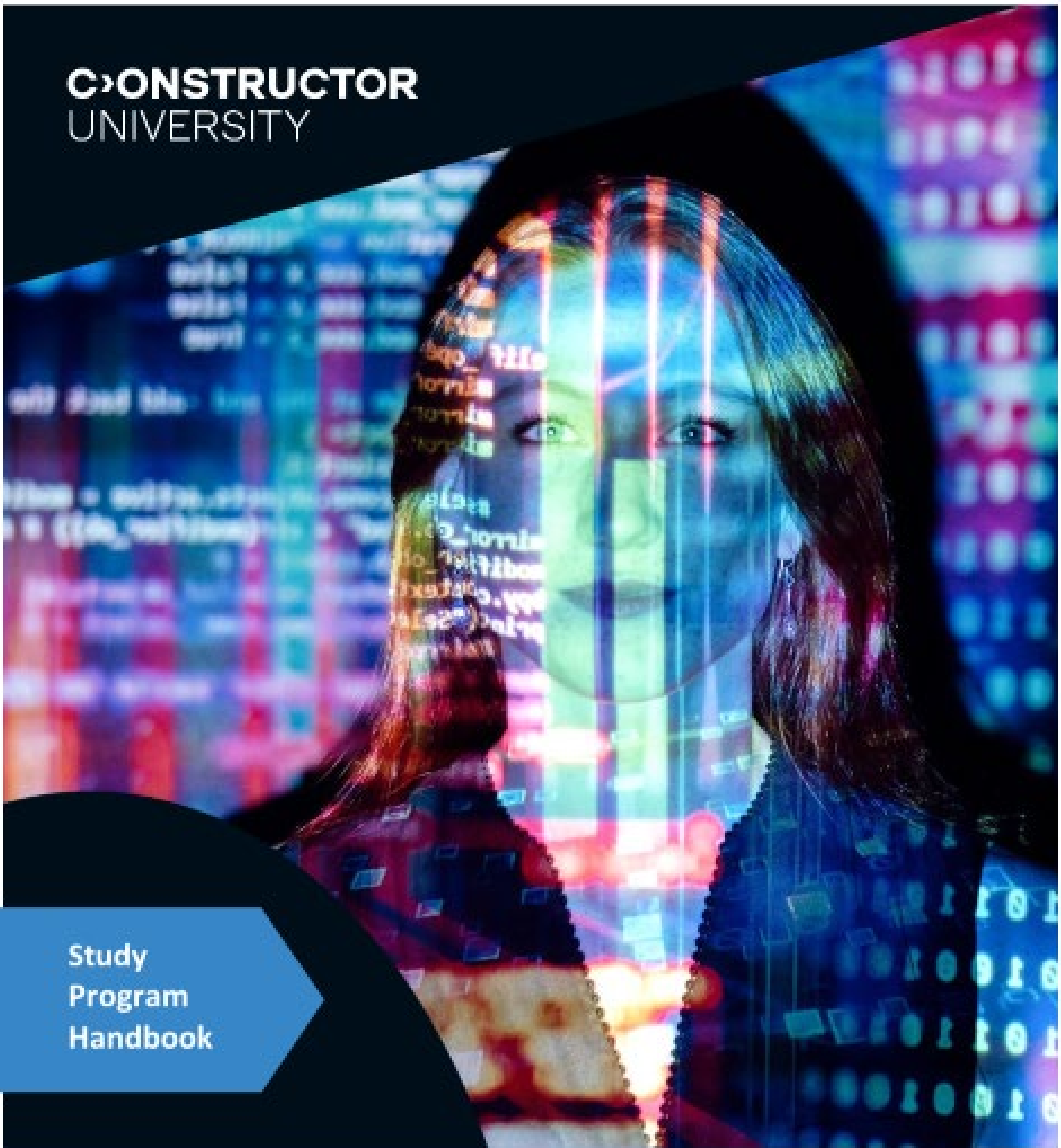
CONSTRUCTOR
UNIVERSITY



Study Program Handbook

Software, Data and Technology

Bachelor of Science

Subject-specific Examination Regulations for Software, Data and Technology

(Fachspezifische Prüfungsordnung)

The subject-specific examination regulations for Software, Data and Technology 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 4 of this handbook).

Version	Valid as of	Decision	Details
Fall 2023 – V1.2	Sep 01, 2023	Aug 03, 2023	Editorial change of all study schemes by Program Support and Development
Fall 2023 – V1.1		June 28, 2023	Academic Senate approval of study program name change from “Data Science and Software Development” to “Software, Data and Technology”
		May 26, 2023	Editorial changes in all handbooks by Program Support and Development
Fall 2023 – V1		May 24, 2023	Originally approved by Academic Senate

Contents

1 Program Overview

1.1 Concept

1.1.1 The Constructor University Educational Concept

Constructor University aims to educate students for both an academic and a professional career by emphasizing three core objectives: academic excellence, personal development, and employability to succeed in the working world. Constructor University offers an excellent research driven education experience across disciplines to prepare students for graduate education as well as career success by combining disciplinary depth and interdisciplinary breadth with supplemental skills education and extra-curricular elements. Through a multi-disciplinary, wholistic approach and exposure to cutting-edge technologies and challenges, Constructor University develops and enables the academic excellence, intellectual competences, societal engagement, professional and scientific skills of tomorrows leaders for a sustainable and peaceful future.

In this context, it is Constructor 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 for the democratic, peaceful, and sustainable development of the societies in which they live. This is achieved through a high-quality teaching as well as manageable study loads and supportive study conditions. Study programs and related study abroad programs 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 Constructor University, both in terms of actual disciplinary subject matter and also to the social skills and intercultural competence. Study-program-specific modules and additional specializations provide the necessary depth, interdisciplinary offerings and the minor option provide breadth while the university-wide general foundation and methods modules, optional German language and Humanity modules, and an extended internship period strengthen the employability of students. The concept of living and learning together on an international campus with many cultural and social activities supplements students' education. In addition, Constructor University offers professional advising and counseling.

Constructor 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 and 2021. Since 2022 Constructor University is considered to be among the top 30 percent out of more than 1600 universities worldwide and is ranked the most international university in Germany. 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

Software, Data and Technology are at the forefront of modern industries and play a major role in most areas of science and technology. The field is constantly evolving, but the fundamental principles underlying these technologies have now developed into a mature discipline. The BSc Software, Data and Technology program at Constructor University focuses on the understanding of these principles and their application in practice.

Students will obtain core software, data and technology competencies and skills (e.g., programming, data analysis, and machine learning) and they will learn about fundamental abstractions and abstract notions of software, data and technology (e.g., data structures, algorithms, and software design principles). They will learn about the principles behind and the proper usage of core technologies (e.g., databases, parallel programming, compilers, and data analysis). Finally, students will develop an understanding of the limitations of technology and side effects of software, data and technology systems (e.g., security, privacy, and ethical aspects).

Because software, data and technology are rooted in mathematics and computer science, students will take mathematical and computer science methods modules covering calculus, linear algebra, probability theory, statistics, and numerical methods or discrete mathematics.

The job market for computer scientists has been very favorable in the last few years, and there is no indication that this will change in the near future. Because of the rapid changes in the field, it is important to focus the education on the fundamental principles, as well as, subfields of promising future relevance. Cross-disciplinary breadth and flexibility, as well as social and work organization skills are increasingly important. The program offers a major option in Software, Data and Technology designed for students who plan to work in the information technology industry or join graduate programs related to the discipline.

In summary, the BSc Software, Data and Technology program at Constructor University is designed to provide students with the foundational knowledge, skills and understanding of the principles and application of software, data and technology in modern industries. With an emphasis on cross-disciplinary breadth and flexibility, students will be well-prepared for a wide range of career opportunities in the field.

1.2 Specific Advantages of Software, Data and Technology at Constructor University

The Software, Data and Technology program at Constructor University aims to provide students with a comprehensive and rigorous education in the foundations of software, data and technology, while also keeping the curriculum contemporary and internationally oriented.

- The program will focus on relating the theoretical concepts to their practical application in industry and research, with instructors incorporating recent developments and trends in the field to demonstrate how basic methods and techniques are being used today.
- Early involvement in research projects will be an integral aspect of the program, providing students with the opportunity to gain hands-on experience and potentially develop interdisciplinary collaborations.
- The program will be constantly fine-tuned through direct and open dialogue with students and alumni, ensuring that the curriculum meets their specific needs and prepares them for internships and job opportunities.

- One of the specific advantages of the program is its carefully designed curriculum with a diverse range of course offerings, providing students with a well-rounded understanding of the field and preparing them for various career paths. The program covers foundational subjects such as Linear Algebra, Analysis, Programming in Python and C++, Core and Advanced Algorithms & Data Structures, as well as more specialized subjects. These courses are structured to ensure a progressive learning experience, with advanced modules building on the knowledge acquired in earlier modules.
- In addition to the core curriculum, the program also offers three specialized tracks: Machine Learning, Software Development, and Programming Languages. These specialized tracks provide students with the opportunity to delve deeper into specific areas of interest and gain expertise in their chosen field. The Machine Learning specialization, for example, offers additional courses such as Optimization Methods, Stochastic Modeling and Financial Mathematics, Deep Learning, and NLP. The Software Development specialization includes courses such as Databases Internals, Integrated Development and IT Operations, Software Design, Parallel Programming, and Distributed Algorithms, while the Programming Languages specialization includes Formal Languages and Parsers, Compilers, and Semantics of Programming Languages.

The close ties and support and participation of JetBrains in the development of the program will provide students with unique opportunities for project work, internships, and access to special courses, as well as the chance to receive scholarships covering tuition, boarding, insurance, and a monthly allowance.

1.3 Program-Specific Educational Aims

1.3.1 Qualification Aims

The main subject-specific qualification aim of the BSc Software, Data and Technology program at Constructor University is to enable students to take up qualified employment in modern industries involving software, data and technology or to enter graduate programs related to these fields. Graduates of the program will have the following competencies:

- Software, Data and Technology competence

Graduates will be familiar with the theoretical foundations of software, data and technology and will be able to design and develop systems addressing a given application scenario. They will be able to analyze and structure complex problems and will be able to address them using programspecific methods. Graduates will be able to construct and maintain complex systems using a structured, analytic, and creative approach.

- Communication competence

Graduates will be able to communicate subject-specific topics convincingly in both spoken and written form to fellow data scientists, software developers, or customers.

- Teamwork and project management competence

Graduates will be able to work effectively in a team and will be able to organize workflows in complex development efforts. They will be familiar with tools that support the development, testing, and maintenance of large systems and will be able to take design decisions in a constructive way.

- Learning competence

Graduates will 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 fields of software, data and technology.

- Personal and professional competence

Graduates will be 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. Additionally, the program being developed with the support and participation of JetBrains will provide students with the opportunity for project work and internships in the company.

1.3.2 Intended Learning Outcomes

By the end of the BSc Software, Data and Technology program, students will be able to

1. work professionally in the field of software, data and technology and enter graduate programs related to these fields;
2. apply fundamental concepts of mathematics, statistics, and computer science while solving data-related problems;
3. analyze at multiple levels of abstraction and use appropriate mathematical and computational methods to model and analyze real-world problems;
4. develop, analyze and implement algorithms using modern software engineering methods and programming languages;
5. understand the characteristics of a range of computing platforms and their advantages and limitations;
6. choose from multiple programming paradigms, languages and algorithms to solve a given problem adequately;
7. apply the necessary mathematical methods, such as linear algebra, analysis, calculus, and discrete mathematics;
8. recognize the context in which data science and software systems operate, including interactions with people and the physical world;
9. describe the state of published knowledge in the field of software, data and technology and in a chosen specialization (Machine Learning, Software Development, Programming Languages);
10. analyze and model real-life scenarios in organizations and industries using contemporary techniques of data science and software development, also taking methods and insights of other disciplines into account;
11. appropriately communicate solutions of problems in software, data and technology in both spoken and written form to specialists and non-specialists;
12. draw scientifically founded conclusions that consider social, professional, scientific, and ethical aspects;

13. work effectively in a diverse team and take responsibility in a team;

14. take responsibility for their own learning, personal and professional development and role in society, reflecting on their practice and evaluating critical feedback;

15. adhere to and defend ethical, scientific, and professional standards.

1.4 Career Options and Support

The Software, Data and Technology program at Constructor University offers students a wide range of career opportunities in the rapidly growing fields of computer science. As two of the key disciplines of the 21st century, software, data and technology affect almost all modern industries, making the job market highly favorable for graduates with a degree in this field. The program equips students with the skills and knowledge necessary to excel in various roles such as data scientist, data analyst, software engineer, full-stack developer, information systems manager, database administrator, application developer, machine learning engineer, IT consultant, and system analyst.

In addition to the broad range of career options available to graduates, the program also boasts strong industry partnerships with companies such as JetBrains, Acronis, Alemira, Virtuozzo, Rolos, and others. These partnerships provide students with valuable opportunities for internships, networking, and career development.

The Career Service Center (CSC) helps students in their career development. It provides students with high-quality training and coaching in CV creation, cover letter formulation, interview preparation, effective presenting, business etiquette, and employer research as well as in many other aspects, thus helping students identify and follow up on rewarding careers after graduating from Constructor University. Furthermore, the Alumni Office helps students establish a long-lasting and global network which is useful when exploring job options in academia, industry, and elsewhere.

1.5 Admission Requirements

Admission to Constructor University is selective and based on a candidate's school and/or university achievements, recommendations, self-presentation, and performance on standardized tests. Students admitted to Constructor 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 Score: 90, IELTS: Level 6.5 or equivalent)

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

For more detailed information about the admission visit:
https://constructor.university/admission-aid/application-information-undergraduate

1.6 More information and contacts

For more information on the study program, please contact the Study Program Coordinator:

Prof. Dr. Alexander Omelchenko

Professor of Applied Mathematics, Data Science and Computing

Email: aomelchenko@constructor.university

or visit our program website: https://constructor.university/programs/undergraduate-education/data-science-software-development

For more information on Student Services please visit:

https://constructor.university/student-life/student-services

2 The Curricular Structure

2.1 General

The curricular structure provides multiple elements for enhancing employability, interdisciplinarity, and internationality. The unique CONSTRUCTOR Track, offered across all undergraduate study programs, provides comprehensive tailor-made modules designed to achieve and foster career competency. Additionally, a mandatory internship of at least two months after the second year of study and the possibility to study abroad for one semester give students the opportunity 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 Constructor University are based on a coherently modularized structure, which provides students with an extensive and flexible choice of study plans to meet the educational aims of their major as well as minor study interests and complete their studies within the regular period.

The framework policies and procedures regulating undergraduate study programs at Constructor University can be found on the website (https://constructor.university/student-life/student-services/university-policies).

2.2 The Constructor University 4C Model

Constructor University offers study programs that comply with the regulations of the European Higher Education Area. All study programs are structured according to the European Credit Transfer System (ECTS), which facilitates credit transfer between academic institutions. The three-year undergraduate programs involve six semesters of study with a total of 180 ECTS credit points (CP). The undergraduate curricular structure follows an innovative and student-centered modularization scheme - the 4C Model. It groups the disciplinary content of the study program in three overarching themes, CHOICE-CORE-CAREER according to the year of study, while the university-wide CONSTRUCTOR Track is dedicated to multidisciplinary content dedicated to methods as well as intellectual skills and is integrated across all three years of study. The default module size is 5 CP, with smaller 2.5 CP modules being possible as justified exceptions, e.g. if the learning goals are more suitable for 2.5 CP and the overall student workload is balanced.

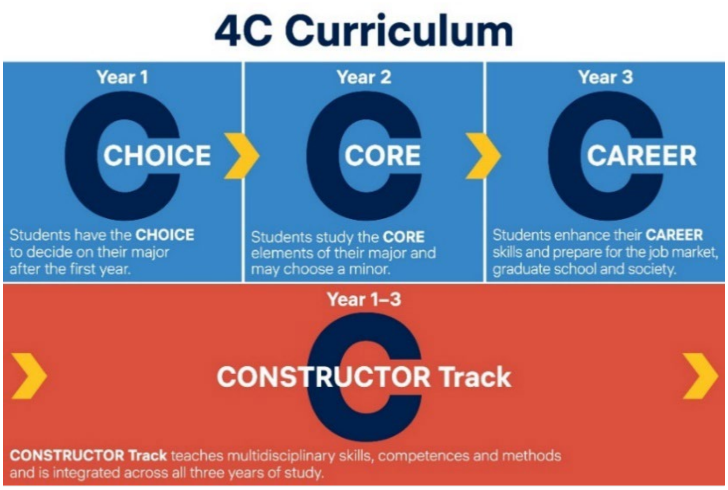


Figure 1: The Constructor University 4C-Model

2.2.1 Year 1 – CHOICE

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 select introductory modules for a total of 45 CP from the CHOICE area of a variety of study programs, of which 15-45 CP will be from their intended major. A unique feature of our curriculum structure allows students to select their major freely upon entering Constructor University. The team of Academic Advising Services 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 ontheir major study program.

