requirement for the Computer Science and Software Engineering program.							
CH-232	**Module: Introduction to Computer Science[5]**				**m**	**1**	**7.5**
CH-232-A	Introduction to Computer Science	Lecture	Written examination	Examination period			7.5
CH-230	**Module: Programming in C and C++**				**m**	**1**	**7.5**
CH-230-A	Programming in C and C++	Lecture	Written examination	Examination period			2.5
CH-230-B	Programming in C and C++ Tutorial	Tutorial	Practical assessment	During the semester			5
CH-700	**Module: Introduction to Data Science**				**m**	**1**	**7.5**
CH-700-A	Introduction to Data Science	Lecture	Written examination	Examination period			7.5
JTMS-MAT-09	**Module: Calculus and Elements of Linear Algebra I**				**m**	**1**	**5**
JTMS-09	Calculus and Elements of Linear Algebra I	Lecture	Written examination	Examination period			5
CH-231	**Module: Algorithms and Data Structures**				**m**	**2**	**7.5**
CH-231-A	Algorithms and Data Structures	Lecture	Written examination	Examination period			7.5
BCSSE-Y1-02	**Module: Introduction to Cyber Physical Systems**				**m**	**2**	**7.5**
BCSSE-Y1-CPS-A	Introduction to Cyber Physical Systems (CPS) Lecture	Lecture	Written examination	Examination period			5
BCSSE-Y1-CPS-B	Introduction to Cyber Physical Systems (CPS) Tutorial	Tutorial	Assignments	During the semester			2.5
BCSSE-Y1-03	**Module: Software Design and Prototyping**				**m**	**2**	**7.5**
BCSSE-Y1-SDP-A	Software Design and Prototyping Lecture	Lecture	Written examination	Examination period			5
BCSSE-Y1-SDP-B	Software Design and Prototyping Tutorial	Tutorial	Project	During the semester			2.5
JTMS-MAT-10	**Module: Calculus and Elements of Linear Algebra II**				**m**	**2**	**5**
JTMS-10	Calculus and Elements of Linear Algebra II	Lecture	Written examination	Examination period			5
BCSSE-Y1-01	**Module: Distributed Development**				**m**	**1/2**	**5**
BCSSE-Y1-DD-A	Distributed Development I	Lecture & Lab	Practical assessment	During the semester		1	2.5
BCSSE-Y1-DD-B	Distributed Development II	Lecture & Lab	Practical assessment	During the semester		2	2.5
Year 2							**60**
Take all the mandatory YEAR 2 modules listed below (50 CP), as this is a requirement for the Computer Science and Software Engineering program.Further, please choose 10 CP of Electives.							
CO-560	**Module: Databases and Web Services**				**m**	**3**	**7.5**
CO-560-A	Databases and Web Services - Lecture	Lecture	Written examination	Examination period			5
CO-560-B	Databases and Web Services - Project	Project	Project	During the semester			2.5
CO-562	**Module: Operating Systems**				**m**	**3**	**7.5**
CO-562-A	Operating Systems	Lecture	Written examination	Examination period			7.5
CO-710	**Data Analytics and Modeling**				**m**	**3**	**7.5**
CO-710-A	Data Analytics and Modeling	Lecture	Written examination	Examination period			7.5
JTMS-MAT-12	**Module: Probability and Random Processes**				**m**	**3**	**5**
JTMS-12	Probability and Random Processes	Lecture	Written examination	Examination period			5
CO-561	**Module: Software Engineering**				**m**	**4**	**7.5**
CO-561-A	Software Engineering	Lecture	Written examination	Examination period			2.5
CO-561-B	Software Engineering Project	Project	Project	During the semester			5
CO-547	**Module: Artificial Intelligence**				**m**	**4**	**7.5**
CO-547-A	Artificial Intelligence	Lecture	Written examination	Examination period			5
xxx	Artificial Intelligence Tutorial (CSSE)	Tutorial	Assignments	During the semester			2.5
xxx	**Module: Machine Learning + Machine Learning Tools**				**m**	**4**	**7.5**
CO-541-A	Machine Learning	Lecture	Written examination	Examination period			5
xxx	Machine Learning Tools	Lab	Lab Assignments	During semester			2.5
	Electives[2] *Take a total of 10 CP CSSE Electives*				**m**	**3/4**	**10**

Year 3							**60**
Take all mandatory Year 3 modules listed below (30 CP). Further, select 15 CP of CSSE Specialization Modules, 10 CP of Management Modules and 5 CP of Elective Modules.							
CA-INT-900	**Module: Summer Internship**				**m**	**4/5**	**15**
CA-INT-900-0	Summer Internship		Report	During the 5th semester			15
xxx	**Module: Thesis / Seminar CSSE**				**m**	**6**	**10**
xxx	Thesis CSSE	Thesis	Thesis	15th of May			10
xxx	**Module: Collaborative Software Project**				**m**	**5**	**5**
xxx	Collaborative Software Project	Project	Project report	During the semester			5
	CSSE Specialization Modules[3] *Take a total of 15 CP CSSE Specialization Modules*				**m**	**5/6**	**15**
	Management Modules[4] *Take a total of 10 CP Management Modules.*				**m**	**5/6**	**10**
	Electives[2] *Take a total of 5 CP Electives*				**m**	**6**	**5**
Total CP							**180**
[1] Status (m = mandatory, me = mandatory elective)							
[2] For a full listing of all Elective modules please consult the current online course catalogue and /or the study program handbooks.							
[3] For a full listing of all CSSE Specialization modules please consult the current online course catalogue and /or the study program handbooks.							
[4] For a full listing of all Management modules please consult the current online course catalogue and /or the study program handbooks.							

Figure 2: Study and Examination Plan

6 Module Descriptions

6.1 YEAR 1

6.1.1 Introduction to Computer Science

<table>
<tr><td colspan="2">Module Name
Introduction to Computer Science</td><td>Module Code
CH-232</td><td>Level (type)
Year 1
(CHOICE)</td><td>CP
7.5</td></tr>
<tr><td colspan="5">Module Components

Number | Name | Type | CP
CH-232-A | Introduction to Computer Science | Lecture | 7.5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Jürgen Schönwälder</td><td colspan="2">Program Affiliation

• Computer Science (CS)</td><td colspan="2">Mandatory Status

Mandatory for CS, CSSE, ECE and RIS</td></tr>
<tr><td colspan="2">Entry Requirements

Pre-requisites
☒ None

Co-requisites
☒ None

Knowledge, Abilities, or Skills</td><td>Frequency

Every semester (Fall/Spring)

Duration

1 semester</td><td colspan="2">Forms of Learning and Teaching

• Class (online) (52.5 hours)
• Independent study (115 hours)
• Exam preparation (20 hours)

Workload

187.5 hours</td></tr>
<tr><td colspan="5">Recommendations for Preparation

It is recommended that students install a Linux system such as Ubuntu on their notebooks and that they become familiar with basic tools such as editors (vim or emacs) and the basics of a shell. The Glasgow Haskell Compiler (GHC) will be used for implementing Haskell programs.</td></tr>
<tr><td colspan="5">Content and Educational Aims

The module introduces fundamental concepts and techniques of computer science in a bottom-up manner. Based on clear mathematical foundations (which are developed as needed), the course discusses abstract and concrete notions of computing machines, information, and algorithms, focusing on the question of representation versus meaning in Computer Science.

The module introduces basic concepts of discrete mathematics with a focus on inductively defined structures, to develop a theoretical notion of computation. Students will learn the basics of the functional programming language Haskell because it treats computation as the evaluation of pure and typically inductively defined functions. The module covers a basic subset of Haskell that includes types, recursion, tuples, lists, strings, higher-order functions, and finally monads. Back on the theoretical side, the module covers the syntax and semantics of Boolean expressions and it explains how Boolean algebra relates to logic gates and digital circuits. On the technical side, the course introduces the representation of basic data types such as numbers, characters, and</td></tr>
</table>

strings as well as the von Neuman computer architecture. On the algorithmic side, the course introduces the notion of correctness and elementary concepts of complexity theory (big O notation).

Intended Learning Outcomes

By the end of this module, students will be able to

1. explain basic concepts such as the correctness and complexity of algorithms (including the big O notation);
2. illustrate basic concepts of discrete math (sets, relations, functions);
3. recall basic proof techniques and use them to prove properties of algorithms;
4. explain the representation of numbers (integers, floats), characters and strings, and date and time;
5. summarize basic principles of Boolean algebra and Boolean logic;
6. describe how Boolean logic relates to logic gates and digital circuits;
7. outline the basic structure of a von Neumann computer;
8. explain the execution of machine instructions on a von Neumann computer;
9. describe the difference between assembler languages and higher-level programming languages;
10. define the differences between interpretation and compilation;
11. illustrate how an operating system kernel supports the execution of programs;
12. determine the correctness of simple programs;
13. write simple programs in a pure functional programming language.

Indicative Literature

Eric Lehmann, F. Thomson Leighton, Albert R. Meyer: Mathematics for Computer Science, online 2018.

David A. Patterson, John L Hennessy: Computer Organization and Design: The Hardware/Software Interface, 4th edition, Morgan Kaufmann, 2011.

Miran Lipovaca: Learn You a Haskell for Great Good!: A Beginner's Guide, 1st edition, No Starch Press, 2011.

Usability and Relationship to other Modules

- Mandatory for a major in CS, CSSE, ECE and RIS
- Pre-requisite for the CORE modules Automata, Computability, and Complexity and Operating Systems
- This module introduces key mathematical concepts and various notions of computing machines and computing abstractions and is in particularly important for subsequent courses covering theoretical aspects of computer science. This module is also important for courses that require a basic understanding of computer architecture and program execution at the hardware level.

Examination Type: Module Examination

Assessment Type: Written examination — Duration: 120 min
Weight: 100%

Scope: All intended learning outcomes of the module

Module achievement: 50% of the assignments correctly solved

This module introduces the functional programming language Haskell. Students develop their functional programming skills by solving programming problems. The module achievement ensures that a sufficient level of practical programming and problem-solving skills has been obtained.

6.1.2 Programming in C and C++

<table>
<tr><td colspan="3">Module Name
Programming in C and C++</td><td>Module Code
CH-230</td><td>Level (type)
Year 1
(CHOICE)</td><td>CP
7.5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
CH-230-A | Programming in C and C++ | Lecture | 2.5
CH-230-B | Programming in C and C++ - Tutorial | Tutorial | 5</td></tr>
<tr><td>Module Coordinator

Dr. Kinga Lipskoch</td><td colspan="3">Program Affiliation

• Computer Science (CS)</td><td colspan="2">Mandatory Status

Mandatory for CS, CSSE, RIS and ECE</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites | Co-requisites | Knowledge, Abilities, or Skills
☒ None | ☒ None | ☒ None</td><td>Frequency

Annually (Fall)

Duration

1 semester</td><td colspan="2">Forms of Learning and Teaching
• Lecture attendance (online) (17.5 hours)
• Tutorial attendance (35 hours)
• Independent study (115 hours)
• Exam preparation (20 hours)

Workload

187.5 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

It is recommended that students install a suitable programming environment on their notebooks. It is recommended to install a Linux system such as Ubuntu, which comes with open-source compilers such as gcc and g++ and editors such as vim or emacs. Alternatively, the open-source Code: Blocks integrated development environment can be installed to solve programming problems.</td></tr>
<tr><td colspan="6">Content and Educational Aims

This course offers an introduction to programming using the programming languages C and C++. After a short overview of the program development cycle (editing, preprocessing, compiling, linking, executing), the module presents the basics of C programming. Fundamental imperative programming concepts such as variables, loops, and function calls are introduced in a hands-on manner. Afterwards, basic data structures such as multidimensional arrays, structures, and pointers are introduced and dynamically allocated multidimensional arrays and linked lists and trees are used for solving simple practical problems. The relationships between pointers and arrays, pointers and structures, and pointers and functions are described, and they are illustrated using examples that also introduce recursive functions, file handling, and dynamic memory allocation.