To pursue a SDT major, the following CHOICE modules (30 CP) need to be taken as mandatory (m) modules during the first year of study:

- CHOICE Module: Programming in Python and C++ (m, 7.5 CP)
- CHOICE Module: Analysis (m, 7.5 CP)
- CHOICE Module: Core Algorithms and Data Structures (m, 7.5 CP)
- CHOICE Module: Development in JVM Languages (m, 7.5 CP)

The remaining two own CHOICE modules (15 CP) can be selected in the first year of study according to interest and/or with the aim of pursuing a minor or allowing a change of major up until the beginning of the second semester or after the first year, when the major becomes fixed .For students not pursuing a minor the following module is recommended in the second semester:

- CHOICE module: Linear Algebra (me, 7.5 CP)

In total, the first-year modules lay the foundation for the second year of education within the SDT major.

Students can still change to another major at the beginning of their second year of studies, provided they have taken the corresponding mandatory CHOICE modules in their first year of studies. All students must participate in an entry advising session with their Academic Advisors to learn about their major change options and consult their Academic Advisor prior to changing their major.

Students that would like to retain a further option are strongly recommended to additionally register for the CHOICE modules of one of the following study programs in their first year:

- IRPH
  CHOICE Module: Introduction to International Relations Theory (m, 7.5 CP)
  CHOICE Module: Introduction to Modern European History (m, 7.5 CP)

To allow further major changes after the first semester the students are strongly recommended to register for the CHOICE modules of one of the following study programs:

- Physics and Data Science (PHDS)
  CHOICE Module: Classical Physics (m, 7.5 CP)
  CHOICE Module: Programming in Python and C++ (m, 7.5 CP)
  CHOICE Module: Modern Physics (m, 7.5 CP)
  CHOICE Module: Mathematical Modeling (m, 7.5 CP)

- Mathematics, Modeling and Data Analytics (MMDA)
  CHOICE Module: Analysis (m, 7.5 CP)
  CHOICE Module: Programming in Python and C++ (m, 7.5 CP)
  CHOICE Module: Linear Algebra (m, 7.5 CP)
  CHOICE Module: Mathematical Modelling (m, 7.5 CP)
  CHOICE Module: Core Algorithms and Data Structures (me, 7.5 CP) or
  CHOICE Module: Algorithms and Data Structures (me, 7.5 CP)

The module descriptions can be found in the respective Study Program Handbook.

2.2.2 Year 2 – CORE

In their second year, students take a total of 45 CP from a selection of in-depth, discipline-specific CORE modules. Building on the introductory CHOICE modules and applying the methods and skills acquired thus far (see 2.3.1), these modules aim to expand students' critical understanding of the key theories, principles, and methods in their major for the current state of knowledge and best practice.

To pursue SDT as a major, the following mandatory (m) CORE modules (30 CP) must be taken:

- CORE Module: Operating Systems (m, 7.5 CP)
- CORE Module: Software Engineering and Design (m, 7.5 CP)
- CORE Module: Computer Architecture (m, 5 CP)
- CORE Module: Advanced Algorithms and Data Structures (m, 5 CP)
- CORE Module: Machine Learning (m, 5 CP)

Furthermore, students who are not pusuing a minor should take the following mandatory elective (me) modules:

- CORE Module: Functional Programming (me, 5 CP) OR Scientific Data Analysis (me, 5 CP)
- CORE Module: Databases (me, 5 CP)
- CORE Module: Discrete Mathematics (me, 5 CP) OR Artificial Intelligence (me, 5 CP)

2.2.2.1 Minor Option

SDT students can take CORE modules (or more advanced Specialization modules) from a second discipline, which allows them to incorporate a minor study track into their undergraduate education, within the 180 CP required for a bachelor's degree. The educational aims of a minor are to broaden the students' knowledge and skills, support the critical reflection of statements in complex contexts, foster an interdisciplinary approach to problem-solving, and to develop an individual academic and professional profile in line with students' strengths and interests. This extra qualification will be highlighted in a student`s final transcript.

The Academic Advising Coordinator, Academic Advisor, and the Study Program Chair of the minor study program support students in the realization of their minor selection; the consultation with the Academic Advisor is mandatory when choosing a minor.

As a rule, this requires SDT students to:

- select CHOICE modules (15 CP) from the desired minor program in the first year and thus substitute the mandatory elective SDT CHOICE module “Linear Algebra”.
- substitute the CORE modules "Databases", "Functional Programming OR Scientific Data Analysis" and “Discrete Mathematics OR Artificial Intelligence” in the second year (15 CP total) with the default minor CORE modules of the minor study program.

The requirements for the specific minors are described in the handbook of the study program offering the minor and are marked in the respective Study and Examination Plans. For an overview of accessible minors, please check the Major/Minor Combination Matrix, which is published at the beginning of each academic year.

2.2.3 Year 3 – CAREER

During their third year, students prepare and make decisions about their career path after graduation. To explore available choices and to gain professional experience, students undertake a mandatory summer internship. The third year of studies allows SDT students to take Specialization modules within their discipline, but also focuses on the responsibility of students beyond their discipline (see CONSTRUCTOR Track).

The fifth semester also opens a mobility window for a diverse range of study abroad options. Finally, the sixth semester is dedicated to fostering the students’ research experience by involving them in an extended Bachelor thesis project.

2.2.3.1 Internship / Start-up and Career Skills Module

As a core element of Constructor 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 role in employment and society, and find 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 of their business plans.

For further information, please contact the Career Service Center (CSC) (https://constructor.university/student-life/career-services).

2.2.3.2 Specialization Modules

In the third year of their studies, students take 15 CP from major-specific or major-related, advanced 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 which students can make free selections within a track during their fifth and sixth semester. The three specialization tracks are: Data Science, Software Development and Programming languages. The default Specialization Module size is 5 CP, with smaller 2.5 CP modules being possible as justified exceptions.

To pursue SDT as a major, 15 CP from the following major-specific Specialization modules within one track need to be taken:

Data Science track:

- SDT Specialization: Optimization Methods (me, 5 CP)
- SDT Specialization: Stochastic Modeling and Financial Mathematics (me, 5 CP)
- MSc CSSE CORE: Deep Learning (me, 5 CP)
- SDT Specialization: Natural Language Processing (me, 5 CP)

Software Development track:

- SDT Specialization: Databases Internals (me, 5 CP)
- SDT Specialization: Integrated Development and IT Operations (me, 5 CP)
- SDT Specialization: Parallel Programming (me, 5 CP)
- CS Specialization: Distributed Algorithms (me, 5 CP)
- CS CORE: Computer Networks (me, 5 CP)

Programming Languages track:

- SDT Specialization: Formal Languages and Parsers (me, 5 CP)
- SDT Specialization: Compilers (me, 5 CP)
- SDT Specialization: Semantics of Programming Languages (me, 5 CP)

Specialization modules are designed to allow an SDT student to become more focused on a particular subject of their choice within the SDT program or an affiliated program. The intention is to simultaneously support their personal development and career choices.

2.2.3.3 Study Abroad

Students have the opportunity to study abroad for a semester to extend their knowledge and abilities, broaden their horizons and reflect on their values and behavior in a different context as well as on their role in a global society. For a semester abroad (usually the 5th semester), modules related to the major with a workload equivalent to 22.5 CP must be completed. Modules recognized as study abroad CP need to be pre-approved according to Constructor University study abroad procedures. Several exchange programs allow students to directly enroll at prestigious partner institutions worldwide. Constructor University's participation in Erasmus+, the European Union's exchange program, provides an exchange semester at a number of European universities that include Erasmus study abroad funding.

For further information, please contact the International Office (https://constructor.university/student-life/study-abroad/international-office).

SDT students that wish to pursue a study abroad in their fifth semester are required to select their modules at the study abroad partners such that they can be used to substitute between 10-15 CP of major-specific Specialization modules and between 5-15 CP of modules equivalent to the non-disciplinary New Skills modules (see CONSTRUCTOR Track). In their sixth semester, according to the study plan, returning study-abroad students complete the Bachelor Thesis/Seminar module (see next section), they take any missing Specialization modules to reach the required 15 CP in this area, and they take any missing New Skills modules to reach 15 CP in this area.

2.2.3.4 Bachelor Thesis/Seminar Module

This module is a mandatory graduation requirement for all undergraduate students. It consists of two module components in the major study program guided by a Constructor University faculty member: the Bachelor Thesis (12 CP) and a Seminar (3 CP). 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 Constructor University Faculty Supervisor, the Bachelor Thesis can also have an interdisciplinary nature. In the seminar, students present and discuss their theses in a course environment and reflect on their theoretical or experimental approach and conduct. They learn to present their chosen research topics concisely and comprehensively in front of an audience and to explain their methods, solutions, and results to both specialists and non-specialists.

2.3 The CONSTRUCTOR Track

The CONSTRUCTOR Track is another important feature of Constructor University's educational model. The CONSTRUCTOR Track runs parallel to the disciplinary CHOICE, CORE, and CAREER modules across all study years and is an integral part of almost all undergraduate study programs. It reflects a university-wide commitment to help transform late-stage adolescents into confident, competent and responsible young adults by providing an intellectual tool kit to become life-long learners and by giving them the capacity to employ a range of methodologies to approach potential solutions to problems across disciplines. The CONSTRUCTOR track contains Methods, New Skills and German Language/Humanities modules.

2.3.1 Methods Modules

Methods such as mathematics, statistics, programming, data handling, presentation skills, academic writing, and scientific and experimental skills are offered to all students as part of the Methods area in their curriculum. The modules that are specifically assigned to each study program equip students with transferable academic skills. They convey and practice specific methods that are indispensable for each students' chosen study program. Students are required to take 20 CP in the Methods area. The size of all Methods modules is 5 CP.

To pursue Software, Data and Technologyas a major, the following Methods modules (20 CP) need to be taken as mandatory modules:

- Methods Module: Matrix Algebra & Advanced Calculus I (m, 5 CP)
- Methods Module: Matrix Algebra & Advanced Calculus II (m, 5 CP)
- Methods Module: Probability and Random Processes (m, 5 CP)
- Methods Module: Statistics and Data Analytics (m, 5 CP)

2.3.2 New Skills Modules

This part of the curriculum constitutes the intellectual and conceptual tool kit, and is designed to cultivate and nurture the capacity for a particular set of intellectual dispositions – curiosity, imagination, critical thought, transferability – as well as a range of individual and societal capacities – self-reflection, argumentation and communication – and to introduce students to the normative aspects of inquiry and research – including the norms governing sourcing, sharing, withholding materials and research results as well as others governing the responsibilities of expertise as well as the professional point of view.

All students are required to take the following modules in their second year:

- New Skills Module: Logic (m, 2.5 CP)
- New Skills Module: Causation and Correlation (m, 2.5 CP)

These modules will be offered with two different perspectives from which the students can choose. The module perspectives are independent modules which examine the topic from different points of view. Please see the module description for more details.

In the third year, students take three 5 CP modules that build upon previous modules in the track and are partially constituted by modules that are more closely linked to each student's disciplinary field of study. The following module is mandatory for all students:

- New Skills Module: Argumentation, Data Visualization and Communication (m, 5 CP)

This module will also be offered with two different perspectives of which the students can choose.

In their fifth semester, students may choose between:

- New Skills Module: Linear Model/Matrices (me, 5 CP) and
- New Skills Module: Complex Problem Solving (me, 5 CP).

The sixth semester also contains the choice between two modules, namely:

- New Skills Module: Agency, Leadership and Accountability (me, 5 CP) and
- New Skills Module: Community Impact Project (me, 5 CP).

Students who study abroad during the fifth semester and are not substituting the mandatory Argumentation, Data Visualization and Communication module, are required to take this module during their sixth semester. Students who remain on campus are free to take the Argumentation, Data Visualization and Communication module in person in either the fifth or sixth semester as they prefer.

2.3.3 German Language and Humanities Modules

German language abilities foster students' intercultural awareness and enhance their employability in their host country. They are also beneficial for securing mandatory internships (between the 2nd and 3rd year) in German companies and academic institutions. Constructor University supports its students in acquiring basic as well as advanced German skills in the first year of the Constructor Track. Non-native speakers of German are encouraged to take 2 German modules (2.5 CP each), but are not obliged to do so. Native speakers and other students not taking advantage of this offering take alternative modules in Humanities in each of the first two semesters:

- Humanities Module: Introduction to Philosophical Ethics (me, 2.5 CP)
- Humanities Module: Introduction to the Philosophy of Science (me, 2.5 CP)
- Humanities Module: Introduction to Visual Culture (me, 2.5 CP)

3 Software Development as a minor

3.1 Qualification Aims

Students obtaining a Minor in Software Development will gain a foundational understanding of key principles and practices in computer science and data science. They will learn programming languages such as Python and C++, core algorithms and data structures, computer architecture, advanced algorithms, and machine learning. Upon completion of the minor, students will have acquired sufficient knowledge to effectively collaborate with professionals in the fields of computer science and data science. They will be able to apply their knowledge and skills to drive digitalization processes and develop efficient solutions for problems in their domain. Students majoring in a technical discipline can obtain this minor to complement their skills and deepen their understanding of software and hardware components. The minor will prepare students to work in a variety of industries and sectors, where they can leverage their knowledge to analyze data, design software systems, and develop innovative solutions to complex problems.

3.2 Intended Learning Outcomes

With a minor in Software Development, students will be able to

- apply key principles and practices in computer science and data science to design, develop, and deploy software systems.
- analyze data, develop efficient algorithms, and apply machine learning techniques to solve complex problems.
- work collaboratively with professionals in the fields of computer science and data science, communicate effectively with stakeholders, and understand the technical aspects of a solution.
- gain a deep understanding of programming languages such as Python and C++, core algorithms and data structures, computer architecture, advanced algorithms, and machine learning.
- evaluate design choices and assess their impact on the efficiency and effectiveness of a solution.
- prepare to work in a variety of industries and sectors, where they can leverage their knowledge and skills to develop innovative solutions to complex problems.

3.3 Module Requirements

The following mandatory modules need to be taken in order to receive a minor:

- Programming in Python and C++ (m, 7,5 CP)
- Core Algorithms and Data Structures (m, 7,5 CP)
- Computer Architecture (m, 5 CP)
- Functional Programming (m, 5 CP)
- Databases (m, 5 CP)

3.4 Degree

After successful completion, the minor in Data Science and Software Design will be listed on the final transcript under PROGRAM OF STUDY and BA/BSc – [name of the major] as "(Minor: Software Development)."

4 Software, Data and Technology Undergraduate Program Regulations

4.1 Scope of these Regulations

The regulations in this handbook are valid for all students who entered the Software, Data and Technology undergraduate program at Constructor University in Fall 2023. In the case of a conflict between the regulations in this handbook and the general Policies for Bachelor studies, the latter applies (see https://constructor.university/student-life/student-services/university-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).

In general, Constructor University reserves therefore the right to change or modify the regulations of the program handbook according to relevant policies and processes also after its publication at any time and in its sole discretion.

4.2 Degree

Upon successful completion of the study program, students are awarded a Bachelor of Science degree in Software, Data and Technology.

4.3 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 6 of this handbook.

5 Schematic Study Plan for Software, Data and Technology

Figure 2 schematically shows 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 Plan in the following section.

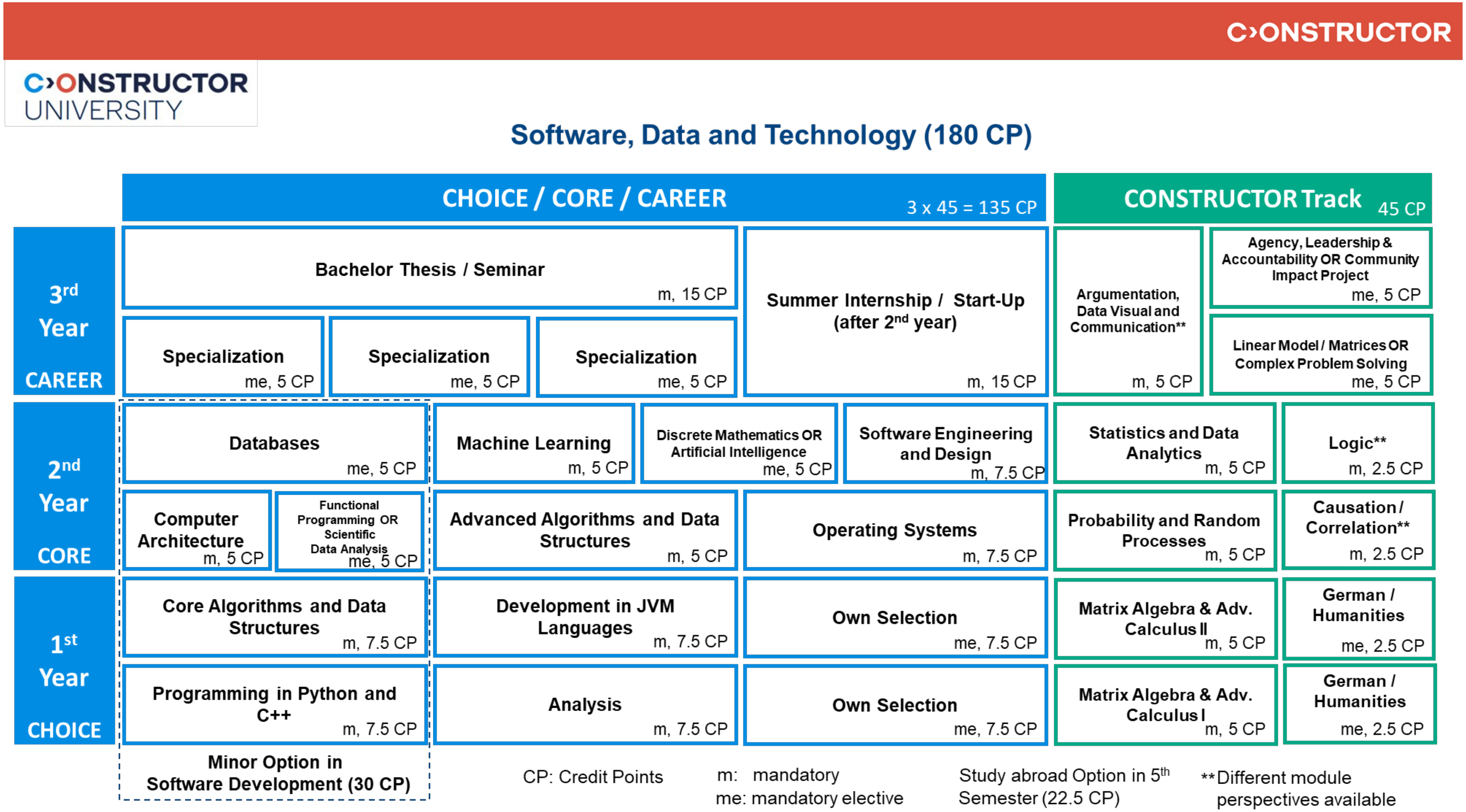


Figure 2: Schematic Study Plan for SDT

6 Study and Examination Plan

Software, Data, and Technology BSc

Matriculation Fall 2023

	Program-Specific Modules	Type	Assessment	Period	Status[1]	Sem.	ECTS
Year 1 - CHOICE							**45**
Take the mandatory CHOICE unit ***(s)*** *listed below, this is a requirement for the SDT program.*							
	Unit: Mathematics						**15**
CH-150	**Module: Analysis**				**m**	**1**	**7.5**
CH-150-A	Analysis	Lecture	Written examination	Examination period			7.5
CH-151	**Module: Linear Algebra**				**me**	**2**	**7.5**
CH-151-A	Linear Algebra	Lecture	Written examination	Examination period			7.5
	Unit: Programming						**15**
CH-250	**Module: Programming in Python and C++**				**m**	**1**	**7.5**
CH-250-A	Programming in Python and C++	Lecture	Written examination	Examination period			5
CH-250-B	Programming in Python and C++ Lab	Lab	Practical Assessment	During the semester			2.5
SDT-103	**Module:Development in JVM LanguagesProgramming**				**m**	**2**	**7.5**
SDT-103-A	Development in JVM Languages	Lecture	Written examination	Examination period			2.5
SDT-103-B	Development in JVM Languages	Tutorial	Practical Assessment	During the semester			5
	Unit: Data Science						**7.5**
CH-252	**Module: Core Algoriths & Data Structures**				**m**	**2**	**7.5**
CH-252-A	Core Algorithms and Data Structures	Lecture	Written examination	Examination period			5
CH-252-B	Core Algorithms and Data Structures Lab	Lab	Practical Assessment	During the semester			2.5
	Unit: CHOICE				**m**	**1**	**7.5**
Students take one further CHOICE unit from those offered for all other study programs.[2]							
Year 2 - CORE							**45**
Take all units listed below							
	Unit: Software Development						**25**
CO-562	**Module: Operating Systems**				**m**	**3**	**7.5**
CO-562-A	Operating Systems	Lecture	Written examination	Examination period	m		7.5
SDT-204	**Module: Software Engineering and Design**				**m**	**4**	**7.5**
SDT-204-A	Software Engineering and Design	Lecture	Written examination	Examination period	m	4	2.5
SDT-204-B	Software Engineering and Design Project	Project	Practical Assessment	During the semester	m	4	5
SDT-205	**Module: Databases**				**me**	**4**	**5**
SDT-205-A	Databases	Lecture	Written examination	Examination period	me	4	2.5
SDT-205-B	Databases Project	Project	Practical Assessment	During the semester	me	4	2.5
Take one of the two modules listed below							
SDT-202	**Module: Functional Programming**				**me**	**3**	**5**
SDT-202-A	Functional Programming	Lecture	Written examination	Examination period	me	3	2.5
SDT-202-B	Functional Programming Tutorial	Tutorial	Practical Assessment	During the semester	me	3	2.5
CO-450	**Module: Scientific Data Analysis**				**me**	**3**	**5**
CO-450-A	Scientific Data Analysis	Lecture	Portfolio assessment	During the semester	me	3	5
	Unit: Data Science				**m**		**15**
SDT-201	**Module: Advanced Algorithms and Data Structures**				**m**	**3**	**5**
SDT-201- A	Advanced Algoriths and Data Structures	Lecture	Written examination	Examination period	m	3	2.5
SDT-201-B	Advanced Algoriths and Data Structures Tutorial	Tutorial	Practical Assessment	During the semester	m	3	2.5
CO-541	**Module: Machine Learning**				**m**	**4**	**5**
CO-541-A	Machine Learning	Lecture	Written examination	Examination period	m	4	5
Take one of the two modules listed below							
CO-501	**Module: Discrete Mathematics**				**me**	**4**	**5**
CO-501-A	Discrete Mathematics	Lecture	Written examination	Examination period	me	4	5
CO-547	**Module: Artificial Intelligence**				**me**	**4**	**5**
CO-547-A	Artificial Intelligence	Lecture	Written examination	Examination period	me	4	5
	Unit: Further CORE modules						**5**
SDT-203	**Module: Computer Architecture**				**m**	**3**	**5**
SDT-203-A	Computer Architecture	Lecture	Written examination	Examination period	m	3	5

	Construtor Track Modules (General Education)	Type	Assessment	Period	Status[1]	Sem.	ECTS
							15
	Unit: Skills / Methods						**10**
CTMS-MAT-22	**Module: Matrix Algebra & Advanced Calculus I**				**m**	**1**	**5**
CTMS-22	Matrix Algebra & Advanced Calculus I	Lecture		Examination period			
CTMS-MAT-23	**Module: Matrix Algebra & Advanced Calculus II**				**m**	**2**	**5**
CTMS-23	Matrix Algebra & Advanced Calculus II	Lecture		Examination period			
	Unit: German Language and Humanities (choose one module for each sememster)				**m**		**5**
German is default language and open to Non-German speakers (on campus and online).[3]							
CTLA-	**Module: Language 1**				**me**	**1**	**2.5**
CTLA-	Language 1	Seminar	Various	Various	me		
CTLA-	**Module: Language 2**				**me**	**2**	**2.5**
CTLA-	Language 2	Seminar	Various	Various	me		
CTHU-HUM-001	**Humanities Module: Introduction into Philosophical Ethics**				**me**	**1**	**2.5**
CTHU-001	Introduction into Philosophical Ethics	Online Lecture	Written examination	Examination period	me		
CTHU-HUM-002	**Humanities Module: Introduction to the Philosophy of Science**				**me**	**2**	**2.5**
CTHU-002	Introduction to the Philosophy of Science	Online Lecture	Written examination	Examination period	me		
CTHU-HUM-003	**Introduction to Visual Culture**				**me**	**2**	**2.5**
CTHU-003	Introduction to Visual Culture	Online Lecture	Written examination	Examination period	me		
							15
	Unit: Methods					**3+4**	**10**
CTMS-MAT-12	**Module: Probability and Random Processes**				**me**	**3**	**5**
CTMS-12	Probability and Random Processes	Lecture	Written exam	Examination period			5
CTMS-MET-21	**Module: Statistics and Data Analytics**				**me**	**4**	**5**
CTMS-21	Statistics and Data Analytics	Lecture	Written exam	Examination period			5
	Unit: New Skills						**5**
CTNS-NSK-	**Module: Logic**				**m**	**3**	**2.5**
CTNS-01	Logic (perspective I)	Online Lecture	Written examination	Examination period	me		2.5
CTNS-02	Logic (perspective II)	Online Lecture	Written examination	Examination period	me		2.5
CTNS-NSK	**Module: Correlation and Causation**				**m**	**4**	**2.5**
CTNS-03	Correlation and Causation (perspective I)	Online Lecture	Written examination	Examination period	me		2.5
CTNS-04	Correlation and Causation (perspective II)	Online Lecture	Written examination	Examination period	me		2.5

Year 3 - CAREER							45
CA-INT-900	**Module: Summer Internship / Startup and Career Skills**				**m**	**4/5**	**15**
CA-INT-900-0	Internship / Startup and Career Skills	Internship	Report or Business plan				15
SDT-400	**Module: Seminar / Thesis SDT**				**m**	**6**	**15**
SDT-400-S	Seminar SDT	Seminar	Presentation	During the semester			3
SDT-400-T	Thesis SDT	Thesis	Thesis	During the semester			12
	Unit: Specialization (take a total 15 ECTS of specialization modules)						**15**
	Subunit: Machine Learning						
MCSSE-AI-01	**Module: Deep Learning**				**me**	**5**	**5**
MCSSE-AI-01	Deep Learning	Lecture	Written examination	Examination period	me	5	5
CA-S-MMDA-803	**Module: Stochastic Modeling and Financial Mathematics**				**me**	**6**	**5**
CA-MMDA-803	Stochastic Modeling and Financial Mathematics	Lecture	Portfolio	During the semester	me	6	5
SDT-301	**Module: Optimization Methods**				**me**	**5**	**5**
SDT-301-A	Optimization Methods	Lecture	Written examination	Examination period	me	5	2.5
SDT-301-B	Optimization Methods Tutorial	Tutorial	Practical Assessment	During the semester	me	5	2.5
SDT-305	**Module: Natural Language Processing**				**me**	**6**	**5**
SDT-305-A	Natural Language Processing	Lecture	Written examination	Examination period	me	6	5
	Subunit: Software Development						
SDT-302	**Module: Databases Internals**				**me**	**5**	**5**
SDT-302-A	Databases Internals	Lecture	Written examination	Examination period	me	5	5
SDT-306	**Module: Integrated Development and IT Operations**				**me**	**6**	**5**
SDT-306-A	Integrated Development and IT Operations	Lecture	Written examination	Examination period	me	6	5
SDT-303	**Module: Parallel Programming**				**me**	**5**	**5**
SDT-303-A	Parallel Programming	Lecture	Written examination	Examination period	me	5	5
CA-S-CS-803	**Module: Distributed Algorithms**				**me**	**6**	**5**
CA-CS-803	Distributed Algorithms	Lecture	Written examination	Examination period	me	6	5
CO-564	**Module: Computer Networks**				**me**	**5**	**5**
CO-564-A	Computer Networks	Lecture	Written examination	Examination period	me	5	5
	Subunit Programming Languages						
SDT-304	**Module: Formal Languages and Parsers**				**me**	**5**	**5**
SDT-304-A	Formal Languages and Parsers	Lecture	Written examination	Examination period	me	5	2.5
SDT-304-B	Formal Languages and Parsers Tutorial	Tutorial	Practical Assessment	During the semester	me	5	2.5
SDT-307	**Module: Compilers**				**me**	**6**	**5**
SDT-307-A	Compilers	Lecture	Written examination	Examination period	me	6	2.5
SDT-307-B	Compilers Project	Project	Practical Assessment	During the semester	me	6	2.5
SDT-308	**Module: Semantics of Programming Languages**				**me**	**6**	**5**
SDT-308-A	Semantics of Programming Languages	Lecture	Written examination	Examination period	me	6	2.5
SDT-308-B	Semantics of Programming Languages Tutorial	Tutorial	Practical Assessment	During the semester	me	6	2.5

							15
	Unit: New Skills						**15**
Choose one of the two modules							
CTNS-NSK-05	**Module: Linear Model / Matrices**				**me**	**5**	**5**
CTNS-05	Linear Model/ Matrices	Online Lecture	Written examination	Examination period			5
CTNS-NSK-06	**Module: Complex Problem Solving**				**me**	**5**	**5**
CTNS-06	Complex Problem Solving	Online Lecture	Written examination	Examination period	me	5	5
Take this module either in semester 5 or 6							5
CTNS-NSK	**Module: Argumentation, Data Visualization and Communication**				**m**	**5/6**	**5**
CTNS-07	Argumentation, Data Visualization and Communication (perspective I)	Online Lecture	Written examination	Examination period	me	**5**	5
CTNS-08	Argumentation, Data Visualization and Communication (perspective II)	Online Lecture	Written examination	Examination period	me	**6**	5
Choose one of the two modules							
CTNS-NSK	**Module: Agency, Accountability & Leadership**				**me**	**6**	**5**
CTNS-09	Agency, Accountability & Leadership	Online Lecture	Written examination	Examination period			5
CTNC-CIP-10	**Module: Community Impact Project**				**me**	**5/6**	**5**
JTCI-10	Community Impact Project	Project	Project	Examination period			5

Total ECTS	180

[1] Status (m = mandatory, me = mandatory elective)
[2] For a full listing of all CHOICE / CORE / CAREER / Constructor Track modules please consult the **CampusNet** online catalogue and /or the study program handbooks.
[3] German native speakers will have alternatives to the language courses (in the field of Humanities). Humanities I and II are optional to all students, except for German native speakers.

Specialization (Machine Learning*) (me, 5 CP)

- Optimization Methods **JB**
- Data Visualization and Image Processing **CUB**
- Stochastic Modeling and Financial Mathematics **CUB**
- Deep Learning **CUB**
- Natural Language Processing **JB**

Specialization (Software Development*) (me, 5 CP)

- Databases Internals **JB**
- Introduction to DevOps **JB**
- Software Design **JB**
- Parallel and Distributed Programming **JB**
- Distributed Algorithms **CUB**

Specialization (Programming Languages*) (me, 5 CP)

- Formal Languages **JB**
- Compilers **JB**
- Semantics of Programming Languages **JB**
- Parallel and Distributed Programming **JB**
- Distributed Algorithms CUB

Figure 3: Study and Examination Plan

7 Software, Data and Technology Modules

7.1 Analysis

<table>
<tr><td colspan="3">Module Name
Analysis</td><td>Module Code
CH-150</td><td>Level (type)
Year 1
(CHOICE)</td><td>CP
7.5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CH-150-A</td><td colspan="3">Analysis</td><td>Lecture</td><td>7.5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Sören Petrat</td><td colspan="3">Program Affiliation

• Mathematics, Modeling and Data Analytics (MMDA)</td><td colspan="2">Mandatory Status

Mandatory for MMDA, SDT and Minor in Mathematics</td></tr>
<tr><td colspan="3" rowspan="2">Entry Requirements

Pre-requisites: ☒ None
Co-requisites: ☒ None
Knowledge, Abilities, or Skills:
• Good command of high-school mathematics, in particular pre-calculus topics
• Good command of high-school calculus helps, but is not a prerequisite</td><td>Frequency

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

• Lectures (35 hours)
• Tutorials (17.5 hours)
• Private study (135 hours)</td></tr>
<tr><td>Duration

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

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

• It is recommended to co-enroll in the Methods module “Matrix Algebra & Advanced Calculus I”
• Revise your high school mathematics
• Read general interest expositions about mathematics and mathematicians
• Work on mathematics problems over the summer

For a detailed set of preparation instruction, references, and links, see http://math.Constructor-university.de/undergraduate/prepare/index.php</td></tr>
<tr><td colspan="6">Content and Educational Aims

This module introduces fundamental concepts and techniques in a concise and rigorous way. The class conveys the pleasure of doing mathematics, and motivates mathematics concepts from problems and concrete examples, but also shows the power of abstraction and of formal reasoning.