The module then introduces basic concepts of object-oriented programming languages using the programming language C++ in a hands-on manner. Concepts such as classes and objects, data abstractions, and information hiding are introduced. C++ mechanisms for defining and using objects, methods, and operators are introduced and the relevance of constructors, copy constructors, and destructors for dynamically created objects is explained. Finally, concepts such as inheritance, polymorphism, virtual functions, and overloading are introduced. The learned concepts are applied by solving programming problems.</td></tr>
</table>

Intended Learning Outcomes

By the end of this module, students will be able to

1. explain basic concepts of imperative programming languages such as variables, assignments, loops, and function calls;
2. write, test, and debug programs in the procedural programming language C using basic C library functions;
3. demonstrate how to use pointers to create dynamically allocated data structures such as linked lists;
4. explain the relationship between pointers and arrays;
5. illustrate basic object-oriented programming concepts such as objects, classes, information hiding, and inheritance;
6. give original examples of function and operator overloading and polymorphism;
7. write, test, and debug programs in the object-oriented programming language C++.

Indicative Literature

Brian Kernighan, Dennis Ritchie: The C Programming Language, 2nd edition, Prentice Hall Professional Technical Reference, 1988.

Steve Oualline: Practical C Programming, 3rd edition, O'Reilly Media, 1997.

Bruce Eckel: Thinking in C++: Introduction to Standard C++, Prentice Hall, 2000.

Bruce Eckel, Chuck Allison: Thinking in C++: Practical Programming, Prentice Hall, 2004.

Bjarne Stroustrup: The C++ Programming Language, 4th edition, Addison Wesley, 2013.

Michael Dawson: Beginning C++ Through Game Programming, 4th edition, Delmar Learning, 2014.

Usability and Relationship to other Modules

- Mandatory for a major in CS, CSSE, RIS, and ECE
- Mandatory for a minor in CS and RIS
- Pre-requisite for the CHOICE module Algorithms and Data Structures
- Elective for all other undergraduate study programs
- This module introduces the programming languages C and C++ and several other modules build on this foundation. Certain features of C++ such as templates and generic data structures and an overview of the standard template library will be covered in the Algorithms and Data Structures module.

Examination Type: Module Component Examinations

Component 1: Lecture

Assessment types: Written examination — Duration: 120 min, Weight: 33%

Scope: All theoretical intended learning outcomes of the module

Component 2: Tutorial

Assessment: Practical assessment (Programming assignments) — Weight: 67%

Scope: All practical intended learning outcomes of the module

Completion: To pass this module, the examination of each module component has to be passed with at least 45%.

6.1.3 Introduction to Data Science

<table>
<tr><td colspan="3">Module Name

Introduction to Data Science</td><td>Module Code
CH-700</td><td>Level (type)
Year 1</td><td>ECTS
7.5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | ECTS
CH-700-A | Introduction to Data Science | Lecture | 7.5</td></tr>
<tr><td>Module Coordinator</td><td colspan="3">Program Affiliation

• Minor in Data Science</td><td colspan="2">Mandatory Status

Mandatory for CSSE and Minor in Data Science</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites
☒ None

Co-requisites
☒ None

Knowledge, Abilities, or Skills</td><td>Frequency

Annually (Fall)</td><td colspan="2">Forms of Learning and Teaching

• Lectures (hybrid/online) (52.5 hours)
• Private Study (135 hours)</td></tr>
<tr><td colspan="3"></td><td>Duration

1 semester</td><td colspan="2">Workload

187.5 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

None.</td></tr>
<tr><td colspan="6">Content and Educational Aims

The module introduces data science with an integrated presentation of three essential components, namely, (1) societal/legal implications and business opportunities, (2) technical/theoretical background and case studies, (3) an introduction to the Python coding environment. The first component entails a conceptual introduction to the opportunities and the challenges of a digitally transformed and data-driven society, presentations on industry standards and legal frameworks, and discussions of critical issues such as cybersecurity and surveillance. The second component includes topics such as data science terminology, digital data and their representations, and introductions to exploratory data analysis and prominent supervised and unsupervised learning tasks. The third component offers an introduction to the Python ecosystem of data representation, processing, analysis, and visualization, starting with Jupyter notebooks, installing suitable environments, and introductions to data science related packages such as NumPy, SciPy, Matplotlib, Seaborn, and Pandas. Fundamental data science concepts are summarized and illustrated using real-world data from various disciplines. Flexible educational formats (mostly online and hybrid) allow for asynchronous learning. Lectures are combined with an exposure to Python programming and data processing and visualization environments, including hands-on practicals, examples, and exercises.</td></tr>
</table>

Intended Learning Outcomes

By the end of this module, students will be able to

- explain societal implications of the digital transformation,
- understand the legal data protection framework,
- carry out basic data processing and visualization tasks,
- apply fundamental data science methods to structured data,
- understand the logic of Python scripts and functions,
- compose Python code using templates

Indicative Literature

Ani Adhikari, John DeNero, David Wagner. Computational and Inferential Thinking: The Foundations of Data Science. Originally developed for the UC Berkeley course Data 8: Foundations of Data Science. An online version of the textbook is available at https://inferentialthinking.com/.
The Alan Turing Institute, Data Science for the Social Good.
Philip D . Brooker. Programing with Python for Social Scientists. Sage 2020.
Shin Takahasi, Iroha Inoue. The Manga Guide to Linear Algebra. Trend-Pro 2012.
Steven S. Skiena. The Data Science Design Manual. Springer 2017.
Jake Vanderplas. Python Data Science Handbook. O'Reilly 2016. An online version is available at https://jakevdp.github.io/PythonDataScienceHandbook/.
Shoshana Zuboff. The Age of Surveillance Capitalism. London: Profile 2019.

Usability and Relationship to other Modules

-

Examination Type: Module Examination

Type: Written Examination — Duration/Length: 180 min
Scope: All intended learning outcomes of the module. — Weight: 100 %

Module achievement: 50% of the assignments need to be correctly solved.

6.1.4 Calculus and Elements of Linear Algebra I

<table>
<tr><td colspan="3">Module Name
Calculus and Elements of Linear Algebra I</td><td>Module Code
JTMS-MAT-09</td><td>Level (type)
Year 1 (Methods)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
JTMS-09 | Calculus and Elements of Linear Algebra I | Lecture | 5</td></tr>
<tr><td colspan="2">Module Coordinator

Dr. Keivan Mallahi Karai, Prof. Dr. Tobias Preußer</td><td colspan="2">Program Affiliation

• Jacobs Track – Methods and Skills</td><td colspan="2">Mandatory Status

Mandatory for CS, CSSE, ECE, RIS, MATH and Physics
Mandatory elective for EES</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites
☒ None

Co-requisites
☒ None

Knowledge, Abilities, or Skills
• Knowledge of Pre-Calculus at High School level (Functions, inverse functions, sets, real numbers, polynomials, rational functions, trigonometric functions, logarithm and exponential function, parametric equations, tangent lines, graphs, elementary methods for solving systems of linear and nonlinear equations)
• Knowledge of Analytic Geometry at High School level (vectors, lines, planes, reflection, rotation, translation, dot product, cross product, normal vector, polar coordinates)
• Some familiarity with elementary Calculus (limits, derivative) is helpful, but not strictly required.</td><td>Frequency

Annually (Fall)

Duration

1 semester</td><td colspan="2">Forms of Learning and Teaching

• Lectures (online) (35 hours)
• Private study (90 hours)

Workload

125 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

Review all of higher-level High School Mathematics, in particular the topics explicitly named in “Entry Requirements – Knowledge, Ability, or Skills” above.</td></tr>
</table>

Content and Educational Aims

This module is the first in a sequence introducing mathematical methods at the university level in a form relevant for study and research in the quantitative natural sciences, engineering, Computer Science, and Mathematics. The emphasis in these modules is on training operational skills and recognizing mathematical structures in a problem context. Mathematical rigor is used where appropriate. However, a full axiomatic treatment of the subject is provided in the first-year modules "Analysis I" and "Linear Algebra".

The lecture comprises the following topics

- Brief review of number systems, elementary functions, and their graphs
- Brief introduction to complex numbers
- Limits for sequences and functions
- Continuity
- Derivatives
- Curve sketching and applications (isoperimetric problems, optimization, error propagation)
- Introduction to Integration and the Fundamental Theorem of Calculus
- Review of elementary analytic geometry
- Vector spaces, linear independence, bases, coordinates
- Matrices and matrix algebra
- Solving linear systems by Gauss elimination, structure of general solution
- Matrix inverse

Intended Learning Outcomes

By the end of the module, students will be able to

- apply the methods described in the content section of this module description to the extent that they can solve standard text-book problems reliably and with confidence;
- recognize the mathematical structures in an unfamiliar context and translate them into a mathematical problem statement;
- recognize common mathematical terminology used in textbooks and research papers in the quantitative sciences, engineering, and mathematics to the extent that they fall into the content categories covered in this module.

Indicative Literature

S.I. Grossman (2014). Calculus of one variable, 2nd edition. Cambridge: Academic Press.
S.A. Leduc (2003). Linear Algebra. Hoboken: Wiley.
K. Riley, M. Hobson, S. Bence (2006). Mathematical Methods for Physics and Engineering, third edition. Cambridge: Cambridge University Press.

Usability and Relationship to other Modules

- The module is a mandatory / mandatory elective module of the Methods and Skills area that is part of the Jacobs Track (Methods and Skills modules; Community Impact Project module; Language modules; Big Questions modules).
- The module is followed by "Calculus and Elements of Linear Algebra II". All students taking this module are expected to register for the follow-up module.
- A rigorous treatment of Calculus is provided in the module "Analysis I". All students taking "Analysis I" are expected to either take this module or exceptionally satisfy the conditions for advanced placement as laid out in the Jacobs Academic Policies for Undergraduate Study.
- The second-semester module "Linear Algebra" will provide a complete proof-driven development of the theory of Linear Algebra. Students enrolling in "Linear Algebra" are expected to have taken this module; in particular, the module "Linear Algebra" will assume that students are proficient in the operational aspects of Gauss elimination, matrix inversion, and their elementary applications.
- This module is a prerequisite for the module "Applied Mathematics" which develops more advanced theoretical and practical mathematical tools essential for any physicist or mathematician.
- Mandatory for a major in CS, CSSE, ECE, RIS, MATH and Physics
- Mandatory elective for a major in EES.
- Pre-requisite for Calculus and Elements of Linear Algebra II
- Elective for all other study programs.