The following topics will be covered:
• Proof by induction, and elementary combinatorics
• Groups, equivalence relations, and quotients
• Natural numbers, integers, rationals, and real numbers
• Sequences and series, and convergence
• Functions of a single real variable, continuity, and the intermediate value theorem,
• Metric spaces, and the continuous functions as a metric space
• Differentiation, mean value theorem, and the inverse mapping theorem in one variable
• Riemann integral</td></tr>
</table>

- Fundamental theorem of Calculus, and the integration by parts with applications
- Integral mean value theorem
- Change of variables
- Taylor series with integral and Lagrange remainders
- Elementary point-set topology (neighborhoods, open and closed sets, compactness, and Heine-Borel)

Intended Learning Outcomes

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

1. cleanly formulate mathematical concepts and results discussed in class;
2. outline proofs which have been given in the lectures;
3. independently prove results which are direct consequences of those proved in the lectures;
4. understand and use fundamental mathematical terminology to communicate mathematics at a university level.

Indicative Literature

W. Rudin (1976). Principles of Mathematical Analysis, third edition. New York: McGraw-Hill.
T. Tao (2016). Analysis, third edition. New Delhi: Hindustan Book Agency.

Usability and Relationship to other Modules

- This module is part of the core education in Mathematics, Modeling and Data Analytics.
- It is also valuable for students in Physics, Computer Science, RIS, and ECE, either as part of a minor in Mathematics, or as an elective module.
- The curriculum is integrated with the curriculum of the module "Matrix Algebra and Advanced Calculus" in the following way: "Matrix Algebra and Advanced Calculus" emphasizes the operational aspects, computational skills, and intuitive understanding, while Analysis builds rigorous foundations of the field, emphasizing proof, abstraction, and mathematical rigor.

Examination Type: Module Examination

Assessment Type: Written examination Duration: 120 min

Weight: 100%

Scope: All intended learning outcomes of this module.

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

7.2 Programming in Python and C++

<table>
<tr><td colspan="3">Module Name
Programming in Python and C++</td><td>Module Code
SDT-101</td><td>Level (type)
Year 1
(CHOICE)</td><td>CP
7.5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>SDT-101-A</td><td colspan="3">Programming in Python and C++</td><td>Lectures</td><td>5</td></tr>
<tr><td>SDT-101-B</td><td colspan="3">Programming in Python and C++ - Lab</td><td>Lab</td><td>2.5</td></tr>
<tr><td>Module Coordinator
Prof. Dr. Aleksander Omelchenko</td><td colspan="3">Program Affiliation
• Software, Data and Technology (SDT)</td><td colspan="2">Mandatory Status
Mandatory for, MMDA, PHDS, SDT and Minor in Software Development
Mandatory elective for ECE</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
• Lectures (35 hours)
• Tutorials (17.5 hours)
• Independent study (115 hours)
• Exam preparation (20 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
Set up a suitable programming environment.</td></tr>
<tr><td colspan="6">Content and Educational Aims
This module provides a solid foundation in imperative programming concepts and techniques, with a focus on Python and C++ programming languages. This module enables students to write programs in Python that solve problems and perform various operations using functions, data structures, and control structures, provides a basic introduction to the C++ programming language and its standard library, with a focus on data structures and algorithms, develops students' problem-solving and algorithmic thinking skills through hands-on programming exercises and projects, fosters students' ability to design, write, and test programs that are robust, maintainable, and scalable, encourages students to pursue further studies and practice in the field of programming and data science.

Content:
• Introduction to Imperative Programming: Overview of basic concepts of imperative programming languages, including variables, assignments, loops, function calls, data structures, and more.
• Python Programming: Writing interactive programs in Python, working with user input, and testing and debugging code.
• Object-Oriented Programming in Python: Overview of basic object-oriented programming concepts, such as objects, classes, information hiding, inheritance, and function and operator overloading.
• File Input/Output in Python: Retrieving and processing data from/to files and generating data using Python.
• Scientific Computing with Python: Using NumPy arrays for vectorized code and SciPy for special</td></tr>
</table>

functions and black-boxed algorithms (root solvers, quadrature, ODE solvers, and fast Fourier transform).

- Visualization in Python: Visualizing data using Matplotlib.
- Introduction to C++ Programming: Writing basic programs in C++ using standard library functions.
- Pointers in C++: Using pointers to create dynamically allocated data structures, such as linked lists, and understanding the relationship between pointers and arrays.
- Standard Library Data Types in C++: Overview of C++ standard library data types, including vector, string, list, map, set, and sort.
- Risks and Limitations of C/C++: Understanding the risks of C/C++ programming, including implicit type conversions, lack of bounds checking, and manual memory ownership management.

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. explain basic concepts of imperative programming languages such as variables, assignments, loops, function calls, data structures, etc.;
2. work with user input from the keyboard, write interactive Python programs;
3. write, test, and debug programs;
4. illustrate basic object-oriented programming concepts such as objects, classes, information hiding and inheritance;
5. give original examples of function and operator overloading;
6. retrieve data and process and generate data from/to files;
7. write vectorized code using NumPy arrays
8. use SciPy for special functions and black-boxed algorithms (root solvers, quadrature, ODE solvers, and fast Fourier transform)
9. visualize data in appropriate ways using Matplotlib
10. write basic programs in the programming languages C/C++ using standard library functions
11. demonstrate how to use pointers to create dynamically allocated data structures such as linked lists;
12. explain the relationship between pointers and arrays;
13. use C++ standard library data types (vector, string, list, map, set, sort);
14. describe C/C++ risks (implicit type conversions, lack of bounds checking, manual memory ownership management)

Indicative Literature

Mark Lutz: "Learning Python", 5th edition, O'Reilly Media, 2013.

Lillian Pierson: "Data Science from Scratch: First Principles with Python", 2nd edition, O'Reilly Media, 2019.

Mark Summerfield: "Programming in Python 3: A Complete Introduction to the Python Language", 2nd edition, Addison-Wesley Professional, 2009.

David J. Pine: “Introduction to Python for Science and Engineering”, CRC Press, 2019.

John V. Guttag: "Introduction to Computation and Programming Using Python", 2nd edition, MIT Press, 2013.

Bjarne Stroustrup: "Programming -- Principles and Practice Using C++", Second edition, Addison-Wesley Professional, 2014.

Stanley Lippman: "C++ Primer (5th Edition)", 2012

Scott Meyers: "Effective Modern C++", O'Reilly Media, 2014.

H. M. Deitel and P. J. Deitel: "C++ How to Program", 10th edition, Pearson, 2015.

John Zelle: "Python Programming: An Introduction to Computer Science", 3rd edition, Franklin, Beedle & Associates, Inc., 2016.

Usability and Relationship to other Modules

Examination Type: Module Component Examination	
Component 1: Lecture	
Assessment: Written examination	Duration: 120 min
	Weight: 67%
Scope: All theoretical intended learning outcomes of the module	
Component 2: Lab	
Assessment: Practical assessment (Programming assignments)	Weight: 33%
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%	

7.3 Linear Algebra

Module Name		Module Code	Level (type)	CP
Linear Algebra		CH-151	Year 1 (CHOICE)	7.5
Module Components				
Number	Name		Type	CP
CH-151	Linear Algebra		Lecture	7.5
Module Coordinator Dr. Ivan Ovsyannikov	**Program Affiliation** • Mathematics, Modeling and Data Analytics (MMDA)		**Mandatory Status** Mandatory for MMDA and minor in Mathematics Mandatory elective for SDT	
Entry Requirements Pre-requisites ☒ None	Co-requisites ☒None	Knowledge, Abilities, or Skills • Basic matrix algebra at the level achieved in “Matrix Algebra and Advanced Calculus”	**Frequency** Annually (Spring) **Duration** 1 semester	**Forms of Learning and Teaching** • Lectures (35 hours) • Tutorials (17.5 hours) • Private study (135 hours) **Workload** 187.5 hours

Recommendations for Preparation

- Revise your matrix algebra.
- Unless prepared otherwise, take the Methods module “Matrix Algebra and Advanced Calculus I” in the first semester.

Content and Educational Aims

This module continues the introduction to Linear Algebra from the methods module “Matrix Algebra and Advanced Calculus I”. The fundamental concepts and techniques of Linear Algebra are introduced in a rigorous and more abstract way. The first half of this module covers vector spaces and linear maps, while the second half covers inner products and geometry.

The following topics will be covered:

- Vector spaces
- Linear Operators
- Dual spaces
- Isomorphisms
- Connection to matrices
- Sums and direct sums
- Fundamental spaces of a linear operator
- Diagonalization of linear operators (on finite dimensional spaces)
- Cayley-Hamilton theorem
- Jordan decomposition
- Jordan normal form and its applications to linear differential equations
- Decomplexification and complexification
- Bilinear Forms and their classification
- Quadratic forms and orthogonalization
- Euclidean and unitary spaces
- Orthogonal and unitary operators

- Self-adjoint operators

Intended Learning Outcomes

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

1. describe the concept of a vector space and linear operator in an abstract way
2. explain the connection of abstract linear algebra in the context of matrix algebra
3. discuss the proofs of the major theorems from class
4. illustrate the use of bilinear forms and their role in geometry
5. distinguish bilinear forms in the context of Euclidean, unitary and symplectic spaces

Indicative Literature

P.K. Kostrikin, Yu. Manin (1997) Linear Algebra and Geometry. London: Gordon and Breach.
S. Axler (2005) Linear Algebra Done Right, third edition. Berlin: Springer.
G. Strang (2016). Introduction to Linear Algebra. Wellesley: Wellesley-Cambridge Press, fifth edition.
S. Lang (1986). Introduction to Linear Algebra, second edition. Berlin: Springer.

Usability and Relationship to other Modules

- This module is part of the core education in Mathematics
- This module is valuable for students in Computer Science, RIS, and ECE, either as part of a minor in Mathematics, or as an elective module
- The curriculum is integrated with the curriculum of the module “Matrix Algebra and Advanced Calculus I and II” in the following way: “Matrix Algebra and Advanced Calculus I and II” emphasizes the operational aspects, computational skills, and intuitive understanding, while Linear Algebra builds rigorous foundations of the field, emphasizing proof, abstraction, and mathematical rigor.

Examination Type: Module Examination

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

Scope: All intended learning outcomes of this module

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

7.4 Development in JVM Languages

<table>
<tr><td colspan="3">Module Name
Development in JVM Languages</td><td>Module Code

SDT-103</td><td>Level (type)

Year 1
(CHOICE)</td><td>CP

7.5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>SDT-103-A</td><td colspan="3">Development in JVM Languages</td><td>Lectures</td><td>2.5</td></tr>
<tr><td>SDT-103-B</td><td colspan="3">Development in JVM Languages</td><td>Tutorials</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Aleksander Omelchenko</td><td colspan="3">Program Affiliation
Software, Data and Technology (SDT)</td><td colspan="2">Mandatory Status

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

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

Annually
(Spring)</td><td colspan="2">Forms of Learning and Teaching
Lecture attendance (35 hours)
Tutorial attendance (35 hours)
Independent study (97.5 hours)
Exam preparation (20 hours)</td></tr>
<tr><td>Duration

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

187.5 hours</td></tr>
</table>

Recommendations for Preparation

Students should refresh their knowledge of the C++ and Python programming language and be able to solve simple programming problems in C++ and Python. Students are expected to have a working programming environment.

Content and Educational Aims

In this module students will learn about the Kotlin programming language, a modern, powerful and expressive language that is used for various purposes from android development, web development to data science. Students will learn how to apply Kotlin to solve practical problems in software development and will learn about data types, variables and control flow, functions, object-oriented programming, exception handling, collections and generics, lambdas, and higher-order functions. They will also learn about the unique features of Kotlin such as null safety, extension functions and type inference.

Educational Aims:

- To provide students with a solid foundation in the Kotlin programming language
- To teach students how to apply Kotlin to solve practical problems in software development
- To enable students to write efficient, readable and maintainable code using Kotlin
- To familiarize students with the unique features of Kotlin such as null safety, extension functions, and type inference
- To prepare students for using Kotlin in Android Development.
- To give students a deeper understanding of the fundamental concepts of computer science, such as algorithms and data structures and how they can be applied to software development.

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. write, understand and debug Kotlin code effectively.
2. use the unique features of Kotlin to write readable, maintainable and expressive code.
3. use Kotlin to solve practical problems in software development.
4. design and implement object-oriented programs in Kotlin.
5. use Kotlin collections and Generics in their programs.
6. use Lambdas and Higher-Order functions in Kotlin.
7. use Kotlin for android development.
8. write efficient and optimized code using Kotlin.
9. use Kotlin for web development.
10. use Kotlin for data science.

Indicative Literature

- Venkat Subramaniam: Programming Kotlin, Pragmatic Bookshelf, 2017.
- Hadi Hariri: Kotlin in Action, Manning Publications, 2017.
- Dmitry Jemerov and Svetlana Isakova: Kotlin in Practice, JetBrains, 2016.
- Antonio Leiva: Kotlin for Android Developers, Leanpub, 2015.
- Marcin Moskala: Kotlin Programming, Packt Publishing, 2018.

Usability and Relationship to other Modules

- Familiarity with Kotlin programming language is essential for students who wish to specialize in android development, web development or data science. This module will provide a solid foundation in Kotlin programming, including its unique features such as null safety, extension functions, and type inference. Additionally, this module will introduce advanced concepts of programming that are needed in advanced programming-oriented modules in the 2nd and 3rd years of the SDT program.

Examination Type: Module Component Examination

Component 1: Lecture

Assessment: Written examination — Duration: 60 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%.

7.5 Core Algorithms and Data Structures

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

Core Algorithms and Data Structures</td><td>Module Code

SDT-102</td><td>Level (type)

Year 1
(CHOICE)</td><td>CP

7.5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>SDT-102-A</td><td colspan="3">Core Algorithms and Data Structures</td><td>Lecture</td><td>5</td></tr>
<tr><td>SDT-102-B</td><td colspan="3">Core Algorithms and Data Structures - Lab</td><td>Lab</td><td>2.5</td></tr>
<tr><td>Module Coordinator

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

• Software, Data and Technology (SDT)</td><td colspan="2">Mandatory Status

Mandatory for SDT and Minor in Software Development

Mandatory elective for ECE, MMDA, and PHDS</td></tr>
<tr><td rowspan="2">Entry Requirements

Pre-requisites

☒ Programming in Python and C++ or Programming in C/C++</td><td rowspan="2">Co-requisites

☒ none</td><td rowspan="2">Knowledge, Abilities, or Skills</td><td>Frequency

Annually

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

• Lecture (35 hours)
• Tutorial (17.5 hours)
• Independent study (115 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td>Duration

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

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

Students should refresh their knowledge of the C, C++ and Python programming language and be able to solve simple programming problems in C, C++ and Python. Students are expected to have a working programming environment.</td></tr>
<tr><td colspan="6">Content and Educational Aims

Algorithms and data structures are the foundation of computer science and are crucial for the design and implementation of efficient software programs. In this module, students will learn about fundamental algorithms for solving problems and about data structures for storing, accessing, and modifying data in an efficient manner. They will also learn techniques for analyzing the computational and memory complexities of algorithms and data structures. These concepts and techniques form the basis for almost all computer programs and are essential for success in the fields of software, data and technology.

Content:

• Introduction (asymptotic analysis of algorithms, analysis of recurrence relations, sums and integrals, time complexity, non-asymptotic optimizations, cache)
• Basic data structures (array, list, stack, queue, vector, hash tables, binary heap, heapsort, etc.)
• Sorting algorithms and heaps (quadratic sorting, stable sorting, mergesort, etc.)
• Graphs: depth-first search (DFS) and breadth-first search (BFS) algorithms.
• Graphs: matchings, colorings, flows, cuts.
• Graphs: shortest paths
• Introduction to Complexity Theory, Probabilistic Algorithms</td></tr>
</table>

- Numerical and Algebraic Algorithms

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. Analyze the time and space complexity of algorithms and optimize them using asymptotic analysis and non-asymptotic techniques such as cache optimization.
2. Implement and evaluate various data structures including arrays, lists, stacks, queues, vectors, hash tables, binary heaps, and heapsort.
3. Compare and contrast different sorting algorithms, including quadratic sorting, stable sorting, and mergesort, and understand the trade-offs involved in their use.
4. Implement depth-first search (DFS) and breadth-first search (BFS) algorithms and understand their applications in graph theory.
5. Analyze matchings, colorings, flows, and cuts in graphs, and understand the algorithms and mathematical foundations used to solve these problems.
6. Implement shortest path algorithms in graphs and understand their applications in network design and routing.
7. Understand the fundamental concepts of complexity theory and probabilistic algorithms, and apply them in solving computational problems.
8. Analyze and implement numerical and algebraic algorithms and understand their applications in a variety of fields.
9. Develop the ability to analyze, design, and implement algorithms for solving real-world problems and understand the trade-offs involved in their use.

Indicative Literature

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

Robert Sedgewick and Kevin Wayne: Algorithms, 4th edition, Addison-Wesley, 2011.

Steven Skiena: The Algorithm Design Manual, 2nd edition, Springer, 2008.

Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser: Data Structures and Algorithms in Python, John Wiley & Sons, 2013.