Examination Type: Module Examination

Assessment type: Written examination — Duration: 120 min
Weight: 100%

Scope: All intended learning outcomes of this module

6.1.5 Algorithms and Data Structures

<table>
<tr><td colspan="2">Module Name
Algorithms and Data Structures</td><td>Module Code
CH-231</td><td>Level (type)
Year 1
(CHOICE)</td><td>CP
7.5</td></tr>
<tr><td colspan="5">Module Components

Number | Name | Type | CP
CH-231-A | Algorithms and Data Structures | Lecture | 7.5</td></tr>
<tr><td>Module Coordinator

Dr. Kinga Lipskoch</td><td colspan="2">Program Affiliation

• Computer Science (CS)</td><td colspan="2">Mandatory Status

Mandatory for CS, CSSE and RIS</td></tr>
<tr><td colspan="2">Entry Requirements

Pre-requisites: ☒ Programming in C and C++
Co-requisites: ☒ None
Knowledge, Abilities, or Skills</td><td>Frequency

Annually (Spring)</td><td colspan="2">Forms of Learning and Teaching

• Class attendance (online) (52.5 hours)
• Independent study (115 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td colspan="2"></td><td>Duration

1 semester</td><td colspan="2">Workload

187.5 hours</td></tr>
<tr><td colspan="5">Recommendations for Preparation

Students should refresh their knowledge of the C and C++ programming language and be able to solve simple programming problems in C and C++. Students are expected to have a working programming environment.</td></tr>
<tr><td colspan="5">Content and Educational Aims

Algorithms and data structures are the core of computer science. An algorithm is an effective description for calculations using a finite list of instructions that can be executed by a computer. A data structure is a concept for organizing data in a computer such that data can be used efficiently. This introductory module allows students to learn about fundamental algorithms for solving problems efficiently. It introduces basic algorithmic concepts; fundamental data structures for efficiently storing, accessing, and modifying data; and techniques that can be used for the analysis of algorithms and data structures with respect to their computational and memory complexities. The presented concepts and techniques form the basis of almost all computer programs.</td></tr>
<tr><td colspan="5">Intended Learning Outcomes

By the end of this module, students will be able to
1. explain asymptotic (time and memory) complexities and respective notations;
2. able to prove asymptotic complexities of algorithms;
3. illustrate basic data structures such as arrays, lists, queues, stacks, trees, and hash tables;
4. describe algorithmic design concepts and apply them to new problems;
5. explain basic algorithms (sorting, searching, graph algorithms, computational geometry) and their complexities;</td></tr>
</table>

6. summarize and apply C++ templates and generic data structures provided by the standard C++ template library.

Indicative Literature

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: Introduction to Algorithms, 3rd edition, MIT Press, 2009.

Donald E. Knuth: The Art of Computer Programming: Fundamental Algorithms, volume 1, 3rd edition, Addison Wesley Longman Publishing, 1997.

Usability and Relationship to other Modules

- Mandatory for a major in CS, CSSE and RIS
- Mandatory for a minor in CS
- Pre-requisite for the following CORE modules:
  - Databases and Web Services
  - Software Engineering
  - Legal and Ethical Aspects of Computer Science
  - Computer Graphics
  - Distributed Algorithms
- Familiarity with basic algorithms and data structures is fundamental for almost all advanced modules in computer science. This module additionally introduces advanced concepts of the C++ programming language that are needed in advanced programming-oriented modules in the 2nd and 3rd years of the CS and RIS programs.

Examination Type: Module Examination

Assessment Type: Written examination — Duration: 120 min, Weight: 100%

Scope: All intended learning outcomes of the module

6.1.6 Introduction to Cyber Physical Systems

<table>
<tr><td colspan="2">Module Name
Introduction to Cyber Physical Systems (CPS)</td><td>Module Code</td><td>Level (type)
Year 1</td><td>CP
7.5</td></tr>
<tr><td colspan="5">Module Components

Number | Name | Type | CP
| Introduction to Cyber Physical Systems (CPS) Lecture | Lecture | 5
| Introduction to Cyber Physical Systems (CPS) Tutorial | Tutorial | 2.5</td></tr>
<tr><td>Module Coordinator
NN</td><td colspan="2">Program Affiliation

• BSc Computer Science and Software Engineering</td><td colspan="2">Mandatory Status

Mandatory for CSSE</td></tr>
<tr><td colspan="2">Entry Requirements

Pre-requisites
☒ Calculus and Elements of Linear Algebra I

Co-requisites
☒ none

Knowledge, Abilities, or Skills</td><td>Frequency

Annually (Spring)

Duration

1 semester</td><td colspan="2">Forms of Learning and Teaching

• Lectures (online) (35 hours)
• Tutorials (17.5 hours)
• Private study (115 hours)
• Exam preparation (20 hours)

Workload

187.5 hours</td></tr>
<tr><td colspan="5">Recommendations for Preparation

Students are expected to be familiar with the core elements of calculus and linear algebra.</td></tr>
<tr><td colspan="5">Content and Educational Aims

The area of Cyber Physical Systems (CPS) deals with the interface between the digital and the physical world, i.e., the relations and interfaces of software to computer hardware, embedded systems, sensors and actuators, and networking. Application examples range from large entities like power-grids, factories, or warehouses, down to smaller systems like automobiles, home automation, or machinery in production or warehouses. CPS builds on interconnected smart devices and intelligent autonomous systems, which may range from small simple sensor-nodes to more capable systems that may also features mobility and manipulation. It hence relates software development to aspects of computer architecture, communications, system integration, modelling, control, and artificial intelligence.</td></tr>
<tr><td colspan="5">Intended Learning Outcomes

Upon completion of this module, students will be able to:
1. Describe the different use-cases and application areas of CPS
2. Explain the components of CPS and their interplay
3. Understand computer architecture and be able to apply core concepts within embedded computing
4. Generate software interfaces to sensors and actuators
5. Understand the networking aspects related to CPS and apply them within the context of embedded computing
6. Explain real-time requirements and understand the related core software concepts and algorithms
7. Be able to model systems
8. Understand and apply the basics of control of physical systems in form of software
9. Explain core concepts and methods of software for intelligent autonomous systems</td></tr>
</table>

10. Understand and use software methods for remote access for monitoring, operation, and maintenance of physical systems and processes

Indicative Literature

-

Usability and Relationship to other Modules

- The module serves as a mandatory module for CSSE students.

Examination Type: Module Examination

Module Component 1: Lecture
Assessment Type: Written examination — Duration/length: 120 min
Weight: 50%

Scope: All intended learning outcomes of the module (with focus on theory).

Module Component 2: Tutorial
Assessment Type: Assignments
Weight: 50%

Scope: All intended learning outcomes of the module (with focus on practical content).

Completion: To pass this module, the examination of each module component has to be passed with at least 45%.

6.1.7 Software Design and Prototyping

<table>
<tr><td colspan="2">Module Name
Software Design and Prototyping</td><td>Module Code</td><td>Level (type)
Year 1</td><td>CP
7.5</td></tr>
<tr><td colspan="5">Module Components

Number | Name | Type | CP
| Software Design and Prototyping Lecture | Lecture | 5
| Software Design and Prototyping Tutorial | Tutorial | 2.5</td></tr>
<tr><td>Module Coordinator
NN</td><td colspan="2">Program Affiliation

• BSc Computer Science and Software Engineering</td><td colspan="2">Mandatory Status

Mandatory for CSSE</td></tr>
<tr><td colspan="2">Entry Requirements

Pre-requisites
☒ Programming in C/C++

Co-requisites
☒ Distributed Development II

Knowledge, Abilities, or Skills</td><td>Frequency

Annually (Spring)</td><td colspan="2">Forms of Learning and Teaching

• Lectures (online) (35 hours)
• Tutorials (17.5 hours)
• Private study (115 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td colspan="2"></td><td>Duration

1 semester</td><td colspan="2">Workload

187.5 hours</td></tr>
<tr><td colspan="5">Recommendations for Preparation

Students are expected to be familiar with programming in C/C++ and the basics of collaborative, remote software development.</td></tr>
<tr><td colspan="5">Content and Educational Aims

During the early phases of software projects, it is often unclear what the exact requirements are and how a suitable software design could look like. Since wrong decisions taken during the early phases of a software project frequently have significant impact on the completion time and the overall costs of a software project, it is often desirable to quickly construct prototype systems. Prototype systems can not only be used to collect early feedback in order to clarify requirements. They can also be used to acquire additional customers. This module introduces software design pattern with a specific focus on the construction of early prototypes, sometimes also called mockup systems.</td></tr>
<tr><td colspan="5">Intended Learning Outcomes

Upon completion of this module, students will be able to:

1. select software architectures supporting fast prototyping
2. implement interaction prototypes using suitable mockup tools
3. implement backend and server prototypes using suitable mockup tools
4. derive designs of interaction prototypes from incomplete user input
5. conduct an evaluation of mockup prototypes with target users
6. be able to revise prototypes efficiently in an agile manner
7. effectively work in a team prototyping different software components
8. create mock objects that can be used effectively for unit tests</td></tr>
<tr><td colspan="5">Indicative Literature

▪</td></tr>
</table>

Usability and Relationship to other Modules

- The module serves as a mandatory module for CSSE students. It builds on Programming in C/C++ and Distributed Development I.

Examination Type: Module Examination

Assessment: Written examination — Duration: 60 min, Weight: 50%

Scope: Intended Learning outcomes 1,4 and 6.

Assessment: Project — Weight: 50%

Scope: Intended Learning outcomes 2,3,5,7 and 8.

Completion: To pass this module, the examination of each module component has to be passed with at least 45%.

6.1.8 Calculus and Elements of Linear Algebra II

<table>
<tr><td colspan="3">Module Name
Calculus and Elements of Linear Algebra II</td><td>Module Code
JTMS-MAT-10</td><td>Level (type)
Year 1 (Methods)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
JTMS-10 | Calculus and Elements of Linear Algebra II | Lecture | 5</td></tr>
<tr><td colspan="2">Module Coordinator

Dr. Keivan Mallahi Karai, Prof. Dr. Tobias Preußer</td><td colspan="2">Program Affiliation

• Jacobs Track – Methods and Skills</td><td colspan="2">Mandatory Status

Mandatory for CS, CSSE, ECE, MATH, Physics and RIS</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites
☒ Calculus and Elements of Linear Algebra I

Co-requisites
☒ None

Knowledge, Abilities, or Skills
• None beyond formal pre-requisites</td><td>Frequency

Annually (Spring)

Duration

1 semester</td><td colspan="2">Forms of Learning and Teaching

• Lectures (online) (35 hours)
• Private study (90 hours)

Workload

125 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

Review the content of Calculus and Elements of Linear Algebra I</td></tr>
<tr><td colspan="6">Content and Educational Aims
This module is the second in a sequence introducing mathematical methods at the university level in a form relevant for study and research in the quantitative natural sciences, engineering, Computer Science, and Mathematics. The emphasis in these modules is on training operational skills and recognizing mathematical structures in a problem context. Mathematical rigor is used where appropriate. However, a full axiomatic treatment of the subject is provided in the first-year modules “Analysis I” and “Linear Algebra”.

The lecture comprises the following topics
• Directional derivatives, partial derivatives
• Linear maps
• The total derivative as a linear map
• Gradient and curl (elementary treatment only, for more advanced topics, in particular the connection to the Gauss and Stokes’ integral theorems, see module “Applied Mathematics”
• Optimization in several variables, Lagrange multipliers
• Elementary ordinary differential equations
• Eigenvalues and eigenvectors
• Hermitian and skew-Hermitian matrices
• First important example of eigendecompositions: Linear constant-coefficient ordinary differential equations
• Second important example of eigendecompositions: Fourier series
• Fourier integral transform
• Matrix factorizations: Singular value decomposition with applications, LU decomposition, QR decomposition</td></tr>
<tr><td colspan="6">Intended Learning Outcomes
By the end of the module, students will be able to
• apply the methods described in the content section of this module description to the extent that they can solve standard text-book problems reliably and with confidence;</td></tr>
</table>

- recognize the mathematical structures in an unfamiliar context and translate them into a mathematical problem statement;
- recognize common mathematical terminology used in textbooks and research papers in the quantitative sciences, engineering, and mathematics to the extent that they fall into the content categories covered in this module.

Indicative Literature

S.I. Grossman (2014). Calculus of one variable, 2nd edition. Cambridge: Academic Press.
S.A. Leduc (2003). Linear Algebra. Hoboken: Wiley.
K. Riley, M. Hobson, S. Bence (2006). Mathematical Methods for Physics and Engineering, third edition. Cambridge: Cambridge University Press.