Jon Kleinberg and Éva Tardos: Algorithm Design, 1st edition, Pearson, 2005.

David E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

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

- This module will provide students with a solid foundation for understanding how to design and analyze algorithms for solving problems, as well as data structures for efficiently storing and manipulating data.

Examination Type: Module Component Examination

Component 1: Lecture

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

Scope: All theoretical intended learning outcomes of the module

Component 2: Lab

Assessment: Practical assessment (Programming assignments) — Weight: 33%

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%

7.6 Operating Systems

<table>
<tr><td colspan="3">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="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CO-562-A</td><td colspan="3">Operating Systems</td><td>Lecture</td><td>7.5</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 for CS and SDT</td></tr>
<tr><td colspan="3">Entry Requirements</td><td>Frequency

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

• Class attendance (52.5 hours)
• Independent study (115 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td>Pre-requisites

☒ Algorithms and Data Structures or Core Algorithms and Data Structures</td><td>Co-requisites

☒ None</td><td>Knowledge, Abilities, or Skills
Students are expected to understand data representation and program execution at the machine instruction level as covered in the module Introduction to Computer Science.</td><td>Duration

1 semester</td></tr>
<tr><td colspan="4"></td><td colspan="2">Workload

187.5 hours</td></tr>
<tr><td colspan="6">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="6">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="6">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;</td></tr>
</table>

7. solve basic inter-process communication problems using signals and pipes;
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

- 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.

Completion: To pass this module, the examinations must be passed with at least 45%.

7.7 Functional Programming

<table>
<tr><td colspan="2">Module Name
Functional Programming</td><td>Module Code

SDT-202</td><td>Level (type)
Year 2
(CORE)</td><td>CP
5</td></tr>
<tr><td colspan="5">Module Components</td></tr>
<tr><td>Number</td><td colspan="2">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>SDT-202-A</td><td colspan="2">Functional Programming</td><td>Lecture</td><td>2.5</td></tr>
<tr><td>SDT-202-B</td><td colspan="2">Functional Programming Tutorial</td><td>Tutorial</td><td>2.5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Aleksander Omelchenko</td><td colspan="2">Program Affiliation

• Software, Data and Technology (SDT)</td><td colspan="2">Mandatory Status

Mandatory for Minor in Software Development
Mandatory elective for SDT</td></tr>
<tr><td colspan="2" rowspan="2">Entry Requirements

Pre-requisites: ☒ Core Algorithms and Data Structures or 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
• Lecture attendance (17.5 hours)
• Tutorial attendance (17.5 hours)
• Independent study (70 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td>Duration

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

125 hours</td></tr>
</table>

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.

Content and Educational Aims

The goal of this discipline is to provide students with a solid foundation in functional programming principles and techniques, focusing on the theoretical knowledge and practical skills required to effectively work with functional languages. The module explores the core concepts, language structures, syntax, and semantic constructs of functional programming languages, emphasizing their applicability in modern software development

.

Content:

- Fundamentals of functional programming: lambda calculus and combinatory logic.
- Haskell programming language: syntax, semantics, standard library.
- Manage effects using applicative functors and monads.
- Type systems of functional languages.

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. Collaborate effectively within a team in the IT field, utilizing project management tools, communication skills, and software for team project activities to jointly develop projects.
2. Compare and contrast the advantages and disadvantages of the functional programming paradigm, and apply functional programming techniques to solve applied problems using languages such as Haskell.
3. Understand and utilize the basic type systems of functional languages and their extensions with polymorphic and recursive types to create efficient, well-structured code in a functional programming context.
4. Choose between lazy and eager evaluation strategies based on the specific requirements of a problem or application, and implement software solutions using a functional programming paradigm.
5. Explain the computational model underlying functional programming and implement algorithms in functional languages using key concepts such as immutable data structures, recursion, and pattern matching.
6. Employ generic annotations and type classes to describe interfaces and ensure static control, promoting code reusability and maintainability in functional programming projects.

Indicative Literature

- Miran Lipovača. Learn You a Haskell for Great Good.
- O'Sullivan, Bryan, John Goerzen, and Don Stewart. Real World Haskell. O'Reilly Media, Inc., 2008
- Hughes, John. "Why functional programming matters." The computer journal 32.2 (1989): 98-107.

Usability and Relationship to other Modules

- Familiarity with functional programming concepts and principles is increasingly important in fields such as data science, artificial intelligence, and software development. This module provides a solid foundation in functional programming techniques and languages, which are essential for advanced modules in computer science and data science. Additionally, this module introduces advanced concepts of functional programming that are needed in advanced programming-oriented modules in the 2nd and 3rd years of the SDT program.

Examination Type: Module Component Examination

Component 1: Lecture

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

Scope: All theoretical intended learning outcomes of the module

Component 2: Tutorial

Assessment: Practical assessment (Programming assignments) — Weight: 50%

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%.

7.8 Computer Architecture

<table>
<tr><td colspan="2">Module Name
Computer Architecture</td><td>Module Code
SDT-203</td><td>Level (type)
Year 2
(CORE)</td><td>CP
5</td></tr>
<tr><td colspan="5">Module Components</td></tr>
<tr><td>Number</td><td colspan="2">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>SDT-203-A</td><td colspan="2">Computer Architecture</td><td>Lecture</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Kirill Krinkin</td><td colspan="2">Program Affiliation

● Software, Data and Technology (SDT)</td><td colspan="2">Mandatory Status

Mandatory for SDT and Minor in Software Development</td></tr>
<tr><td colspan="2">Entry Requirements

Pre-requisites: ☒ Programming in Python and C++ or Programming in C/C++

Co-requisites: ☒ none

Knowledge, Abilities, or Skills: Number systems, computer arithmetic, Basic knowledge C/C++</td><td>Frequency

Annually
(Fall)</td><td colspan="2">Forms of Learning and Teaching
● Lecture attendance (35 hours)
● Independent study (70 hours)
● Exam preparation (20 hours)</td></tr>
<tr><td colspan="2"></td><td>Duration

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

125 hours</td></tr>
</table>

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. Repeat school knowledge of electricity: the concept of voltage, resistance, amperage. Ohm's law.

Content and Educational Aims

The discipline introduces students to the architecture of the computer at multiple levels. Starting with electrical signals and the physical structure of the computer's main components, the course material gradually dives into the mechanisms of the processor, memory, and communication between electronic components. The module provides an understanding of how the computer works from the moment you press a key, to changes in the memory state.

Content:

- Principles and concepts of computer architecture, instruction sets, assembly language, memory hierarchy, input/output systems.
- Number systems and representations: binary, octal, decimal, and hexadecimal systems; conversions between number systems; binary arithmetic; and two's complement representation
- Trade-offs and design choices involved in the implementation of computer systems and their impact on performance.
- Advanced topics such as pipelining, parallel processing, and memory management.

Intended Learning Outcomes

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

1. understand the basic principles and concepts of computer architecture, including instruction sets, assembly language, memory hierarchy, and input/output systems.
2. analyze the trade-offs and design choices involved in the implementation of computer systems, and their impact on performance.
3. understand advanced topics in computer architecture such as pipelining, parallel processing, and memory management.
4. apply the knowledge and skills learned in the module to design and implement computer systems that are efficient and optimized for specific applications.

5. analyze and compare different computer architectures and evaluate their suitability for different applications.
6. communicate effectively about computer architecture using technical terminology and diagrams.
7. develop an understanding of the ethical and societal implications of computer architecture.

Indicative Literature

- David A. Patterson, John L. Hennessy: Computer Organization and Design: The Hardware/Software Interface, 5th edition, Morgan Kaufmann Publishers, 2017.
- John P. Hayes: Computer Architecture and Organization, 3rd edition, McGraw-Hill, 1998.
- William Stallings: Computer Organization and Architecture: Designing for Performance, 9th edition, Pearson, 2016.
- Kai Hwang: Advanced Computer Architecture: Parallelism, Scalability, Programmability, 2nd edition, McGraw-Hill, 1998.
- Michael J. Flynn: Computer Architecture: Pipelined and Parallel Processor Design, Jones and Bartlett Publishers, 2009.

Usability and Relationship to other Modules

- An understanding of computer architecture is essential for students pursuing careers in computer science, software engineering, and related fields. This module provides a solid foundation in the basic principles and concepts of computer architecture, including instruction sets, assembly language, memory hierarchy, and input/output systems. It also covers advanced topics such as pipelining, parallel processing, and memory management, which are essential for understanding the performance and design of modern computer systems. This knowledge is crucial for students who wish to pursue advanced coursework in computer science and related fields, as well as for those who wish to pursue careers in fields such as software development, network engineering, and data science.

Examination Type: Module Examination

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

Scope: All intended learning outcomes of the module

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

7.9 Scientific Data Analysis

<table>
<tr><td colspan="2">Module Name
Scientific Data Analysis</td><td colspan="2">Module Code
CO-450</td><td>Level (type)
Year 2 (CORE)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CO-450-A</td><td colspan="3">Scientific Data Analysis</td><td>Lecture</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Veit Wagner</td><td colspan="3">Program Affiliation

• Physics and Data Science (PHDS)</td><td colspan="2">Mandatory Status

Mandatory for MMDA and PHDS
Mandatory elective for SDT</td></tr>
<tr><td colspan="3" rowspan="2">Entry Requirements

Pre-requisites: ☒ Core Algorithms and Data Structures or Algorithms and Data Structures
Co-requisites: ☒ none
Knowledge, Abilities, or Skills: Mathematics at the level of the Mathematical Modelling module; Basic programming skills in Python</td><td>Frequency

Annually (Fall)</td><td colspan="2">Forms of Learning and Teaching
• Lecture (35 hours)
• Homework exercises (55 hours)
• Private study (35 hours)</td></tr>
<tr><td>Duration

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

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

Review mathematics/linear algebra/statistics and programming at the level of the first-year modules.</td></tr>
<tr><td colspan="6">Content and Educational Aims

Interpretation of scientific data is at the core of knowledge creation in any science. Proper tools and analysis techniques are the foundation for new theory validation against experimental findings, parameter extraction from computational or experimental data, and to discover data relationships in given data sets. This holds for all fields of physics, for the natural sciences in general and for fields beyond. This module provides a calculus-based introduction to analytical techniques applied to scientific data sets. Topics include probability distributions, linear and non-linear least square estimation, Bayesian statistics, Fourier analysis, (time) sequence Analysisncluding power spectra and convolution, principal component analysis, data visualization techniques, as well as error and outlier analysis. Exemplary datasets from experimental and computational sources are used throughout the course. The module introduces their proper handling and data organization in databases. The module is part of the core physics and data science as well as the core mathematics, modeling and data analytics education. It builds on the foundation of the programming lab, the data handling in first year lab courses and first year mathematics foundations. Essential practical experience in applying the various analysis techniques and their visualization will be supported by homework exercises in close coordination with the lectures. The aim of the module is to enable students to properly handle, store, analyze and visualize larger multidimensional scientific datasets by various methods and from various fields, and to prepare students for the data handling in their BSc thesis research. At the same time, students' programming and mathematical repertoires as well as their problem-solving skills are developed. The module also serves as a foundation for specialization subject modules.</td></tr>
<tr><td colspan="6">Intended Learning Outcomes

Upon completion of this module, students will be able to
1. perform curve and model fitting
2. conduct advanced data Analysis including Fourier analysis and Bayesian statistics
3. understand error handling in multidimensional complex data analysis
4. store, import, handle and visualize large data sets</td></tr>
</table>

Indicative Literature
Graham Currell: Scientific Data Analysis, Oxford University Press, 2015. Edward L. Robinson: Data Analysis for Scientists and Engineers, Princeton University Press, 2016.
Usability and Relationship to other Modules
Examination Type: Module Examination Assessment Type: Portfolio (assignments, quizzes) Weight: 100% Scope: All intended learning outcomes of the module Completion: To pass this module, the examination has to be passed with at least 45%.

7.10 Advanced Algorithms and Data Structures

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

Advanced Algorithms and Data Structures</td><td>Module Code

SDT-201</td><td>Level (type)

Year 2
(CORE)</td><td>CP

5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>SDT-201-A</td><td colspan="3">Advanced Algorithms and Data Structures</td><td>Lecture</td><td>2.5</td></tr>
<tr><td>SDT-201-B</td><td colspan="3">Advanced Algorithms and Data Structures Tutorial</td><td>Tutorial</td><td>2.5</td></tr>
<tr><td>Module Coordinator

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

• Software, Data and Technology (SDT)</td><td colspan="2">Mandatory Status

Mandatory for SDT</td></tr>
<tr><td rowspan="2">Entry Requirements

Pre-requisites

☒ Core Algorithms and Data Structures or Algorithms and Data Structures</td><td rowspan="2">Co-requisites

☒ none</td><td rowspan="2">Knowledge, Abilities, or Skills</td><td>Frequency

Annually

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

• Lecture attendance (17.5 hours)
• Tutorial attendance (17.5 hours)
• Independent study (70 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td>Duration

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

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

Students should refresh their knowledge of the C++ and Python programming language and be able to solve simple programming problems in C++ and Python. Students are expected to have a working programming environment. Also they should refresh their knowledge of the basics of algorithms and data structures.</td></tr>
<tr><td colspan="6">Content and Educational Aims

This module builds on the concepts and techniques covered in "Core Algorithms and Data Structures" and provides students with a deeper understanding of these important topics. The module will focus on more advanced algorithms and data structures that are commonly used in practice, and will provide students with a solid foundation for more advanced coursework in the program and for professional development in the field of software, data and technology.

Content:

• Advanced data structures such as B-trees, AVL trees, and hash tables
• Advanced algorithms for sorting, searching, and graph manipulation
• Techniques for parallel and distributed algorithms
• Algorithms for network flow and linear programming
• Algorithms for approximation and randomization
• Advanced algorithms for specific areas such as computational geometry, cryptography, and machine learning
• Techniques for analyzing the performance of algorithms and data structures and making trade-offs between time and space complexity</td></tr>
</table>

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. design, implement and analyze advanced algorithms and data structures for various problems.
2. understand the trade-offs between time and space complexity and make appropriate decisions when choosing algorithms and data structures.
3. apply advanced algorithms and data structures to solve problems in different areas of computer science such as distributed systems, databases, and machine learning.
4. understand and use parallel and distributed algorithms.
5. understand the concepts of computational complexity theory and use them to analyze the performance of algorithms and data structures.
6. understand the properties and use of specific algorithms and data structures used in different areas of computer science.
7. apply mathematical concepts and formalize algorithms to solve practical problems.

Indicative Literature

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein: Introduction to Algorithms, 3rd edition, MIT Press, 2009.
- Robert Sedgewick and Kevin Wayne: Algorithms, 4th edition, Addison-Wesley, 2011.
- Steven Skiena: The Algorithm Design Manual, 2nd edition, Springer, 2008.
- Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser: Data Structures and Algorithms in Python, John Wiley & Sons, 2013.
- Jon Kleinberg and Éva Tardos: Algorithm Design, 1st edition, Pearson, 2005.
- David E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

Usability and Relationship to other Modules

- Familiarity with advanced algorithms and data structures is essential for almost all advanced modules in SDT. This module builds upon the concepts covered in "Core Algorithms and Data Structures" and introduces more advanced algorithms and data structures that are commonly used in practice. Additionally, the module covers techniques for designing, implementing and analyzing efficient algorithms and data structures, and provides students with hands-on experience implementing these concepts in a programming language. This module is essential for students planning to continue their studies in the 2nd and 3rd years of the SDT program, as well as for those planning to pursue a career in the field of computer science.

Examination Type: Module Component Examination

Component 1: Lecture

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

Scope: All theoretical intended learning outcomes of the module

Component 2: Tutorial

Assessment: Practical assessment (Programming assignments) — Weight: 50%

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%.

7.11 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</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CO-541-A</td><td colspan="3">Machine Learning</td><td>Lecture</td><td>5</td></tr>
<tr><td>Module Coordinator
Prof. Dr. Francesco Maurelli</td><td colspan="3">Program Affiliation
• Robotics and Intelligent Systems (RIS)</td><td colspan="2">Mandatory Status
Mandatory for MMDA, PHDS, RIS, SDT and minor in RIS
Mandatory elective for CS</td></tr>
<tr><td colspan="3">Entry Requirements</td><td rowspan="2">Frequency
Annually
(Spring)</td><td colspan="2" rowspan="2">Forms of Learning and Teaching
• Class attendance (35 hours)
• Private study (70 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td rowspan="2">Pre-requisites
☒ none</td><td rowspan="2">Co-requisites
☒None</td><td rowspan="2">Knowledge, Abilities, or Skills
• Knowledge and command of probability theory and methods, as in the module “Probability and Random Process (JTMS-12)</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
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>
<tr><td colspan="6">Intended Learning Outcomes
By the end of this module, students will 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.</td></tr>
</table>

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

7.12 Discrete Mathematics

Module Name	**Module Code**	**Level (type)**	**CP**
Discrete Mathematics	CO-501	Year 2 (CORE)	5

Module Components

Number	Name	Type	CP
CO-501-A	Discrete Mathematics	Lecture	5

Module Coordinator	**Program Affiliation**	**Mandatory Status**
Dr. Keivan Mallahi Karai	• Mathematics, Modeling and Data Analytics (MMDA)	Mandatory for MMDA Mandatory elective for CS, RIS and SDT

Entry Requirements

Pre-requisites	Co-requisites	Knowledge, Abilities, or Skills
☒ None	☒ None	Basic university mathematics: can be acquired via the Methods Modules “Calculus and Elements of Linear Algebra I + II” or Matrix Algebra and Advanced Calculus.