Usability and Relationship to other Modules

- The module is a mandatory / mandatory elective module of the Methods and Skills area that is part of the Jacobs Track (Methods and Skills modules; Community Impact Project module; Language modules; Big Questions modules).
- A more advanced treatment of multi-variable Calculus, in particular, its applications in Physics and Mathematics, is provided in the second-semester module "Applied Mathematics". All students taking "Applied Mathematics" are expected to take this module as well as the module topics are closely synchronized.
- The second-semester module "Linear Algebra" provides a complete proof-driven development of the theory of Linear Algebra. Diagonalization is covered more abstractly, with particular emphasis on degenerate cases. The Jordan normal form is also covered in "Linear Algebra", not in this module.
- Mandatory for CS, ECE, MATH, Physics and RIS.
- Elective for all other study programs.

Examination Type: Module Examination

Assessment type: Written examination — Duration: 120 min, Weight: 100%

Scope: All intended learning outcomes of this module

6.1.9 Distributed Development

<table>
<tr><td colspan="2">Module Name
Distributed Development</td><td>Module Code</td><td>Level (type)
Year 1</td><td>CP
5</td></tr>
<tr><td colspan="5">Module Components

Number | Name | Type | CP
| Distributed Development I | Lecture & Lab | 2.5
| Distributed Development II | Lecture & Lab | 2.5</td></tr>
<tr><td>Module Coordinator
N.N.</td><td colspan="2">Program Affiliation

• BSc Computer Science and Software Engineering</td><td colspan="2">Mandatory Status

Mandatory for CSSE</td></tr>
<tr><td colspan="3" rowspan="2">Entry Requirements

Pre-requisites | Co-requisites | Knowledge, Abilities, or Skills
☒ none | ☒ Programming in C/C++</td><td>Frequency

Annually
Fall (I)
Spring (II)</td><td>Forms of Learning and Teaching

• Lectures (online) (17.5 hours)
• Tutorials (17.5 hours)
• Independent studies (90 hours)</td></tr>
<tr><td>Duration

2 semesters</td><td>Workload

125 hours</td></tr>
<tr><td colspan="5">Recommendations for Preparation

Previous experience with programming is a plus but not required.</td></tr>
<tr><td colspan="5">Content and Educational Aims

Software development is increasingly done in collaborative teams who work in a remote fashion, i.e., with team members who are spatially distributed at different locations, sometimes even across different time-zones. This can be very convenient for employers, who can recruit from around the globe without the need for expecting the employees to relocate, as well as for the employees, who gain some freedom in where and when they execute their tasks. But it includes also quite some challenges, e.g., for the development of a joined approach, the coordination of tasks, or the meeting of deadlines. This module provides a hands-on introduction into the methods and tools for handling these opportunities and challenges.</td></tr>
<tr><td colspan="5">Intended Learning Outcomes

Upon completion of this module, students will be able to:
1. Understand the opportunities and challenges that are involved in collaborative, remote software development
2. Comprehend the needs for and limitations of synchronous online-meeting tools
3. Use the different standard features of tools for synchronous online-meetings
4. Comprehend the concepts of versioning software and be able to apply them
5. Understand the pro's and con's of asynchronous online communication
6. Use standard features of online communication teams for brain-storming and the development of a joined approach to solve problems and the distribution of tasks
7. Understand the needs for calendars and to-do lists and how to handle them
8. Comprehend bug-trackers and be able to use them
9. Understand the possibilities and limitations, e.g., legal restrictions, of monitoring tools that, e.g., keep track of the time spend on tasks per individual team member</td></tr>
</table>

Indicative Literature ▪
Usability and Relationship to other Modules • The module serves as a mandatory module for CSSE students.
Examination Type: Module Examination Assessment: Practical assessment (tool-use assignments) Weight: 100% Scope: All intended learning outcomes of the module.

6.2 YEAR 2

6.2.1 Databases and Web Services

<table>
<tr><td colspan="2">Module Name
Databases and Web Services</td><td>Module Code
CO-560</td><td>Level (type)
Year 2 (CORE)</td><td>CP
7.5</td></tr>
<tr><td colspan="5">Module Components

Number | Name | Type | CP
CO-560-A | Databases and Web Services | Lecture | 5
CO-560-B | Databases and Web Services - Project | Project | 2.5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Peter Baumann</td><td colspan="2">Program Affiliation

• Computer Science (CS)</td><td colspan="2">Mandatory Status

Mandatory for CS and CSSE
Mandatory elective for RIS</td></tr>
<tr><td colspan="2">Entry Requirements

Pre-requisites
☒ Algorithms and Data Structures

Co-requisites
☒ None

Knowledge, Abilities, or Skills</td><td>Frequency

Annually (Fall)</td><td colspan="2">Forms of Learning and Teaching

• Class attendance (online) (35 hours)
• Project (97.5 hours)
• Independent Studies (35 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td colspan="2"></td><td>Duration

1 semester</td><td colspan="2">Workload

187.5 hours</td></tr>
<tr><td colspan="5">Recommendations for Preparation

Working knowledge of basic data structures, such as trees, is required as well as familiarity with an object-oriented programming language such as C++. Basic knowledge of algebra is useful. For the project work, students benefit from having basic hands-on skills using Linux and, ideally, basic knowledge of a scripting language such as Python (the official Python documentation is available on https://docs.python.org/).</td></tr>
<tr><td colspan="5">Content and Educational Aims

This module offers a combined introduction to databases and web services. The database part starts with database design using the Entity Relationship (ER) and Unified Modeling Language (UML) models, followed by relational databases and querying them through SQL, relational design theory, indexing, query processing, transaction management, and NoSQL/Big Data databases. In the web services part, the topics addressed include markup languages, three-tier application architectures, and web services. Security aspects are addressed from both perspectives.

A hands-on group project complements the theoretical aspects: on a self-chosen topic, students implement the core of a web-accessible information system using Python (or a similar language), MySQL, and Linux, guided through homework assignments.</td></tr>
</table>

Intended Learning Outcomes

By the end of this module, students will be able to

1. read and write ER and UML diagrams;
2. design and normalize data models for relational databases;
3. write SQL queries and understand their evaluation by a database server;
4. explain the concept of transactions and how to use transactions in application design;
5. use web application frameworks to create dynamic websites;
6. describe the differences of selected NoSQL data models and make a requirement-driven choice;
7. restate three-tier architectures and their components;
8. discuss the principles and basic mechanisms of reactive website design;
9. summarize the security and privacy issues in the context of databases and web services.

Indicative Literature

Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer D. Widom: Database Systems: The Complete Book. 2nd edition, Pearson, 2008.

Ragu Ramakrishnan: Database Management Systems. 3rd edition, McGraw Hill, 2003.

James Lee: Open Source Web Development with LAMP. Pearson, 2003.

Usability and Relationship to other Modules

- Mandatory for a major in CS and CSSE
- Mandatory for a minor in CS
- Serves as a mandatory elective specialization module for RIS major students.
- Pre-requisite for the CORE module Secure and Dependable Systems
- This module introduces components that are widely used by modern applications and information systems. Students can apply their knowledge in the software engineering module. This module serves as a default advanced level minor module.

Examination Type: Module Component Examinations

Module Component 1: Lecture

Assessment Type: Written examination — Duration: 120 min
Weight: 67%

Scope: All intended learning outcomes of the excluding the practical aspects

Module Component 2: Project

Assessment Type: Project
Weight: 33%

Scope: All practical aspects of the intended learning outcomes

Completion: To pass this module, the examination of each module component has to be passed with at least 45%.

6.2.2 Operating Systems

<table>
<tr><td colspan="2">Module Name
Operating Systems</td><td>Module Code
CO-562</td><td>Level (type)
Year 2 (CORE)</td><td>CP
7.5</td></tr>
<tr><td colspan="5">Module Components

Number | Name | Type | CP
CO-562-A | Operating Systems | Lecture | 7.5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Jürgen Schönwälder</td><td colspan="2">Program Affiliation

• Computer Science (CS)</td><td colspan="2">Mandatory Status

Mandatory for CS and CSSE</td></tr>
<tr><td colspan="2">Entry Requirements

Pre-requisites
☒ Introduction to Computer Science
☒ Algorithms and Data Structures

Co-requisites
☒ None

Knowledge, Abilities, or Skills</td><td>Frequency

Annually (Fall)

Duration

1 semester</td><td colspan="2">Forms of Learning and Teaching

• Class attendance (online) (52.5 hours)
• Independent study (115 hours)
• Exam preparation (20 hours)

Workload

187.5 hours</td></tr>
<tr><td colspan="5">Recommendations for Preparation

Students are expected to have a working Linux installation, which allows them to compile and run sample programs provided by the instructor and to implement their own solutions for homework assignments.</td></tr>
<tr><td colspan="5">Content and Educational Aims

This module introduces concepts and principles used by operating systems to provide programming abstractions that enable an efficient and robust execution of application programs. Students will gain an understanding of how an operating system kernel manages hardware components and how it provides abstractions such as processes, threads, virtual memory, file systems, and inter-process communication facilities. Students learn the principles of event-driven and concurrent programming and the mechanisms that are necessary to solve synchronization and coordination problems, thereby avoiding race conditions, deadlocks, and resource starvation. The Linux kernel and runtime system will be used throughout the course to illustrate how key ideas and concepts have been implemented and how application programs can use them.</td></tr>
<tr><td colspan="5">Intended Learning Outcomes

By the end of this module, students will be able to

1. explain the differences between processes, threads, application programs, libraries, and operating system kernels;
2. describe well-known mutual exclusion and coordination problems;
3. use semaphores to achieve mutual exclusion and solve coordination problems;
4. use mutual exclusion locks and condition variables to solve synchronization and coordination problems;
5. illustrate how deadlocks can be avoided, detected, and resolved;
6. summarize the different mechanisms to realize virtual memory and their trade-offs;
7. solve basic inter-process communication problems using signals and pipes;</td></tr>
</table>

8. use socket inter-process communication primitives;
9. multiplex I/O activities using suitable system calls and libraries;
10. describe file system programming interfaces and the design of file systems at the operating system kernel level;
11. explain how memory mapping can improve I/O performance;
12. restate the functionality of a linker and the difference between static linking and dynamic linking;
13. outline how different device types are supported by Unix-like kernels;
14. discuss virtualization mechanisms such as containers or virtual machines.

Indicative Literature

Abraham Silberschatz, Peter B. Galvin, Greg Gagne: Applied Operating System Concepts, John Wiley, 2000.

Andrew S. Tanenbaum, Herbert Bos: Modern Operating Systems, Prentice Hall, 4th edition, Pearson, 2015.

William Stallings: Operating Systems: Internals and Design Principles, 8th edition, Pearson, 2014.

Robert Love: Linux Kernel Development, 3rd edition, Addison Wesley, 2010.

Robert Love: Linux System Programming: Talking Directly to the Kernel and C Library, 2nd edition, O'Reilly, 2013.

Usability and Relationship to other Modules

- Mandatory for a major in CS and CSSE
- Pre-requisite for the CORE module Secure and Dependable Systems
- This module enables students to write programs that make efficient use of the services provided by the operating system kernel. This is particularly important for advanced modules on computer networks, robotics, and embedded systems.

Examination Type: Module Examination

Assessment Type: Written examination

Duration: 120 min
Weight: 100%

Scope: All intended learning outcomes of the module
Module achievement: 50% of the assignments correctly solved

This module includes hands-on assignments so that students can develop their system programming skills. The module achievement ensures that a sufficient level of practical system programming skills has been obtained.

6.2.3 Data Analytics and Modeling

<table>
<tr><td colspan="3">Module Name

Data Analytics and Modeling</td><td>Module Code
CO-710</td><td>Level (type)
Year 2</td><td>CP
7.5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
CO-710-A | Data Analytics and Modeling | Lecture | 7.5</td></tr>
<tr><td>Module Coordinator

N.N.</td><td colspan="3">Program Affiliation

• Minor in Data Science</td><td colspan="2">Mandatory Status

Mandatory for Minor in Data Science and CSSE</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites | Co-requisites | Knowledge, Abilities, or Skills
☒ none | ☒ none | •</td><td>Frequency

Annually (Fall)</td><td colspan="2">Forms of Learning and Teaching

• Lectures (hybrid / online) (52.5 hours)
• Private Study (135 hours)</td></tr>
<tr><td colspan="3"></td><td>Duration

1 semester</td><td colspan="2">Workload

187.5 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

Required for solving the coding assignments are Python skills at the level achieved after successful completion of the module Introduction to Data Science. Furthermore, students are encouraged to review first-year level statistics and linear algebra.</td></tr>
<tr><td colspan="6">Content and Educational Aims

The module offers an introduction to the principles of data analytics and predictive data modeling and is structured into four parts. First, essential concepts from statistics are reviewed in the data modeling context, illustrating key ideas including randomness, distributions, and confidence regions. Examples and case studies are discussed to distinguish between proper and improper uses of statistics. Basic linear algebra is reviewed in the second part of the module, emphasizing vectors, distances, linear equations, matrices, and inversion. Key ideas such as the least squares approach are motivated with geometrical principles. The third part of the module is concerned with matrix decompositions such as the Singular Value Decomposition (SVD) and its close relatives Principal Component Analysis (PCA) and Empirical Orthogonal Function (EOF) analysis. The fourth part clarifies the distinction between linear and nonlinear modeling, and introduces key nonlinear techniques. Flexible educational formats (mostly online and hybrid) allow for asynchronous learning. Lectures are combined with Python exercises. Disciplinary applications and case studies are immersed as bridging elements.</td></tr>
</table>

Intended Learning Outcomes

Upon completion of this module, students will be able to:

1. identify important problem types and solution approaches in data analytics,
2. understand how key concepts from statistics and linear algebra enter data science,
3. explain matrix decompositions and their usage in data science,
4. discuss regularization concepts and optimality criteria in data analytics,
5. know the basics of nonlinear modeling and related computational approaches,
6. convert data structures to Python/NumPy arrays for usage in data modeling,
7. apply Python statistics and linear algebra tools in data analytics and modeling.

Indicative Literature

- Ani Adhikari, John DeNero, David Wagner. Computational and Inferential Thinking: The Foundations of Data Science. Originally developed for the UC Berkeley course Data 8: Foundations of Data Science. An online version of the textbook is available at https://inferentialthinking.com/.
- Steven S. Skiena. The Data Science Design Manual. Springer 2017.
- Gilbert Strang: Linear Algebra and Learning from Data. Wellesley-Cambridge 2019. See https://math.mit.edu/~gs/learningfromdata/.
- Joe Suzuki: Statistical Learning with Math and Python. Springer 2021.
- Jake Vanderplas. Python Data Science Handbook. O'Reilly 2016. An online version is available at https://jakevdp.github.io/PythonDataScienceHandbook/.

Usability and Relationship to other Modules

-

Examination Type: Module Examination

Type: Written Examination — Duration/Length: 180 min
Scope: All intended learning outcomes of the module. — Weight: 100 %

Module achievement: 50% of the assignments need to be correctly solved.

6.2.4 Probability and Random Processes

<table>
<tr><td colspan="3">Module Name
Probability and Random Processes</td><td>Module Code
JTMS-MAT-12</td><td>Level (type)
Year 2 (Methods)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
JTMS-12 | Probability and random processes | Lecture | 5</td></tr>
<tr><td>Module Coordinator

Dr. Keivan Mallahi Karai, Prof. Dr. Tobias Preußer</td><td colspan="3">Program Affiliation

• Jacobs Track – Methods and Skills</td><td colspan="2">Mandatory Status

Mandatory for CS, CSSE, ECE, MATH, Physics and RIS
Mandatory elective for EES</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites
☒ Calculus and Elements of Linear Algebra I & II

Co-requisites
☒ None

Knowledge, Abilities, or Skills
• Knowledge of calculus at the level of a first year calculus module (differentiation, integration with one and several variables, trigonometric functions, logarithms and exponential functions).
• Knowledge of linear algebra at the level of a first year university module (eigenvalues and eigenvectors, diagonalization of matrices).
• Some familiarity with elementary probability theory at the high school level.</td><td>Frequency

Annually (Fall)

Duration

1 semester</td><td colspan="2">Forms of Learning and Teaching

• Lectures (online) (35 hours)
• Private study (90 hours)

Workload

125 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

Review all of the first year calculus and linear algebra modules as indicated in “Entry Requirements – Knowledge, Ability, or Skills” above.</td></tr>
<tr><td colspan="6">Content and Educational Aims

This module aims to provide a basic knowledge of probability theory and random processes suitable for students in engineering, Computer Science, and Mathematics. The module provides students with basic skills needed for formulating real-world problems dealing with randomness and probability in mathematical language, and methods for applying a toolkit to solve these problems. Mathematical rigor is used where appropriate. A more advanced treatment of the subject is deferred to the third-year module Stochastic Processes.

The lecture comprises the following topics</td></tr>
</table>

- Brief review of number systems, elementary functions, and their graphs
- Outcomes, events and sample space.
- Combinatorial probability.
- Conditional probability and Bayes' formula.
- Binomials and Poisson-Approximation
- Random Variables, distribution and density functions.
- Independence of random variables.
- Conditional Distributions and Densities.
- Transformation of random variables.
- Joint distribution of random variables and their transformations.
- Expected Values and Moments, Covariance.
- High dimensional probability: Chebyshev and Chernoff bounds.
- Moment-Generating Functions and Characteristic Functions,
- The Central limit theorem.
- Random Vectors and Moments, Covariance matrix, Decorrelation.
- Multivariate normal distribution.
- Markov chains, stationary distributions.

Intended Learning Outcomes

By the end of the module, students will be able to

1. command the methods described in the content section of this module description to the extent that they can solve standard text-book problems reliably and with confidence;
2. recognize the probabilistic structures in an unfamiliar context and translate them into a mathematical problem statement;
3. recognize common mathematical terminology used in textbooks and research papers in the quantitative sciences, engineering, and mathematics to the extent that they fall into the content categories covered in this module.

Indicative Literature

J. Hwang and J.K. Blitzstein (2019). Introduction to Probability, second edition. London: Chapman & Hall.

S. Ghahramani. Fundamentals of Probability with Stochastic Processes, fourth edition. Upper Saddle River: Prentice Hall.

Usability and Relationship to other Modules

- The module is a mandatory / mandatory elective module of the Methods and Skills area that is part of the Jacobs Track (Methods and Skills modules; Community Impact Project module; Language modules; Big Questions modules).
- Students taking this module are expected to be familiar with basic tools from calculus and linear algebra.
- Mandatory for a major in CS, CSSE, ECE, MATH, Physics and RIS.
- Mandatory elective for a major in EES (if pre-requisites are met).
- Elective for all other study programs.

Examination Type: Module Examination

Assessment type: Written examination — Duration: 120 min, Weight: 100%

Scope: All intended learning outcomes of this module

6.2.5 Software Engineering

<table>
<tr><td colspan="3">Module Name
Software Engineering</td><td>Module Code
CO-561</td><td>Level (type)
Year 2 (CORE)</td><td>CP
7.5</td></tr>
<tr><td colspan="6">Module Component

Number | Name | Type | CP</td></tr>
<tr><td>CO-561-A</td><td colspan="3">Software Engineering</td><td>Lecture</td><td>2.5</td></tr>
<tr><td>CO-561-B</td><td colspan="3">Software Engineering Project</td><td>Project</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Peter Baumann</td><td colspan="3">Program Affiliation

• Computer Science (CS)</td><td colspan="2">Mandatory Status

Mandatory for CS and CSSE
Mandatory elective for RIS</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites: ☒ Databases and Web Services
Co-requisites: ☒ None
Knowledge, Abilities, or Skills</td><td>Frequency

Annually
(Spring)

Duration

1 semester</td><td colspan="2">Forms of Learning and Teaching

• Class attendance (online) (35 hours)
• Independent study (10 hours)
• Development work (132.5 hours)
• Exam preparation (10 hours)

Workload

187.5 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

Students are expected to be able to develop software using an object-oriented programming language such as C++, and they should have access to a Linux system and associated software development tools.</td></tr>
<tr><td colspan="6">Content and Educational Aims

This module is an introduction to software engineering and object-oriented software design. The lecture focuses on software quality and the methods to achieve and maintain it in environments of "multi-person construction of multi-version software." Based on their pre-existing knowledge of an object-oriented programming language, students are familiarized with software architectures, design patterns and frameworks, software components and middleware, Unified Modeling Language (UML)-based modelling, and validation by testing. Furthermore, the course addresses the more organizational topics of project management and version control.</td></tr>
</table>

The lectures are accompanied by a software project in which students have to develop a software solution to a given problem. The problem is described from the viewpoint of a customer and students working in teams have to execute a whole software project lifecycle. The teams have to create a suitable software architecture and software design, implement the components, and integrate the components. The teams have to ensure that basic quality requirements for the solution and the components are defined and satisfied. The students produce various artifacts such as design documents, source code, test cases and user documentation. All artifacts need to be maintained in a version control system and the commits should allow the instructor and other team members to track in a meaningful way the changes and who has been contributing them.

Intended Learning Outcomes

By the end of this module, students will be able to

1. understand and apply object-oriented design patterns;
2. read and write UML diagrams;
3. contrast the benefits and drawbacks of different software development models;
4. design and plan a larger software project involving a team development effort;
5. translate requirements formulated by a customer into computer science terminology;
6. evaluate the applicability of different software engineering models for a given software development project;
7. assess the quality of a software design and its implementation;
8. apply tools that assist in the various stages of a software development process;
9. work effectively in a team toward the goals of the team.

Indicative Literature

Ian Sommerville: Software Engineering, Pearson, 2010.

Roger Pressman: Software Engineering – a Practitioner's Approach, McGraw-Hill, 2014.

Usability and Relationship to other Modules

- Mandatory for a major in CS and CSSE
- Mandatory for a minor in CS
- Serves as mandatory elective 3rd year Specialization module for RIS major students.
- Pre-requisite for the CORE module Image Processing

Examination Type: Module Component Examinations

Module Component 1: Lecture

Assessment Type: Written examination Duration: 60 min
Weight: 33%

Scope: The first three intended learning outcomes of the module (the lecture module component)

Module Component 2: Project

Assessment Type: Project
Weight: 66%

Scope: The remaining intended learning outcomes of the module (the project module component)

Completion: To pass this module, the examination of each module component has to be passed with at least 45%.