Frequency	**Forms of Learning and Teaching**
Annually (Spring)	• Lectures (35 hours) • Private Study (90 hours)

Duration	**Workload**
1 semester	125 hours

Recommendations for Preparation

- Some basic familiarity with linear algebra is useful, but not technically required.
- It is recommended to have taken the Methods module: Calculus and Elements of Linear Algebra I + II

Content and Educational Aims

This module is an introductory lecture in discrete mathematics. The lecture consists of two main components, enumerative combinatorics and graph theory. The lecture emphasizes connections of discrete mathematics with other areas of mathematics such as linear algebra and basic probability, and outlines applications to areas of computer science, cryptography, etc. where employment of ideas from discrete mathematics has proven to be fruitful. The first part of the lecture—enumerative combinatorics—deals with several classical enumeration problems (Binomial coefficients, Stirling numbers), counting under group actions and generating function. The second half of the lecture—graph theory—includes a discussion of basic notions such as chromatic number, planarity, matchings in graphs, Ramsey theory, and expanders, and their applications.

Intended Learning Outcomes

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

1. demonstrate their mastery of basic tools in discrete mathematics.
2. develop the ability to use discrete mathematics concepts (such as graphs) to model problems in computer science.
3. analyze the definition of basic combinatorial objects such as graphs, permutations, partitions, etc.
4. formulate and design methods and algorithms for solving applied problems based on concepts from discrete mathematics.

Indicative Literature

J.H. van Lint and R.M. Wilson (2001). A Course in Combinatorics, second edition. Cambridge: Cambridge University Press.

B. Bollobas (1998). Modern Graph Theory, Berlin: Springer.

Usability and Relationship to other Modules • This module is recommended for students pursuing a minor in Mathematics • This module is a good option as an elective module for students in RIS.
Examination Type: Module Examination Assessment Type: Written examination — Duration: 120 min, Weight: 100% Scope: All intended learning outcomes of this module Completion: To pass this module, the examinations must be passed with at least 45%.

7.13 Artificial Intelligence

<table>
<tr><td colspan="3">Module Name
Artificial Intelligence</td><td>Module Code
CO-547</td><td>Level (type)
Year 2
(CORE)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CO-547-A</td><td colspan="3">Artificial Intelligence</td><td>Lecture</td><td>5</td></tr>
<tr><td>Module Coordinator

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

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

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

Pre-requisites

☒ Algorithms and Data Structures or Core Algorithms and Data Structures

Co-requisites

☒ None

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

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

• Class attendance (35 hours)
• Private study (70 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td>Duration

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

125 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.</td></tr>
<tr><td colspan="6">Intended Learning Outcomes

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

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

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.

Examination Type: Module Examination

Assessment Type: Written examination

Duration: 120 min
Weight: 100%

Scope: All intended learning outcomes of the module

Completion: To pass this module, the examinations must be passed with at least 45%.

7.14 Software Engineering and Design

Module Name		Module Code	Level (type)	CP
Software Engineering and Design		SDT-204	Year 2 (CORE)	7.5
Module Components				
Number	Name		Type	CP
SDT-204-A	Software Engineering and Design		Lectures	2.5
SDT-204-B	Software Engineering and Design Project		Project	5
Module Coordinator Prof. Dr. Timofey Bryksin	**Program Affiliation** • Software, Data and Technology (SDT)		**Mandatory Status** Mandatory for SDT	
Entry Requirements Pre-requisites: ☒ Operating Systems Co-requisites: ☒ none Knowledge, Abilities, or Skills		**Frequency** Annually (Spring)	**Forms of Learning and Teaching** • Lecture attendance (17.5 hours) • Tutorial attendance (35 hours) • Independent study (52.5 hours) • Exam preparation (20 hours)	
		Duration 1 semester	**Workload** 125 hours	

Recommendations for Preparation

Students should be familiar with basic concepts of software development, such as data types, control structures, and object-oriented programming.

Content and Educational Aims

Educational Aims:

1. To provide students with a comprehensive understanding of software engineering principles, practices, and techniques, as well as the software development life cycle.
2. To enable students to analyze and design complex software systems, and to apply appropriate software development methodologies and tools.
3. To teach students the best practices for software quality assurance, including testing, debugging, and software maintenance.
4. To foster the development of critical thinking skills, problem-solving skills, and communication skills required for software engineering and design.
5. To introduce students to the latest trends, technologies, and tools in software engineering and design, and to prepare them to work effectively in a rapidly evolving field.
6. To encourage students to apply software engineering and design principles to real-world problems and to develop solutions that meet business and user needs.
7. To prepare students for professional practice in software engineering and design, including the ability to work collaboratively, to manage software development projects, and to apply ethical principles in the workplace.

8. To promote the development of a lifelong learning mindset, and to encourage students to stay current with advances in software engineering and design throughout their careers.

Content:

1. Introduction to Software Engineering and Design
    - Overview of software engineering and design principles and practices
    - Software development life cycle and its phases
    - Roles and responsibilities of software engineers and designers
2. Software Requirements Analysis and Specification
    - Understanding and capturing software requirements
    - Techniques for analyzing requirements
    - Documentation and communication of requirements
    - Requirements validation and verification
3. Software Design Principles and Patterns
    - Object-oriented design principles
    - Design patterns and their applications
    - Modeling techniques and tools
    - Design trade-offs and considerations
4. Software Architecture and Design
    - Architectural styles and patterns
    - Architecture modeling and documentation
    - System and component design
    - Integration and testing of software components
5. Software Testing and Quality Assurance
    - Testing techniques and strategies
    - Test planning and execution
    - Quality metrics and measures
    - Continuous integration and delivery
6. Software Project Management
    - Project planning and estimation
    - Risk management and mitigation
    - Team organization and communication
7. Agile methodologies and practices
    - Emerging Technologies and Trends in Software Engineering and Design
    - Cloud computing and software-as-a-service (SaaS)
    - Mobile and web application development
    - Artificial intelligence and machine learning
    - Blockchain technology and distributed systems
8. Ethical and Legal Issues in Software Engineering and Design
    - Intellectual property and copyright laws
    - Privacy and data protection
    - Software piracy and licensing
    - Ethical considerations in software development and use.

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. understand the software development life cycle and its phases, and be able to apply appropriate software development methodologies and tools to various stages of the process.
2. apply software design principles and patterns to develop complex software systems that meet business and user needs.
3. apply appropriate software quality assurance practices, including testing, debugging, and software maintenance, to ensure the quality of software products.
4. develop critical thinking and problem-solving skills to identify, analyze, and solve software engineering and design problems.
5. develop effective communication and collaboration skills required for professional practice in software engineering and design.
6. analyze and evaluate software requirements and specifications, and develop software that meets those requirements.

7. apply appropriate software architecture and design principles and patterns to design and develop software systems.
8. apply risk management strategies to identify, analyze, and mitigate risks associated with software development projects.
9. understand the ethical and legal considerations associated with software development, and apply ethical principles in the workplace.
10. stay current with advances in software engineering and design, and demonstrate a commitment to lifelong learning in the field.

Indicative Literature

- Ian Sommerville: Software Engineering, 10th edition, Pearson, 2015.
- Roger S. Pressman: Software Engineering: A Practitioner's Approach, 8th edition, McGraw-Hill, 2014.
- Robert Martin: Agile Software Development, Principles, Patterns, and Practices, Pearson, 2002.
- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.
- Martin Fowler: Patterns of Enterprise Application Architecture, Addison-Wesley, 2002.
- Grady Booch, James Rumbaugh, Ivar Jacobson: The Unified Modeling Language User Guide, 2nd edition, Addison-Wesley, 2005.
- Steve McConnell: Code Complete: A Practical Handbook of Software Construction, 2nd edition, Microsoft Press, 2004.
- Kent Beck: Test-Driven Development: By Example, Addison-Wesley, 2002.
- Michael Feathers: Working Effectively with Legacy Code, Prentice Hall, 2004.
- Craig Larman: Agile and Iterative Development: A Manager's Guide, Addison-Wesley, 2004.

Usability and Relationship to other Modules

- The knowledge and skills acquired in this module will be useful for students planning to pursue a career in software development or data science, or continue their studies in advanced software development or data science modules. The module will also be beneficial for students planning to pursue a career in other related fields such as IT management, software testing, or software project management.

Examination Type: Module Component Examination

Component 1: Lecture

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

Scope: All theoretical intended learning outcomes of the module

Component 2: Project

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%.

7.15 Databases

<table>
<tr><td colspan="2">Module Name
Databases</td><td>Module Code
SDT-205</td><td>Level (type)
Year 2 (CORE)</td><td>CP
5</td></tr>
<tr><td colspan="5">Module Components</td></tr>
<tr><td>Number</td><td colspan="2">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>SDT-205-A</td><td colspan="2">Databases</td><td>Lecture</td><td>2.5</td></tr>
<tr><td>SDT-205-B</td><td colspan="2">Databases Project</td><td>Project</td><td>2.5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Aleksander Omelchenko</td><td colspan="2">Program Affiliation

● Software, Data and Technology (SDT)</td><td colspan="2">Mandatory Status

Mandatory for Minor in Software Development
Mandatory elective for SDT</td></tr>
<tr><td colspan="2" rowspan="2">Entry Requirements

Pre-requisites: ☒ Core Algorithms and Data Structures
Co-requisites: ☒ none
Knowledge, Abilities, or Skills:</td><td>Frequency

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

● Lecture attendance (17.5 hours)
● Tutorial attendance (17.5 hours)
● Independent study (70 hours)
● Exam preparation (20 hours)</td></tr>
<tr><td>Duration

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

125 hours</td></tr>
</table>

Recommendations for Preparation

Working knowledge of basic algorithms and data structures, such as trees, is required as well as familiarity with an object-oriented programming language such as Kotlin. 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/).

Content and Educational Aims

Content:

- Introduction to databases and data management
- Database design and modeling
- Relational database management systems (RDBMS)
- SQL for data manipulation and querying
- Database normalization and optimization
- NoSQL databases and data warehousing
- Database security and administration

Educational Aims:

- To provide students with a strong foundation in database design, modeling and management
- To teach students how to use SQL to manipulate and query data in a relational database
- To enable students to design and implement efficient and normalized databases
- To familiarize students with the unique features of NoSQL databases and data warehousing
- To prepare students for using databases in various fields such as software development, data science, and business.

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. design and model databases.
2. use SQL to manipulate and query data in a relational database.
3. implement and optimize databases using normalization techniques.
4. use NoSQL databases and data warehousing.
5. use databases in various fields such as software development, data science, and business.
6. manage and secure databases

Indicative Literature

- Elmasri Navathe: Fundamentals of Database Systems, 7th edition, Addison-Wesley, 2014.
- C. J. Date: An Introduction to Database Systems, 8th edition, Addison-Wesley, 2004.
- Ramez Elmasri, Shamkant B. Navathe: Database Systems: Concepts, Design and Applications, Prentice-Hall, 1999.
- Kyle Simpson: You Don't Know JS: Async & Performance, O'Reilly Media, 2014.
- Martin Kleppmann: Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly Media, 2017.

Usability and Relationship to other Modules

- Familiarity with databases is essential for students who wish to specialize in software development and data science. This module will provide a solid foundation in database design, modeling, and management, including the use of SQL to manipulate and query data. Additionally, this module will introduce advanced concepts of database management and NoSQL databases, which are needed in advanced programming-oriented modules in the 3rd year of the SDT program.

Examination Type: Module Component Examination

Component 1: Lecture

Assessment: Written examination

Duration: 60 min
Weight: 50%

Scope: All theoretical intended learning outcomes of the module

Component 2: Project

Assessment: Practical assessment (Programming assignments)

Weight: 50%

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%.

7.16 Deep Learning

Module Name Deep Learning			**Module Code** MCSSE-AI-01	**Level (type)** YEAR 1/2	**CP** 5
Module Components					
Number	Name			Type	CP
MCSSE-AI-01	Deep Learning			Lecture	5
Module Coordinator Prof. Dr. Alexander Omelchenko	**Program Affiliation** • MSc Computer Science & Software Engineering (CSSE)			**Mandatory Status** Mandatory elective for CSSE and BSc SDT	
Entry Requirements Pre-requisites ☒ none	Co-requisites ☒ none	Knowledge, Abilities, or Skills • Strong knowledge and abilities in mathematics (linear algebra, calculus).	**Frequency** Annually (Fall) **Duration** 1 Semester	**Forms of Learning and Teaching** • Lectures (35 hours) • Private study (70 hours) • Exam preparation (20 hours) **Workload** 125 hours	

Recommendations for Preparation

This module is recommended for students that have been exposed to core knowledge in machine learning / statistical learning on undergraduate level. Students without this background knowledge can still join since required core knowledge is re-introduced. Preparation via auxiliary literature or online courses will facilitate the start into the module.

Content and Educational Aims

In machine learning we aim at extracting meaningful representations, patterns and regularities from high-dimensional data. In recent years, researchers from various disciplines have developed “deep” hierarchical models, i.e. models that consist of multiple layers of nonlinear processing. An important property of these models is that they can “learn” by reusing and combining intermediate concepts, so that these models can be used successfully in a variety of domains, including information retrieval, natural language processing, and visual object detection. After a brief introduction into core knowledge related to training, model evaluation and multilayer perceptrons, this module focuses on the exposing students to deep learning techniques including convolutional and recurrent neural networks, autoencoders, generative adversarial networks and reinforcement learning. The central aim is hence to enable students to critically assess and apply modern methods in machine learning.

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. understand core techniques to train neural networks
2. select from modern neural network architectures the most appropriate method (e.g. convolutional and recurrent neural networks) based on given input data
3. contrast different recent unsupervised learning methods including autoencoders and generative adversarial networks
4. describe techniques in reinforcement learning.

Indicative Literature

Ian Goodfellow, Yoshua Bengio, Aaron Courville: Deep Learning, MIT Press, 2016.

Aurélien Géron: Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, 2nd Edition, O'Reilly, 2019. Christopher M. Bishop: Pattern Recognition and Machine Learning, Springer, 2006. Charu C. Aggarwal: Neural Networks and Deep Learning – A Textbook, Springer, 2018.
Usability and Relationship to other Modules • While the graduate level modules "Data Analytics" and "Machine Learning" provide an applied introduction to the field and are therefore recommended for students with a focus on Software Engineering or Cybersecurity, this module complements the undergraduate module "Machine Learning" or can be used independently as a strong introduction to the field of Deep Learning.
Examination Type: Module Examination Assessment: Written Examination Duration: 120 min Weight: 100% Scope: All intended learning outcomes of the module. Completion: To pass this module, the examinations must be passed with at least 45%.

7.17 Stochastic Modeling and Financial Mathematics

<table>
<tr><td colspan="3">Module Name
Stochastic Modeling and Financial Mathematics</td><td>Module Code
CA-S-MMDA-803</td><td>Level (type)
Year 2 and 3 (Specialization)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CA-MMDA-803</td><td colspan="3">Stochastic Modeling and Financial Mathematics</td><td>Lecture</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Sören Petrat</td><td colspan="3">Program Affiliation

• Mathematics, Modeling, and Data Analytics (MMDA)</td><td colspan="2">Mandatory Status

Mandatory elective for MMDA, PHDS,RIS and SDT</td></tr>
<tr><td colspan="3" rowspan="2">Entry Requirements

Pre-requisites: ☒ Matrix Algebra & Advanced Calculus I & II
Co-requisites: ☒ none
Knowledge, Abilities, or Skills: Good command of Calculus, Linear Algebra, and basic probability basic Python programming</td><td>Frequency

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

- Lectures (35 hours)
- Private Study (90 hours)</td></tr>
<tr><td>Duration

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

125 hours</td></tr>
</table>

Recommendations for Preparation

- Review the content of Matrix Algebra & Advanced Calculus II
- Review Python programming
- Pre-install Anaconda Python on your own laptop and know how to edit and start simple Python programs in a Python IDE like Spyder (which comes bundled as part of Anaconda Python).

Content and Educational Aims

This module is a first hands-on introduction to stochastic modeling. Examples will mostly come from the area of Financial Mathematics, so that this module plays a central role in the education of students interested in Quantitative Finance and Mathematical Economics. The module is taught as an integrated lecture-lab, where short theoretical units are interspersed with interactive computation and computer experiments.

Topics include a short introduction to the basic notions of financial mathematics, binomial tree models, discrete Brownian paths, stochastic integrals and ODEs, Ito's Lemma, Monte-Carlo methods, finite differences solutions, the Black-Scholes equation, and an introduction to time series analysis, parameter estimation, and calibration. Towards the end, the Fokker-Planck equation, Ornstein-Uhlenbeck processes, and nonlinear Stochastic Partial Differential Equations are discussed, and connections to applications in physics and other areas of mathematics are made. Students will program and explore all basic techniques in a numerical programming environment and apply these algorithms to real data whenever possible.

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. apply fundamental concepts of deterministic and stochastic modeling;
2. design, conduct, and interpret controlled in-silico scientific experiments;
3. analyze the basic concepts of financial mathematics and their role in finance;
4. write computer code for basic financial calculations, binomial trees, stochastic differential equations, stochastic integrals and time series analysis;
5. compare their programs and predictions in the context of real data;
6. demonstrate the usage of a version control system for collaboration and the submission of code and reports.

Indicative Literature

Y.-D. Lyuu (2002). Financial Engineering and Computation - Principles, Mathematics, Algorithms. Cambridge: Cambridge University Press.

J.C. Hull (2015). Options, Futures and other Derivatives, 9th edition. New York: Pearson.

A. Etheridge (2002). A Course in Financial Calculus. Cambridge: Cambridge University Press.

D.J. Higham (2001). An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations, SIAM Rev. 43(3):525-546.

D.J. Higham (2004). Black-Scholes Option Valuation for Scientific Computing Students, Computing in Science & Engineering 6(6):72-79.

Usability and Relationship to other Modules

- This module is part of the core education in Mathematics, Modeling and Data Analytics.
- It is also valuable for students in Physics and Data Science, Computer Science, Data Engineering, RIS, SDT, and ECE, either as part of a minor in Mathematics, or as an elective module.

Examination Type: Module Examination

Assessment Type: Portfolio (programming assessments, project) Weight: 100%

Scope: All intended learning outcomes of this module

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

7.18 Optimization Methods

Module Name	Module Code	Level (type)	CP
Optimization Methods	SDT-301	Year 3 (specialization)	5

Module Components			
Number	Name	Type	CP
SDT-301-A	Optimization Methods	Lecture	2.5
SDT-301-B	Optimization Methods Tutorial	Tutorial	2.5

Module Coordinator	**Program Affiliation**	**Mandatory Status**
Prof. Dr. Aleksander Omelchenko	• Software, Data and Technology (SDT)	Mandatory elective for SDT

Entry Requirements			**Frequency**	**Forms of Learning and Teaching**
Pre-requisites	Co-requisites	Knowledge, Abilities, or Skills	Annually (Fall)	• Lecture attendance (17.5 hours) • Tutorial attendance (35 hours) • Independent study (52.5 hours) • Exam preparation (20 hours)
☒ Linear Algebra or Matrix Algebra & Advanced Calculus II ☒ Programming in Python and C++ or Programming in C/C++	☒ none		**Duration** 1 semester	**Workload** 125 hours

Recommendations for Preparation

- Familiarity with basic mathematical concepts such as linear algebra, and calculus.
- Familiarity with basic programming concepts and experience with a programming language such as Python.

Content and Educational Aims

Optimization methods are a set of mathematical techniques used to find the best solution to a problem, given a set of constraints. In this module, students will learn about different optimization techniques such as linear and non-linear programming, gradient-based and heuristic methods, and their applications in various fields such as engineering, finance, and operations research. They will also learn about the implementation of optimization algorithms using programming languages such as Python. This module serves as an introduction to optimization methods and provides students with a solid foundation for more advanced coursework in the program and the industry.

Content:

- Linear programming: Formulation of LP problems, simplex method, duality theory, sensitivity analysis, and applications.

- Non-linear programming: Unconstrained optimization, optimization under constraints, optimization with inequality constraints, optimization with equality constraints, optimization with nonlinear constraints, and applications.
- Gradient-based optimization methods: Newton and quasi-Newton methods, conjugate gradient and conjugate direction methods, and optimization of multivariable functions.
- Heuristic optimization methods: Genetic algorithms, simulated annealing, tabu search, and other metaheuristics.
- Applications of optimization methods: Linear and non-linear programming in engineering, finance, and operations research.
- Implementation of optimization algorithms using programming languages such as Python.
- Analysis and interpretation of the results of optimization problems.
- Trade-offs and limitations of different optimization methods.
- Communication and presentation of optimization results using mathematical notation and technical language.
- Ethical and societal implications of optimization methods.

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. Understand the concepts and principles of optimization methods, including linear and non-linear programming, gradient-based and heuristic methods.
2. Apply optimization techniques to solve real-world problems in various fields such as engineering, finance, and operations research.
3. Understand the trade-offs and limitations of different optimization methods and select the appropriate method for a given problem.
4. Implement optimization algorithms using programming languages such as Python.
5. Analyze and interpret the results of optimization problems and make recommendations based on the results.
6. Communicate effectively about optimization methods using mathematical notation and technical language.
7. Develop an understanding of the ethical and societal implications of optimization methods.

Indicative Literature

Jorge Nocedal, Stephen J. Wright: Numerical Optimization, 2nd edition, Springer, 2006.

Andreu Mas-Colell, Michael D. Whinston, Jerry R. Green: Microeconomic Theory, Oxford University Press, 1995.

Dimitri P. Bertsekas: Nonlinear Programming, 2nd edition, Athena Scientific, 1999.

John C. Hull: Options, Futures, and Other Derivatives, 9th edition, Prentice Hall, 2014.

David G. Luenberger, Yinyu Ye: Linear and Nonlinear Programming, 3rd edition, Springer, 2008

Usability and Relationship to other Modules

- Optimization methods are widely used in fields such as engineering, finance, operations research and computer science. This module provides a solid foundation in optimization techniques such as linear and non-linear programming, gradient-based and heuristic methods, and their applications in various fields. It also covers the implementation of optimization algorithms using programming languages such as Python, which is essential for students who wish to pursue advanced coursework in related fields or pursue careers in fields such as engineering, finance, operations research, and computer science. Understanding optimization methods is also important for making informed decisions in various fields as well as solving real-world problems.

Examination Type: Module Component Examination

Component 1: Lecture

Assessment: Written examination Duration: 60 min

Weight: 50%

Scope: All theoretical intended learning outcomes of the module

Component 2: Tutorial

Assessment: Practical assessment (Programming assignments) Weight: 50%

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%.

7.19 Natural Language Processing

Module Name		**Module Code**	**Level (type)**	**CP**
Natural Language Processing		SDT-305	Year 3 (specialization)	5
Module Components				
Number	Name		Type	CP
SDT-305-A	Natural Language Processing		Lecture	5
Module Coordinator Prof. Dr. Aleksander Omelchenko	**Program Affiliation** • Software, Data and Technology (SDT)		**Mandatory Status** Mandatory elective for SDT	
Entry Requirements Pre-requisites ☒ Linear Algebra or Matrix Algebra & Advanced Calculus II ☒ Programming in Python and C++ or Programming in C/C++ ☒ Machine Learning	Co-requisites ☒ none	Knowledge, Abilities, or Skills ☒ Familiarity with basics of Deep Learning, Python and Pytorch is recommended.	**Frequency** Annually (Spring) **Duration** 1 semester	**Forms of Learning and Teaching** • Lecture attendance (35 hours) • Independent study (70 hours) • Exam preparation (20 hours) **Workload** 125 hours
Recommendations for Preparation Familiarity with basic mathematical concepts such as linear algebra, and calculus. Familiarity with basic programming concepts and experience with a programming language and such as Python.				

Content and Educational Aims

Students learn the fundamental concepts and techniques of natural language processing (NLP) and how to apply them to analyze, understand and generate human language. They gain hands-on experience with popular NLP libraries and frameworks in Python. Students also learn about the key NLP tasks such as tokenization, part-of-speech tagging, syntactic parsing, named entity recognition, about the key NLP applications such as text classification, sentiment analysis, machine translation, and text generation, and about the challenges and limitations of NLP and the state-of-the-art research in the field.

Content:

- Introduction to NLP, including the history and current state of the field
- Key NLP tasks such as tokenization, part-of-speech tagging, syntactic parsing
- Key NLP applications such as text classification, sentiment analysis, machine translation, and text generation
- Techniques for working with large text corpora, such as text pre-processing, data cleaning and data preparation
- Techniques for building NLP systems, such as statistical models, and neural networks
- Techniques for evaluating NLP systems
- State-of-the-art research in NLP
- Hands-on experience with popular NLP libraries and frameworks in Python

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. understand and apply fundamental concepts and techniques of natural language processing (NLP) to analyze, understand, and generate human language.
2. identify and perform the key NLP tasks such as tokenization, part-of-speech tagging, syntactic parsing, semantic role labeling, named entity recognition, and coreference resolution.
3. identify and apply the key NLP applications such as text classification, sentiment analysis, machine translation, and text generation.
4. evaluate NLP systems and understand the challenges and limitations of NLP.
5. understand the state-of-the-art research in NLP, learn how to read and reproduce modern NLP research papers
6. use popular NLP libraries and frameworks in Python to implement NLP tasks and applications.

Indicative Literature

Jacob Eisenstein: Natural Language Processing, 1st edition, Cambridge University Press, 2020.

James H. Martin: Natural Language Processing with GATE, 1st edition, Cambridge University Press, 2016.

Dan Jurafsky and James H. Martin: Speech and Language Processing, 3rd edition, Prentice Hall, 2020.

Yoav Goldberg: Neural Network Methods for Natural Language Processing, 1st edition, Morgan & Claypool Publishers, 2017.

Usability and Relationship to other Modules

- This module will provide students with a solid understanding of the fundamental concepts and techniques of natural language processing, as well as hands-on experience with popular NLP libraries and frameworks in Python. The module will prepare students for more advanced coursework in the program and for professional development in the field of software, data and technology, particularly in areas such as text mining, information retrieval, and language-based AI.

Examination Type: Module Examination

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

Scope: All theoretical intended learning outcomes of the module

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

7.20 Distributed Algorithms

Module Name			Module Code	Level (type)	CP
Distributed Algorithms			CA-S-CS-803	Year 3 (Specialization)	5
Module Components					
Number	Name			Type	CP
CA-CS-803	Distributed Algorithms			Lecture	5
Module Coordinator Dr. Kinga Lipskoch	**Program Affiliation** • Computer Science (CS)			**Mandatory Status** Mandatory elective for CS, RIS and SDT	
Entry Requirements Pre-requisites ☒ Algorithms and Data Structures or Core Algorithms and Data Structures	Co-requisites ☒ None	Knowledge, Abilities, or Skills	**Frequency** Annually (Fall or Spring)	**Forms of Learning and Teaching** • Class attendance (35 hours) • Private study (70 hours) • Exam preparation (20 hours)	
			Duration 1 semester	**Workload** 125 hours	
Recommendations for Preparation None					

Content and Educational Aims

Distributed algorithms are the foundation of modern distributed computing systems. They are characterized by a lack of knowledge of a global state, a lack of knowledge of a global time, and inherent non-determinism in their execution. The module introduces basic distributed algorithms using an abstract formal model, which is centered on the notion of a transition system. The topics covered are logical clocks, distributed snapshots, mutual exclusion algorithms, wave algorithms, election algorithms, reliable broadcast algorithms, and distributed consensus algorithms. Process algebras are introduced as another formalism to describe distributed and concurrent systems.

The distributed algorithms introduced in this module form the foundation of computing systems that have to be scalable and fault-tolerant, e.g., large-scale distributed non-standard databases or distributed file systems. The module is recommended for students interested in the design of scalable distributed computing systems.

Intended Learning Outcomes

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

1. describe and analyze distributed algorithms using formal methods such as transition systems;
2. explain different algorithms to solve election problems;
3. illustrate the limitations of time to order events and how logical clocks and vector clocks overcome these limitations;
4. apply distributed algorithms to produce consistent snapshots of distributed computations;
5. describe the differences among wave algorithms for different topologies;
6. analyze and implement distributed consensus algorithms such as Paxos and Raft;
7. use a process algebra such as communicating sequential processes or □-calculus to model distributed algorithms.

Indicative Literature

Maarten van Steen, Andrew S. Tanenbaum: Distributed Systems, 3rd edition, Pearson Education, 2017.

Nancy A. Lynch: Distributed Algorithms, Morgan Kaufmann, 1996.

Usability and Relationship to other Modules

.

Examination Type: Module Examination

Assessment Type: Written examination

Duration: 120 min
Weight: 100%

Scope: All intended learning outcomes of the module

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

7.21 Computer Networks

Module Name	Module Code	Level (type)	CP
Computer Networks	CO-564	Year 2 (CORE)	5

Module Components

Number	Name	Type	CP
CO-564-A	Computer Networks	Lecture	5

Module Coordinator	Program Affiliation	Mandatory Status
Prof. Dr. Jürgen Schönwälder	• Computer Science (CS)	Mandatory for CS Mandatory elective for SDT

Entry Requirements

Pre-requisites	Co-requisites	Knowledge, Abilities, or Skills
☒ Algorithms and Data Structures or Core Algorithms and Data Structures	☒ Operating Systems	

Frequency	Forms of Learning and Teaching
Annually (Fall)	• Class attendance (35 hours) • Private study (70 hours) • Exam preparation (20 hours)

Duration	Workload
1 semester	125 hours

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://www.youtube.com/watch

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 module 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.

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;
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

- 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

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

7.22 Databases Internals

<table>
<tr><td colspan="2">Module Name
Databases Internals</td><td>Module Code
SDT-302</td><td>Level (type)
Year 3 (Specialization)</td><td>CP
5</td></tr>
<tr><td colspan="5">Module Components</td></tr>
<tr><td>Number</td><td colspan="2">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>SDT-302-A</td><td colspan="2">Databases Internals</td><td>Lectures</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Aleksander Omelchenko</td><td colspan="2">Program Affiliation

• Software, Data and Technology (SDT)</td><td colspan="2">Mandatory Status

Mandatory elective for SDT</td></tr>
<tr><td colspan="2" rowspan="2">Entry Requirements

Pre-requisites
☒ Databases
☒ Development in JVM Languages

Co-requisites
☒ none

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

Annually (Fall)</td><td colspan="2">Forms of Learning and Teaching
• Lecture attendance (17.5 hours)
• Team work (35 hours)
• Independent study (52.5 hours)
• Exam preparation (20 hours)</td></tr>
<tr><td>Duration

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

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

• Familiarity with basic concepts of database management systems (DBMS) and SQL.
• Familiarity with basic concepts of data structures and algorithms.
• Familiarity with basic programming concepts and experience with a programming language such as Kotlin</td></tr>
<tr><td colspan="5">Content and Educational Aims

This discipline is aimed at gaining skills in building data storage systems and operating with them in an effective way. During the course, students will get familiar hor a relational database engine works internally. They will look at the data structures, algorithms and mathematics, which allow for efficient execution of complex search queries in pretty big databases, and will try to implement a few components of a toy database.