6.2.6 Artificial Intelligence (CSSE)

<table>
<tr><td colspan="3">Module Name
Artificial Intelligence (CSSE)</td><td>Module Code
xxx</td><td>Level (type)
Year 2 (CORE)</td><td>CP
7.5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
CO-547-A | Artificial Intelligence | Lecture | 5
xxx | Artificial Intelligence Tutorial (CSSE) | Tutorial | 2.5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Andreas Birk</td><td colspan="3">Program Affiliation

• Lecture: Robotics and Intelligent Systems (RIS)
• Tutorial: Computer Science and Software Engineering (CSSE)</td><td colspan="2">Mandatory Status

Lecture: Mandatory for RIS & CSSE, Mandatory elective for CS
Tutorial: Mandatory for CSSE</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites
☒ Programming in C/C++
☒ Introduction to RIS
OR
☒ Introduction to CPS

Co-requisites
☒ None

Knowledge, Abilities, or Skills</td><td>Frequency

Annually (Spring)

Duration

1 semester</td><td colspan="2">Forms of Learning and Teaching

• Lectures (online) (35 hours)
• Private study (115 hours)
• Tutorials (17.5 hours)
• Exam preparation (20 hours)

Workload

187.5 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

Revise content of the pre-requisite modules.</td></tr>
<tr><td colspan="6">Content and Educational Aims

Artificial Intelligence (AI) is an important subdiscipline of Computer Science that deals with technologies to automate the performance of tasks that are usually associated with intelligence. AI methods have a significant application potential, as there is an increasing interest and need to generate artificial systems that can carry out complex missions in unstructured environments without permanent human supervision. The module teaches a selection of the most important methods in AI. In addition to general-purpose techniques and algorithms, it also includes aspects of methods that are especially targeted for physical systems such as intelligent mobile robots or autonomous cars. The AI lecture is complemented in this module by an online tutorial where the application-oriented side of software development in the context of AI is considered.</td></tr>
<tr><td colspan="6">Intended Learning Outcomes

By the end of this module, students should be able to

• outline and explain the history, general developments, and application areas of AI;
• apply the basic concepts and methods of behavior-oriented AI;
• use concepts and methods of search algorithms for problem-solving;
• explain the basic concepts of path-planning as an application example for domain-specific search;
• apply basic path-planning algorithms and to compare their relations to general search algorithms;
• write and explain concepts of propositional and first-order logic;
• use logic representations and inference for basic examples of artificial planning systems;</td></tr>
</table>

- apply AI concepts and methods to develop software.

Indicative Literature

S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2009.

S. M. LaValle, Planning Algorithms. Cambridge University Press, 2006.

J.-C. Latombe, Robot Motion Planning, Springer, 1991.

Usability and Relationship to other Modules

- This module gives an introduction to Artificial Intelligence (AI) excluding the aspects of machine learning (ML), which are covered in a dedicated module that complements this one.
- Mandatory for a major in CSSE

Module Component Examinations

Module Component 1: Lecture
Assessment Type: Written examination — Duration/length: 60 min
Weight: 50%

Scope: Intended Learning Outcomes 1-7.

Module Component 2: Tutorial
Assessment Type: Assignments
Weight: 50%

Scope: Intended Learning Outcomes 1-8.

Completion: To pass this module, the examination of each module component has to be passed with at least 45%

6.2.7 Machine Learning

<table>
<tr><td colspan="3">Module Name
Machine Learning</td><td>Module Code
CO-541</td><td>Level (type)
Year 2 (CORE)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
CO-541-A | Machine Learning | Lecture | 5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Peter Zaspel</td><td colspan="3">Program Affiliation

• Robotics and Intelligent Systems (RIS)</td><td colspan="2">Mandatory Status

Mandatory for RIS and CSSE

Mandatory elective for CS</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites
☒ None

Co-requisites
☒None

Knowledge, Abilities, or Skills
• Knowledge and command of probability theory and methods, as in the module “Probability and Random Process (JTMS-12)</td><td>Frequency

Annually

(Spring)

Duration

1 semester</td><td colspan="2">Forms of Learning and Teaching

• Class attendance (35 hours)
• Private study (70 hours)
• Exam preparation (20 hours)

Workload

125 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

None</td></tr>
<tr><td colspan="6">Content and Educational Aims

Machine learning (ML) concerns algorithms that are fed with (large quantities of) real-world data, and which return a compressed “model” of the data. An example is the “world model” of a robot; the input data are sensor data streams, from which the robot learns a model of its environment, which is needed, for instance, for navigation. Another example is a spoken language model; the input data are speech recordings, from which ML methods build a model of spoken English; this is useful, for instance, in automated speech recognition systems. There exist many formalisms in which such models can be cast, and an equally large diversity of learning algorithms. However, there is a relatively small number of fundamental challenges that are common to all of these formalisms and algorithms. The lectures introduce such fundamental concepts and illustrate them with a choice of elementary model formalisms (linear classifiers and regressors, radial basis function networks, clustering, online adaptive filters, neural networks, or hidden Markov models). Furthermore, the lectures also (re-)introduce required mathematical material from probability theory and linear algebra.</td></tr>
</table>

Intended Learning Outcomes

By the end of this module, students should be able to

1. understand the notion of probability spaces and random variables;
2. understand basic linear modeling and estimation techniques;
3. understand the fundamental nature of the "curse of dimensionality;"
4. understand the fundamental nature of the bias-variance problem and standard coping strategies;
5. use elementary classification learning methods (linear discrimination, radial basis function networks, multilayer perceptrons);
6. implement an end-to-end learning suite, including feature extraction and objective function optimization with regularization based on cross-validation.

Indicative Literature

T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edition, Springer, 2008.

S. Shalev-Shwartz, Shai Ben-David: Understanding Machine Learning, Cambridge University Press, 2014.

C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

T.M. Mitchell, Machine Learning, Mc Graw Hill India, 2017.

Usability and Relationship to other Modules

- Mandatory for a major in RIS
- Mandatory for a minor in RIS
- This module serves as a third Year Specialization module for CS major students.
- This module gives a thorough introduction to the basics of machine learning. It complements the Artificial Intelligence module.

Examination Type: Module Examination

Assessment Type: Written examination — Duration: 120 min
Weight: 100%

Scope: All intended learning outcomes of the module

6.2.8 Machine Learning Tools

<table>
<tr><td colspan="3">Module Name
Machine Learning Tools</td><td>Module Code
xxx</td><td>Level (type)
Year 2</td><td>CP
2.5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
xxx | Machine Learning Tools | Lab | 2.5</td></tr>
<tr><td>Module Coordinator
Prof. Dr. Peter Zaspel</td><td colspan="3">Program Affiliation

• BSc Computer Science and Software Engineering (CSSE)</td><td colspan="2">Mandatory Status

Mandatory for CSSE</td></tr>
<tr><td colspan="3" rowspan="2">Entry Requirements

Pre-requisites
☒ none

Co-requisites
☒ Machine Learning

Knowledge, Abilities, or Skills
•</td><td>Frequency

Annually (Spring)</td><td colspan="2">Forms of Learning and Teaching

• Self study of online material (17.50 hours)
• Lab meetings (online) (8.75 hours)
• Lab assignments (36.25 hours)</td></tr>
<tr><td>Duration

1 semester</td><td colspan="2">Workload

62.5 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation
None</td></tr>
<tr><td colspan="6">Content and Educational Aims

Modern machine learning in industry and research requires the knowledge of a comprehensive stack of tools and systems that allow to store and administrate data (e.g. Amazon S3, Kaggle, Dataverse, GIT LFS), extract features for various applications (e.g. Word2Vec, TSFEL), build up machine learning pipelines of training, testing, and hyperparameter optimization (e.g. skit-learn, Keras, TensorFlow, PyToarch) and ultimately deploy finalized models (e.g. TensorFlow Serving, MLFlow). This module gives exposure to a regularly updated latest state of the art set of tools that are relevant for the practical use of Machine Learning. It thereby complements the more theoretical and methods-driven module “Machine Learning” with market-oriented skills.</td></tr>
<tr><td colspan="6">Intended Learning Outcomes

By the end of this module, students will be able to
1. describe and use systems and tools to store and administrate data;
2. Explain and apply modern feature extraction libraries to real-world data;
3. understand and use standard tool chains for training, testing and hyperparameter optimization in Machine Learning;
4. deploy machine learning models.</td></tr>
<tr><td colspan="6">Indicative Literature</td></tr>
<tr><td colspan="6">Usability and Relationship to other Modules

• Mandatory module for CSSE students.</td></tr>
</table>

Examination Type: Module Examination

Assessment Type: Lab Assignments Weight: 100%

Scope: All intended learning outcomes of the module

6.3 YEAR 3

6.3.1 Computer Graphics

<table>
<tr><td colspan="3">Module Name
Computer Graphics</td><td>Module Code
CA-S-CS-801</td><td>Level (type)
Year 3
(Specialization)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
CA-CS-801 | Computer Graphics | Lecture | 5</td></tr>
<tr><td>Module Coordinator

N.N.</td><td colspan="3">Program Affiliation

• Computer Science (CS)</td><td colspan="2">Mandatory Status

Mandatory elective for CS, CSSE and RIS</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites
☒ Algorithms and Data Structures

Co-requisites
☒ None

Knowledge, Abilities, or Skills</td><td>Frequency

Annually (Fall)</td><td colspan="2">Forms of Learning and Teaching

• Class attendance (online) (35 hours)
• Private study (70 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td colspan="3"></td><td>Duration

1 semester</td><td colspan="2">Workload

125 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

None</td></tr>
<tr><td colspan="6">Content and Educational Aims

This module deals with the digital synthesis and manipulation of visual content. The creation process of computer graphics spans from the creation of a three-dimensional (3D) scene to displaying or storing it digitally. Prominent tasks in computer graphics are geometry processing, rendering, and animation. Geometry processing is concerned with object representations such as surfaces and their modeling. Rendering is concerned with transforming a model of the virtual world into a set of pixels by applying models of light propagation and sampling algorithms. Animation is concerned with descriptions of objects that move or deform over time. This is an introductory module covering the concepts and techniques of 3D (interactive) computer graphics. It covers mathematical foundations, basic algorithms and principles, and some advanced methods and concepts. An introduction to the implementation of simple programs using a mainstream computer graphics library completes this module.</td></tr>
<tr><td colspan="6">Intended Learning Outcomes

By the end of this module, students will be able to

1. construct 3D geometry representations;
2. apply 3D transformations;
3. understand the algorithms and optimizations applied by graphics rendering systems;
4. explain the stages of modern computer graphics programmable pipelines
5. implement simple computer graphics applications using graphics frameworks such as OpenGL;
6. illustrate the techniques used to create animations.</td></tr>
<tr><td colspan="6">Indicative Literature</td></tr>
</table>

John Hughes, Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley, Computer Graphics - Principles and Practice, 3rd edition, Addison-Wesley, 2013.

Peter Shirley, Steve Marschner, Fundamentals of Computer Graphics, 4th edition, Taylor and Francis Ltd, 2016.

Matt Pharr, Wenzel Jakob, Greg Humphreys, Physically Based Rendering: From Theory to Implementation, 3rd edition, Morgan Kaufmann, 2016.

Usability and Relationship to other Modules

- Mandatory elective for a major in CS and CSSE.
- Serves as a 3rd year specialization module for RIS major students.
- Students with a strong interest in graphical user interfaces are encouraged to also select the Human–Computer Interaction specialization module, which discusses among other things how computer graphics can be used as a component of interactive graphical user interfaces.

Examination Type: Module Examination

Assessment Type: Written examination

Duration: 120 min
Weight: 100%

Scope: All intended learning outcomes of the module

6.3.2 Computer Networks

<table>
<tr><td colspan="3">Module Name
Computer Networks</td><td>Module Code
CO-564</td><td>Level (type)
Year 2 (CORE)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
CO-564-A | Computer Networks | Lecture | 5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Jürgen Schönwälder</td><td colspan="3">Program Affiliation

• Computer Science (CS)</td><td colspan="2">Mandatory Status

Mandatory Elective for CS and CSSE</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites | Co-requisites | Knowledge, Abilities, or Skills
☒ Algorithms and Data Structures | ☒ Operating Systems</td><td>Frequency

Annually (Fall)</td><td colspan="2">Forms of Learning and Teaching

• Class attendance (online) (35 hours)
• Private study (70 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td colspan="3"></td><td>Duration

1 semester</td><td colspan="2">Workload

125 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

Students are expected to be familiar with the C programming language and to learn basics of higher-level scripting languages such as Python (the official Python documentation is available on https://docs.python.org/).</td></tr>
<tr><td colspan="6">Content and Educational Aims

Computer networks such as the Internet play a critical role in today's connected world. This module discusses the technology of Internet services in depth to enable students to understand the core issues involved in the design of modern computer networks. Fundamental algorithms and principles are explained in the context of existing protocols as they are used in today's Internet. Students taking this course should finally understand the technical complexity behind everyday online services such as Google or YouTube.

Students taking this module will understand how computer networks work and they will be able to assess communication networks, including aspects such as performance but also robustness and security. Students will learn that the design of communication networks is not only influenced by technical constraints but also by the necessity to define common standards, which often requires to take engineering decisions that reflect non-technical requirements.</td></tr>
<tr><td colspan="6">Intended Learning Outcomes

By the end of this module, students will be able to

1. recall layering principles and the OSI reference model;
2. articulate the organization of the Internet and the organization involved in providing Internet services;
3. describe media access control, flow control, and congestion control mechanisms;
4. explain how local area networks differ from global networks;
5. illustrate how frames are forwarded in local area networks;
6. contrast addressing mechanisms and translations between addresses used at different layers;</td></tr>
</table>

7. demonstrate how the Internet network layer forwards packets;
8. present how routing algorithms and protocols are used to determine and select routes;
9. describe how the Internet transport layer provides different end-to-end services;
10. demonstrate how names are resolved to addresses and vice versa;
11. summarize how application layer protocols send and access electronic mail or access resources on the world-wide web;
12. design and implement simple application layer protocols;
13. recognize to which extent computer networks are fragile and evaluate strategies to cope with the fragility;
14. analyze traffic traces produced by a given computer network.

Indicative Literature

James F. Kurose, Keith W. Ross: Computer Networking: A Top-Down Approach Featuring the Internet, 3rd Edition, Addison-Wesley, 2004.

Andrew S. Tanenbaum: Computer Networks, 4th Edition, Prentice Hall, 2002.

Usability and Relationship to other Modules

- Mandatory elective module for a major in CS
- Pre-requisite for the CORE module Secure and Dependable Systems
- The module should be taken together with the module Operating Systems, because a significant portion of the communication technology is implemented at the operating system level. An understanding of operating system concepts and abstractions will help students to understand how computer network technology is commonly implemented and made available to applications. The specialization module Distributed Algorithms discusses algorithms for solving problems commonly found in distributed systems that use computer networks to exchange information. The module Secure and Dependable Systems introduces cryptographic mechanisms that can be used to secure communication over computer networks.

Examination Type: Module Examination

Assessment Type: Written examination — Duration: 120 min
Weight: 100%

Scope: All intended learning outcomes of the module

6.3.3 Web Application Development

<table>
<tr><td colspan="3">Module Name
Web Application Development</td><td>Module Code
CA-S-CS-804</td><td>Level (type)
Year 3 (Specialization)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
CA-CS-804-A | Web Application Development | Lecture | 2.5
CA-CS-804-B | Web Application Development - Project | Project | 2.5</td></tr>
<tr><td>Module Coordinator

N.N.</td><td colspan="2">Program Affiliation

• Computer Science (CS)</td><td colspan="3">Mandatory Status

Mandatory elective for CS, CSSE and RIS</td></tr>
<tr><td colspan="3" rowspan="2">Entry Requirements

Pre-requisites
☒ Databases and Web Services

Co-requisites
☒ None

Knowledge, Abilities, or Skills</td><td>Frequency

Annually (Spring)</td><td colspan="2">Forms of Learning and Teaching

• Class attendance (online) (17.5 hours)
• Private study (40 hours)
• Project work (50 hours)
• Exam preparation (17.5 hours)</td></tr>
<tr><td>Duration

1 semester</td><td colspan="2">Workload

125 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

None</td></tr>
<tr><td colspan="6">Content and Educational Aims

A web application is a client-server computer program where the client provides the user interface and the client side logic runs in a web browser or as an app running on a mobile device such as a smart phone or a tablet. A key characteristic is that more complex application logic and data storage is realized by a server offering a web application programming interface.

This module focuses on the client side of web application and introduces technologies that can be used to implement interactive user interfaces and client side logic. It builds on the module databases and web services, which covers the data storage components and server side logic of web applications.

This module consists of a lecture and an associated project. The lecture component introduces programming languages and frameworks that are widely used for implementing the client side of web applications such as Java, Kotlin, Swift, JavaScript and frameworks built on top of them. In the project component, students develop web applications and test them on existing and openly accessible web services.</td></tr>
<tr><td colspan="6">Intended Learning Outcomes

By the end of this module, students will be able to

1. explain the document object model behind HTML and its relation to CSS;
2. discuss the principles and basic mechanisms of reactive website design;</td></tr>
</table>

3. analyze the interactions between web applications and web services.
4. use languages such as Java, Kotlin, or Swift to implement mobile web applications;
5. use web standards such as HTML, CSS, and JavaScript to implement web applications running in standard web browsers.

Indicative Literature

Stoyan Stefanov: JavaScript Patterns, O'Reilly Media, 2010.

Alexey Soshin: Hands-on Design Patterns with Kotlin, Packt Publishing, 2018.

Alex Banks, Eve Porcello: Learning React: Functional Web Development.with React and Flux, O'Reilly, 2017.

Usability and Relationship to other Modules

- Mandatory elective for a major in CS and CSSE.
- Mandatory elective for a major in RIS.

Examination Type: Module Component Examinations

Module Component 1: Lecture

Assessment Type: Written examination — Duration: 120 min, Weight: 50%

Scope: First group of intended learning outcomes of the module

Module Component 2: Project

Assessment Type: Project — Weight: 50%

Scope: Second group of intended learning outcomes of the module

Completion: To pass this module, the examination of each module component has to be passed with at least 45%.

6.3.4 Human-Computer Interaction

<table>
<tr><td colspan="3">Module Name
Human Computer Interaction</td><td>Module Code
CA-S-RIS-802</td><td>Level (type)
Year 3 (Specialization)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
CA-RIS-802 | Human Computer Interaction | Lecture | 5</td></tr>
<tr><td>Module Coordinator

N.N.</td><td colspan="3">Program Affiliation

• Robotics and Intelligent Systems (RIS)</td><td colspan="2">Mandatory Status

Mandatory elective for RIS, CS and CSSE</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites: ☒ None
Co-requisites: ☒ None
Knowledge, Abilities, or Skills: • None</td><td>Frequency

Annually (Fall)</td><td colspan="2">Forms of Learning and Teaching

• Class attendance (online) (35 hours)
• Private study (70 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td colspan="3"></td><td>Duration

1 semester</td><td colspan="2">Workload

125 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation

None</td></tr>
<tr><td colspan="6">Content and Educational Aims

Computer systems often interact with human beings. The design of a good human–computer interface is often crucial for the acceptance and the success of a software system. Human–computer interface designs have to satisfy several requirements such as usability, learnability, efficiency, accessibility, and safety. The module discusses the evolution of human–computer interaction models and introduces design principles for graphical user interfaces and other types of interaction (e.g., visual, voice, gesture). Human–computer interaction designs are often evaluated using prototypes or mockups that can be given to test candidates to evaluate the effectiveness of the design. The module introduces evaluation strategies as well as tools and techniques that can be used to prototype human–computer interfaces.</td></tr>
<tr><td colspan="6">Intended Learning Outcomes

By the end of this module, students should be able to
• explain the evolution of human–computer interaction models;
• design and implement simple graphical user interfaces;
• explain ergonomic principles guiding the design of user interfaces;
• illustrate different types of interaction (e.g., visual, voice, gestures) and their usability aspects;
• evaluate aspects of and tradeoffs between usability, learnability, efficiency, and safety;
• apply scientific methods to evaluate interfaces with respect to their usability and other desirable properties;
• use prototyping tools that can be employed to create mockups of user interfaces during the early stages of a software project.</td></tr>
<tr><td colspan="6">Indicative Literature

Not specified</td></tr>
<tr><td colspan="6">Usability and Relationship to other Modules</td></tr>
</table>

- Students with a strong interest in graphical user interfaces are encouraged to also select the Computer Graphics specialization module, which introduces methods and technologies for creating computer graphics and animations.
- Mandatory elective third year Specialization module for CS, CSSE and RIS major students.

Examination Type: Module Examination

Assessment Type: Written examination — Duration: 120 min, Weight: 100%

Scope: All intended learning outcomes of the module

6.4 Internship / Startup and Career Skills

<table>
<tr><td colspan="3">Module Name
Internship / Startup and Career Skills</td><td>Module Code
CA-INT-900</td><td>Level (type)
Year 3 (CAREER)</td><td>CP
15</td></tr>
<tr><td colspan="6">Module Components

Number | Name | Type | CP
CA-INT-900-0 | Internship | Internship | 15</td></tr>
<tr><td>Module Coordinator

Sinah Vogel & Dr. Tanja Woebs (CSC Organization); SPC / Faculty Startup Coordinator (Academic responsibility)</td><td colspan="3">Program Affiliation

• CAREER module for undergraduate study programs</td><td colspan="2">Mandatory Status

Mandatory for all undergraduate study programs except IEM</td></tr>
<tr><td colspan="3">Entry Requirements

Pre-requisites
☒ at least 15 CP from CORE modules in the major

Co-requisites
☒ None

Knowledge, Abilities, or Skills
• Information provided on CSC pages (see below)
• Major specific knowledge and skills</td><td>Frequency

Annually (Spring/Fall)

Duration
1 semester</td><td colspan="2">Forms of Learning and Teaching

• Internship/Start-up
• Internship event
• Seminars, info-sessions, workshops and career events
• Self-study, readings, online tutorials

Workload

375 Hours consisting of:
• Internship (308 hours)
• Workshops (33 hours)
• Internship Event (2 hours)
• Self-study (32 hours)</td></tr>
<tr><td colspan="6">Recommendations for Preparation

• Please see the section “Knowledge Center” at JobTeaser Career Center for information on Career Skills seminar and workshop offers and for online tutorials on the job market preparation and the application process. For more information, please see https://www.jacobs-university.de/employability/career-services
• Participating in the internship events of earlier classes</td></tr>
<tr><td colspan="6">Content and Educational Aims

The aims of the internship module are reflection, application, orientation, and development: for students to reflect on their interests, knowledge, skills, their role in society, the relevance of their major subject to society, to apply these skills and this knowledge in real life whilst getting practical experience, to find a professional orientation, and to develop their personality and in their career. This module supports the programs’ aims of preparing students for gainful, qualified employment and the development of their personality.

The full-time internship must be related to the students’ major area of study and extends lasts a minimum of two consecutive months, normally scheduled just before the 5th semester, with the internship event and submission of the internship report in the 5th semester. Upon approval by the SPC and CSC, the internship may take place at other</td></tr>
</table>

times, such as before teaching starts in the 3rd semester or after teaching finishes in the 6th semester. The Study Program Coordinator or their faculty delegate approves the intended internship a priori by reviewing the tasks in either the Internship Contract or Internship Confirmation from the respective internship institution or company. Further regulations as set out in the Policies for Bachelor Studies apply.

Students will be gradually prepared for the internship in semesters 1 to 4 through a series of mandatory information sessions, seminars, and career events.
The purpose of the Career Services Information Sessions is to provide all students with basic facts about the job market in general, and especially in Germany and the EU, and services provided by the Career Services Center.
In the Career Skills Seminars, students will learn how to engage in the internship/job search, how to create a competitive application (CV, Cover Letter, etc.), and how to successfully conduct themselves at job interviews and/or assessment centers. In addition to these mandatory sections, students can customize their skill set regarding application challenges and their intended career path in elective seminars.
Finally, during the Career Events organized by the Career Services Center (e.g. the annual Jacobs Career Fair and single employer events on and off campus), students will have the opportunity to apply their acquired job market skills in an actual internship/job search situation and to gain their desired internship in a high-quality environment and with excellent employers.

As an alternative to the full-time internship, students can apply for the StartUp Option. Following the same schedule as the full-time internship, the StartUp Option allows students who are particularly interested in founding their own company to focus on the development of their business plan over a period of two consecutive months. Participation in the StartUp Option depends on a successful presentation of the student's initial StartUp idea. This presentation will be held at the beginning of the 4th semester. A jury of faculty members will judge the student's potential to realize their idea and approve the participation of the students. The StartUp Option is supervised by the Faculty StartUp Coordinator. At the end of StartUp Option, students submit their business plan. Further regulations as outlined in the Policies for Bachelor Studies apply.

The concluding Internship Event will be conducted within each study program (or a cluster of related study programs) and will formally conclude the module by providing students the opportunity to present on their internships and reflect on the lessons learned within their major area of study. The purpose of this event is not only to self-reflect on the whole internship process, but also to create a professional network within the academic community, especially by entering the Alumni Network after graduation. It is recommended that all three classes (years) of the same major are present at this event to enable networking between older and younger students and to create an educational environment for younger students to observe the "lessons learned" from the diverse internships of their elder fellow students.

Intended Learning Outcomes

By the end of this module, students should be able to

1. describe the scope and the functions of the employment market and personal career development;
2. apply professional, personal, and career-related skills for the modern labor market, including self-organization, initiative and responsibility, communication, intercultural sensitivity, team and leadership skills, etc.;
3. independently manage their own career orientation processes by identifying personal interests, selecting appropriate internship locations or start-up opportunities, conducting interviews, succeeding at pitches or assessment centers, negotiating related employment, managing their funding or support conditions (such as salary, contract, funding, supplies, work space, etc.);
4. apply specialist skills and knowledge acquired during their studies to solve problems in a professional environment and reflect on their relevance in employment and society;
5. justify professional decisions based on theoretical knowledge and academic methods;
6. reflect on their professional conduct in the context of the expectations of and consequences for employers and their society;
7. reflect on and set their own targets for the further development of their knowledge, skills, interests, and values;
8. establish and expand their contacts with potential employers or business partners, and possibly other students and alumni, to build their own professional network to create employment opportunities in the future;
9. discuss observations and reflections in a professional network.

Indicative Literature

Not specified

Usability and Relationship to other Modules

- Mandatory for a major in BCCB, CBT, CS, EES, GEM, IBA, IRPH, ISCP, Math, MCCB, Physics, RIS, and SMP.
- This module applies skills and knowledge acquired in previous modules to a professional environment and provides an opportunity to reflect on their relevance in employment and society. It may lead to thesis topics.

Examination Type: Module Examination

Assessment Type: Internship Report or Business Plan and Reflection	Length: approx. 3.500 words
Scope: All intended learning outcomes	Weight: 100%

6.5 Collaborative Software Project

Module Name	*Module Code*	*Level (type)*	*CP*
Collaborative Software Project		Year 3	5

Module Components

Number	*Name*	*Type*	*CP*
xxx	Collaborative Software Project	Project	5

Module Coordinator	*Program Affiliation*	*Mandatory Status*
NN	• BSc Computer Science and Software Engineering	Mandatory for CSSE

Entry Requirements			*Frequency*	*Forms of Learning and Teaching*
Pre-requisites	*Co-requisites*	*Knowledge, Abilities, or Skills*	Annually Fall	• Meetings with the instructor and within the group (online) (45 hours) • Independent project work (80 hours)
☒ Students must have successfully passed 90 CP.	☒ none			
			Duration	*Workload*
			1 semester	125 hours

Recommendations for Preparation

Content and Educational Aims

The project enables the students to deepen their knowledge and skills in one or multiple areas of the 1st and especially 2nd year. They are exposed to state-of-the-art research with the goal to derive ideas and strategies to address application-oriented problems and to develop software for them. Students learn how to organize and execute an application-oriented research and development (R&D) project and how to present the results in the format of a white-paper. Students are expected to organize themselves in group work under the guidance of the instructor.

Intended Learning Outcomes

Upon completion of this module, students will be able to:

1. Understand state-of-the-art research papers in a chosen field of specialization
2. Plan a research project to reproduce research results or to extend ideas of recent research results
3. Explain research questions and choose suitable methodologies to address them
4. Use methods and tools for remote collaborative software development
5. Document a research project in the style of a typical white-paper

Indicative Literature

- State-of-the-art literature provided by the instructor

Usability and Relationship to other Modules

- The module serves as a mandatory module for CSSE students.

Examination Type: Module Examination

Assessment: Project report (4,000 words) Weight: 100%

Scope: All intended learning outcomes of the module.

6.6 Bachelor Thesis

<table>
<tr><td colspan="2">Module Name
Bachelor Thesis</td><td>Module Code
xxxxxxxxxxxxxx</td><td>Level (type)
Year 3 (CAREER)</td><td>ECTS
10</td></tr>
<tr><td colspan="5">Module Components

Number | Name | Type | ECTS
Xxxxxxxxxxxxxx | Thesis | Thesis | 10</td></tr>
<tr><td>Module Coordinator
Study Program Chair</td><td colspan="2">Program Affiliation
• all Bachelor Programs</td><td colspan="2">Mandatory Status
Mandatory for all Bachelor Programs</td></tr>
<tr><td colspan="3">Entry Requirements
Pre-requisites
☒ Students must be in their third year and have taken at least 30 CP from Year 2 modules.

Co-requisites
☒ None

Knowledge, Abilities, or Skills
• Comprehensive knowledge of the subject and deeper insight into the chosen topic;
• ability to plan and undertake work independently;
• skills to identify and critically review literature.</td><td>Frequency
annually

Duration
1 semester</td><td>Forms of Learning and Teaching
• Self-study/lab work (350 hours)
• Seminars (25 hours)

Workload
250 hours</td></tr>
<tr><td colspan="5">Recommendations for Preparation
• Identify an area or a topic of interest and discuss this with your prospective supervisor in good time.
• Create a research proposal including a research plan to ensure timely submission.
• Ensure you possess all required technical research skills or are able to acquire them on time.
• Review again the University's Code of Academic Integrity and Guidelines to Ensure Good Academic Practice.</td></tr>
</table>

Content and Educational Aims

This module is a mandatory graduation requirement for all undergraduate students to demonstrate their ability to deal with a problem from their respective major subject independently by means of academic/scientific methods within a set period. Although supervised, the module requires the student to be able to work independently and regularly and set their own goals in exchange for the opportunity to explore a topic that excites and interests them personally and which a faculty member is interested to supervise. Within this module, students apply their acquired knowledge about the major discipline, skills, and methods to conduct research, ranging from the identification of suitable (short-term) research projects, preparatory literature searches, the realization of discipline-specific research, and the documentation, discussion, interpretation and communication of the results.

This module consists of two components, an independent thesis and an accompanying seminar. The thesis component must be supervised by a Jacobs University faculty member and requires short-term research work, the results of which must be documented in a comprehensive written thesis including an introduction, a justification of the methods, results, a discussion of the results, and conclusions. The seminar provides students with the opportunity to present, discuss and justify their and other students' approaches, methods and results at various stages of their research to practice these skills to improve their academic writing, receive and reflect on formative feedback, thereby growing personally and professionally.

Intended Learning Outcomes

On completion of this module, students will be able to

1. independently plan and organize advanced learning processes;
2. design and implement appropriate research methods taking full account of the range of alternative techniques and approaches;
3. collect, assess and interpret relevant information;
4. draw scientifically founded conclusions that consider social, scientific and ethical insights;
5. apply their knowledge and understanding to a context of their choice;
6. develop, formulate and advance solutions to problems and arguments in their subject area, and defend these through argument;
7. discuss information, ideas, problems and solutions with specialists and non-specialists;

Usability and Relationship to other Modules

- This module builds on all previous modules of the program. Students apply the knowledge, skills and competencies they acquired and practiced during their studies, including research methods and the ability to acquire additional skills independently as and if required.

Assessment

Type: Thesis

Length: approx. 6.000 – 8.000 words (15 – 25 pages), excluding front- and back matter.

Scope: All intended learning outcomes, mainly 1-6.

Weight: 80%

Assessment

Type: Presentation

Duration: approx. 15 to 30 minutes

Scope: The presentation focusses mainly on ILOs 6 and 7, but by nature of these ILOs also touches on the others.

Weight: 20%

7 Appendix

7.1 Intended Learning Outcomes Assessment Matrix

Computer Science and Software Engineering (BSc.)					Introduction to Computer Science	Programming in C and C++	Introduction to Data Science	Calculus and Elements of Linear Algebra I	Algorithms and Data Structures	Introduction to Cyber Physical Systems	Software Design and Prototyping	Calculus and Elements of Linear Algebra II	Distributed Development	Databases and Web Services	Operating Systems	Data Analytics	Probability and Random Processes	Software Engineering	Artificial Intelligence	Machine Learning + ML Tools	Internship/Start-Up	CSSE Specialization	Management Modules	Collaborative Software Project	Bachelor Thesis
Semester					1	1	1	1	2	2	2	2	1-2	3	3	3	3	4	4	4	5	5-6	5-6	5	4
Mandatory/ optional					m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	me	me	m	m
Credits					7.5	7.5	7.5	5	7.5	7.5	7.5	5	5	7.5	7.5	7.5	5	7.5	7.5	7.5	15	15	10	5	30
	Competencies*																								
Program Learning Outcomes	**A**	**E**	**P**	**S**																					
acquire Computer Science and Software Engineering knowledge in an independent, self-governed way	x	x	x		x	x	x	x	x	x	x					x	x	x	x	x	x			x	x
Work in teams distributed around the globe to analyze complex problems, to evaluate them, and to derive solutions	x	x	x		x	x	x	x	x	x	x					x	x			x	x			x	x
Comprehend the processes and tools of Software Engineering for collaborative, remote software and systems development	x	x						x	x	x	x					x			x		x			x	x
Program software in C/C++ and understand algorithms;	x	x							x	x								x			x			x	x
Be able to use libraries and to generate software in core Computer Science areas	x	x	x					x	x	x			x		x	x					x				x
Apply suited mathematical methods	x	x						x		x	x	x	x	x	x	x					x				x
Understand operating systems, databases, and web applications	x	x						x	x	x	x	x	x	x		x					x				x
Comprehend methods from Artificial Intelligence and Machine Learning	x	x		x								x	x	x	x	x					x			x	x
Understand the relation between software and its links to the physical world	x	x		x			x			x						x			x	x				x	x
analyze data and to extract insights from it	x	x		x			x	x	x			x				x	x		x	x				x	x
apply the acquired Software Engineering skills and Computer Science knowledge in collaborative, remote projects	x	x	x								x		x	x				x	x	x				x	x
Use academic or scientific methods as appropriate in the field of Computer Science and Software Engineering such as defining research questions, justifying methods, collecting, assessing and interpreting relevant information, and drawing scientifically-founded conclusions that consider social, scientific and ethical insights;	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x
Develop and advance solutions to problems and arguments in their subject area and defend these in discussions with specialists and non-specialists;		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x
Engage ethically with academic, professional and wider communities and to actively contribute to a sustainable future, reflecting and respecting different views;		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x
Take responsibility for their own learning, personal and professional development and role in society, evaluating critical feedback and self-analysis;		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x
Apply their knowledge and understanding to a professional context;	x	x	x					x	x	x	x		x		x	x					x			x	x
Take on responsibility in a diverse team;		x	x	x				x	x	x	x	x	x	x	x	x					x			x	x
Adhere to and defend ethical, scientific and professional standards.		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x
Assessment Type																									
Oral examination																									
Written examination					x		x	x	x	x	x	x		x	x		x	x	x	x						
Project											x			x				x							
Term paper																									
Report																x			x	x	x			x	
Poster presentation																									
Presentation																									
Various						x							x									x	x		
Thesis																									x

*Competencies: A-scientific/academic proficiency; E-competence for qualified employment; P-development of personality; S-competence for engagement in society

Figure 3: ILO Assessment Matrix