Content:
• Principles of building relational DBMS
• Some issues of building distributed DBMS
• Columnar DBMS
• Non-classical DBMS types: XML, graph, object
• Elements of multidimensional indexing
• The task of configuring DBMS</td></tr>
<tr><td colspan="5">Intended Learning Outcomes

Upon completion of this module, students will be able to:
1. describe the principles of storing and accessing data records on disk, performing relational operations such as selections and joins, building and using indexes appropriately;
2. apply cost-based optimization techniques to the problems of choosing the optimal way of query execution
3. implement lock-based and timestamp-ordering based schedulers in transactional systems</td></tr>
</table>

Indicative Literature Thomas Connolly, Carolyn Begg: Database Systems: A Practical Approach to Design, Implementation and Management, 6th edition, Addison-Wesley, 2016. C. J. Date: An Introduction to Database Systems, 8th edition, Addison-Wesley, 2004. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom: Database Systems: The Complete Book, 2nd edition, Prentice Hall, 2002. Ramez Elmasri, Shamkant B. Navathe: Fundamentals of Database Systems, 7th edition, Pearson, 2018. Michael Stonebraker, Joseph M. Hellerstein, James Hamilton: Readings in Database Systems, 5th edition, Morgan Kaufmann Publishers, 2018
Usability and Relationship to other Modules • An understanding of the internal workings of databases is essential for students pursuing careers in computer science, software engineering, and related fields. This module provides a solid foundation in the internal workings of database management systems, including storage structures, query processing, and transaction management. It also covers the implementation of databases using programming languages such as SQL. This knowledge is crucial for students who wish to pursue advanced coursework in computer science and related fields, as well as for those who wish to pursue careers in fields such as software development, data management and data administration.
Examination Type: Module Examination Assessment: Written examination — Duration: 120 min, Weight: 100% Scope: All theoretical intended learning outcomes of the module Completion: To pass this module, the examination has to be passed with at least 45%

7.23 Integrated Development and IT Operations

<table>
<tr><td colspan="3">Module Name
Integrated Development and IT Operations</td><td>Module Code
SDT-306</td><td>Level (type)
Year 3
(Specialization)</td><td>CP
5.0</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>SDT-306-A</td><td colspan="3">Integrated Development and IT Operations</td><td>Lectures</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Aleksander Omelchenko</td><td colspan="3">Program Affiliation

● Software, Data and Technology (SDT)</td><td colspan="2">Mandatory Status

Mandatory elective for SDT</td></tr>
<tr><td rowspan="2">Entry Requirements

Pre-requisites

☒ Operating Systems

☒ Software Engineering</td><td rowspan="2">

Co-requisites

☒ none</td><td rowspan="2">

Knowledge, Abilities, or Skills

Any programming language knowledge, basic knowledge in networks</td><td>Frequency

Annually
(Spring)</td><td colspan="2">Forms of Learning and Teaching
● Lecture attendance (35 hours)
● Independent study (70 hours)
● Exam preparation (20 hours)</td></tr>
<tr><td>Duration

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

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

Students should be familiar with basic concepts of software development, including version control, testing, and deployment, with basic Linux command line usage and basic administration tasks.</td></tr>
<tr><td colspan="6">Content and Educational Aims
This module provides an introduction to the principles, practices, and tools of DevOps, a set of methodologies that aim to improve the collaboration, communication and integration between software development and IT operations teams. In this module, students will learn about the key concepts and practices of DevOps, including continuous integration, continuous delivery, and infrastructure as code. They will also learn about the tools and technologies used in DevOps, such as version control systems, containerization, and cloud computing. This module will give students a solid understanding of the principles and practices of DevOps and how they can be applied to improve the software development process and increase efficiency.

Content:

● Key concepts of DevOps, such as continuous integration, continuous delivery, and infrastructure as code
● Collaboration, communication and integration between software development and IT operations teams
● Version control systems and their role in DevOps
● Containerization and its usage in DevOps
● Cloud computing and its role in DevOps
● Automation and monitoring in DevOps
● Security considerations in DevOps
● Best practices and real-world examples of DevOps in action.</td></tr>
</table>

Intended Learning Outcomes

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

1. understand the key principles and practices of DevOps and how they can be applied to software development projects
2. use version control systems such as Git to manage and track changes to code
3. set up and configure continuous integration and delivery pipelines using tools like Jenkins
4. use infrastructure as code tools like Docker and Terraform to automate the deployment and management of applications and infrastructure
5. monitor and log the performance and reliability of applications and systems using tools such as Splunk and Grafana
6. collaborate effectively with development and operations teams to improve the efficiency and reliability of software delivery.

Indicative Literature

- Jez Humble, David Farley: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley, 2010.
- Gene Kim, Kevin Behr, George Spafford: The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win, IT Revolution, 2013.
- Patrick Debois: The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations, IT Revolution Press, 2016.
- Michael Nygard: Release It!: Design and Deploy Production-Ready Software, Pragmatic Bookshelf, 2007.
- Kim, Gene, and John Willis. The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise. IT Revolution Press, 2018.

Usability and Relationship to other Modules

- DevOps is a rapidly growing field that combines software development and IT operations to improve the software development process and increase efficiency. This module provides a solid foundation in the principles, practices, and tools of DevOps, including continuous integration, continuous delivery, and infrastructure as code. It also covers the tools and technologies used in DevOps, such as version control systems, containerization, and cloud computing. This knowledge is crucial for students who wish to pursue careers in software development, IT operations, and related fields. It will also help students to understand how to improve the software development process and increase efficiency in their organizations.

Examination Type: Module Examination

Assessment: Written examination

Duration: 120 min
Weight: 100%

Scope: All theoretical intended learning outcomes of the module

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

7.24 Parallel Programming

Module Name	Module Code	Level (type)	CP
Parallel Programming	SDT-303	Year 3 (Specialization)	5.0

Module Components

Number	Name	Type	CP
SDT-303-A	Parallel Programming	Lectures	5

Module Coordinator	Program Affiliation	Mandatory Status
Prof. Dr. Kirill Krinkin	• Software, Data and Technology (SDT)	Mandatory elective for SDT

Entry Requirements

Pre-requisites	Co-requisites	Knowledge, Abilities, or Skills
☒ Operating Systems ☒ Development in JVM Languages	☒ none	Strong knowledge of basic data structures and algorithms, basic programming skills in Kotlin

Frequency	Forms of Learning and Teaching
Annually (Fall)	• Lecture attendance (35 hours) • Independent study (70 hours) • Exam preparation (20 hours)

Duration	Workload
	125 hours

Recommendations for Preparation

- Fundamentals of computer science, including computer architecture, algorithms and data structures, and programming skills in Java or Kotlin.

Content and Educational Aims

This module introduces the principles and practices of concurrent programming, including the state-of-the-art and modern approaches to building concurrent algorithms.

Content:

- Key concepts in concurrent programming, such as the shared-memory model, linearizability correctness property, and the standard progress guarantees
- Lock-based synchronization
- Basic non-blocking concurrent algorithms, such as the Michael-Scott queue
- Modern concurrent algorithms, such as the Fetch-And-Add-based queues
- Various techniques used in concurrent algorithms, such as helping, descriptors, and flat combining

Intended Learning Outcomes

Upon completion of this module, students will:

1. Understand the basic concepts of concurrent programming
2. Able to implement existing concurrent algorithm algorithms
3. Able to develop new concurrent algorithms either using the learned techniques and approaches or even by introducing new ones
4. Able to reason about the correctness of concurrent algorithms

Indicative Literature

Michael J. Quinn: Parallel Programming in C with MPI and OpenMP, McGraw-Hill, 2003.
Peter Pacheco: An Introduction to Parallel Programming, Morgan Kaufmann Publishers, 2011.
Grama, Ananth, et al. Introduction to parallel computing. Pearson Education India, 2003.
William Gropp, Ewing Lusk, Anthony Skjellum: Using MPI: Portable Parallel Programming with the Message-Passing Interface, 2nd edition, MIT Press, 1999.
Andrew S. Tanenbaum, Martín Casado: Computer Networks, 5th edition, Prentice Hall, 2010.

Herlihy, Maurice, Nir Shavit, Victor Luchangco, and Michael Spear. The art of multiprocessor programming. Newnes, 2020.

Usability and Relationship to other Modules

- Nowadays, concurrent programming is crucial for students pursuing careers in computer science, software engineering, and related fields. This module provides a solid foundation in the principles and practices of developing concurrent algorithms.

Examination Type: Module Examination

Assessment: Written examination

Duration: 120 min
Weight: 100%

Scope: All theoretical intended learning outcomes of the module

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

7.25 Formal Languages and Parsers

Module Name Formal Languages and Parsers			**Module Code** SDT-304	**Level (type)** Year 3 (Specialization)	**CP** 5
Module Components					
Number	Name			Type	CP
SDT-304-A	Formal Languages and Parsers			Lectures	2.5
SDT-304-B	Formal Languages and Parsers Tutorial			Tutorial	2.5
Module Coordinator Prof. Dr. Anton Podkopaev	**Program Affiliation** • Software, Data and Technology (SDT)			**Mandatory Status** Mandatory elective for SDT	
Entry Requirements Pre-requisites ☒ Programming in Python and C++ or Programming in C/C++ ☒ Discrete Mathematics	Co-requisites ☒ none	Knowledge, Abilities, or Skills	**Frequency** Annually (Fall)	**Forms of Learning and Teaching** • Lecture attendance (17.5 hours) • Tutorial attendance (35 hours) • Independent study (52.5 hours) • Exam preparation (20 hours)	
			Duration 1 semester	**Workload** 125 hours	

Recommendations for Preparation
Familiarity with basic mathematical concepts such as set theory, logic, and discrete mathematics. Familiarity with basic programming concepts and experience with a programming language such as Python or Kotlin.

Content and Educational Aims

The aim of this discipline is to give students theoretical knowledge and practical skills in the fundamentals of the theory of formal languages. Considerable attention is paid to issues related to the theoretical aspects of the syntax and semantics of programming languages, as well as to the creation of effective algorithms for lexical and syntactic analysis of program code. During the course, students will: − learn basic methods of parsing; main approaches used for generating object code. − be able to describe programming language syntax, using various approaches; construct language semantics using different approaches; apply regular expressions for lexical analysis; create algorithms for efficient syntactic analysis of program code; develop JIT compilers. − Have gained skills in application of methods for describing the syntax and semantics of programming languages using various approaches; methods for creating effective algorithms for lexical and syntactic analysis of program code.

Content:

- Deterministic and nondeterministic finite automata
- Deterministic and nondeterministic pushdown automata
- Turing machines and linear-bounded turing machines
- Recursive descent parsing
- Lexer and parser generators
- LL and LR grammars
- Parser expression grammars
- Combinatory parsing
- Regular languages and expressions
- Context-free languages and grammars
- Context-sensitive, recursively enumerable languages and their useful subsets

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. know the basic modern principles and approaches to the construction of formal languages. Have gained skill in finding relevant information on formal languages and grammars.
2. can design a language with syntax which can be effectively parsed using modern methods. Know the classes of grammars most useful in practice. Know the theoretical complexity bounds for relevant classes of grammars.
3. know the algorithms for syntactic analysis. Generate a lexer and a parser from the language specification. Effectively implement recursive descent parsers. Effectively implement parser combinators.
4. know the basic methods of working with finite state machines. Be able to present and justify the choice of methods of working with a given formal language. Have the skills to describe a given programming language syntax with regular expressions and context-free grammars.

Indicative Literature

- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman: Introduction to Automata Theory, Languages, and Computation, 3rd edition, Addison-Wesley, 2007.
- Michael Sipser: Introduction to the Theory of Computation, 3rd edition, Cengage Learning, 2013.
- Martin Davis, Ron Sigal, Elaine J. Weyuker: Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science, 2nd edition, Elsevier, 1994.
- Dexter Kozen: Automata and Computability, Springer, 1997.
- Harry Lewis, Christos Papadimitriou: Elements of the Theory of Computation, 2nd edition, Prentice Hall, 1998

Usability and Relationship to other Modules

- Formal languages and automata form the theoretical foundations of computer science and are essential for understanding the properties and limits of computation. This module provides a solid foundation in formal languages and automata theory, including regular languages, context-free languages, Turing machines, and decidability. It also covers the applications of formal languages and automata theory in various fields such as compilers, parsing, and verification. This knowledge is crucial for students who wish to pursue advanced coursework in computer science, theoretical computer science, and related fields, as well as for those who wish to pursue careers in fields such as software engineering, verification, and natural language processing.

Examination Type: Module Component Examination

Component 1: Lecture

Assessment: Written examination — Duration: 60 min

Weight: 50%

Scope: All theoretical intended learning outcomes of the module

Component 2: Tutorial

Assessment: Practical assessment (Programming assignments)

Weight: 50%

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%.

7.26 Compilers

Module Name Compilers		**Module Code** SDT-307	**Level (type)** Year 3 (Specialization)	**CP** 5
Module Components				
Number	Name		Type	CP
SDT-307-A	Compilers		Lecture	2.5
SDT-307-B	Compilers Project		Project	2.5
Module Coordinator Prof. Dr. Anton Podkopaev	**Program Affiliation** • Software, Data and Technology (SDT)		**Mandatory Status** Mandatory elective for SDT	
Entry Requirements Pre-requisites ☒ Formal Languages and Parsers ☒ Functional Programming	Co-requisites ☒ none Knowledge, Abilities, or Skills	**Frequency** Annually (Spring)	**Forms of Learning and Teaching** • Lecture attendance (17.5 hours) • Tutorial attendance (17.5 hours) • Independent study (50 hours) • Exam preparation (20 hours)	
		Duration 1 semester	**Workload** 125 hours	

Recommendations for Preparation

Students should have a solid understanding of at least one programming language and its syntax before taking this module. They should review basic data structures and algorithms, as well as math skills, formal languages and automata, functional programming.

Content and Educational Aims

This module provides students with a deep understanding of the principles and techniques used in compilers, including lexical analysis, syntax analysis, semantic analysis, and code generation, give them hands-on experience with the implementation of a compiler using a modern compiler construction toolkit, teach students about the important trade-offs involved in the design and implementation of a compiler, including performance, code size, and maintainability.

Content:

- Introduction to compilers and the compilation process. Programming languages and machine architectures. Compiler, interpreter.
- Syntax analysis. Stack machine.
- x86 command system. Code generator, control structures, procedures and functions.
- Dynamic data structures and symbolic expressions.
- Program Runtime

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. know the basic algorithms for parsing code. Implements these algorithms. Know how to implement a compiler stack machine.
2. know the basic principles of dynamic data structures. Know the disciplines of dynamic memory management.

3. know the basic principles of compiler structure. Be able to organize and conduct a scientific discussion on issues from the professional sphere. Know how to discuss a choice of the optimal compiler for solving practical tasks.
4. know basic stages of compiler development. Implement all compiler components.

Indicative Literature

"Compilers: Principles, Techniques, and Tools" by Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman, commonly known as the "Dragon book" published by Addison-Wesley Professional (1986)

"Modern Compiler Implementation in Java" by Andrew W. Appel, published by Cambridge University Press (1998)

"Principles of Compiler Design" by Alfred V. Aho and Jeffrey D. Ullman, published by Addison-Wesley Professional (1977)

"Engineering a Compiler" by Keith D. Cooper and Linda Torczon, published by Morgan Kaufmann (2012)

"Compiling with Continuations" by Andrew W. Appel, published by Cambridge University Press (1992)

gcc.gnu.org/wiki/ListOfCompilerBooks - a list of books on compiler construction

gcc.gnu.org - compiler GCC;

llvm.org - infrastructure LLVM

Usability and Relationship to other Modules

- This module is suitable for computer science students with a solid understanding of programming and algorithms, as well as an interest in the inner workings of programming languages and the software development process. Knowledge of the module may be used to develop and improve the compiler, to write code generators for specific processors, to design and implement new languages, to improve code optimization, and to design and implement run-time systems.

Examination Type: Module Component Examination

Component 1: Lecture

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

Scope: All theoretical intended learning outcomes of the module

Component 2: Project

Assessment: Practical assessment (Programming assignments) — Weight: 50%

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%.

7.27 Semantics of Programming Languages

Module Name Semantics of Programming Languages			Module Code SDT-308	Level (type) Year 3 (Specialization)	CP 5.0
Module Components					
Number	Name			Type	CP
SDT-308-A	Semantics of Programming Languages			Lectures	2.5
SDT-308-B	Semantics of Programming Languages Tutorial			Tutorial	2.5
Module Coordinator Prof. Dr. Anton Podkopaev	**Program Affiliation** • Software, Data and Technology (SDT)			**Mandatory Status** Mandatory elective for SDT	
Entry Requirements Pre-requisites ☒ Formal languages and Parsers ☒ Functional Programming	Co-requisites ☒ none	Knowledge, Abilities, or Skills Understanding of propositional logic	**Frequency** Annually (Spring)	**Forms of Learning and Teaching** • Lecture attendance (17.5 hours) • Tutorial attendance (17.5 hours) • Independent study (50 hours) • Exam preparation (20 hours)	
			Duration 1 semester	**Workload** 125 hours	

Recommendations for Preparation

To master the module students need the knowledge obtained as a result of studying "Formal languages and Parsers" and "Functional programming".

Content and Educational Aims

The range of topics covered in this module includes approaches for formally describing semantics of a programming language as well as methods for proving the correctness of program transformations.

Content:

- Big-step and small-step operational semantics of imperative programming languages,
- Denotational semantics,
- Hoare's Axiomatic Semantics,
- Semantics of languages with multithreading.

Intended Learning Outcomes

Upon completion of this module, students will know:

1. the basic styles of programming language semantics: denotational, operational, axiomatic. Is able to reasonably describe the chosen style and the advantages of its use for a particular task.
2. the notion of semantic equivalence of programs and expressions. Knows variants of definitions and their relationships, justification of properties of program transformations.
3. the concept of operational semantics. Knows how to use big-step and small-step semantics.
4. the formalization of various semantics and proofs of their properties in Coq as well as verify programs using Hoare logic.

Indicative Literature
"Introduction to the Theory of Programming Languages" by Michel Parigot, published by Cambridge University Press (1992) "Semantics of Programming Languages" by Carl A. Gunter, published by MIT Press (1992) "Semantics Engineering with PLT Redex" by Matthew Flatt, Robert Bruce Findler, and Shriram Krishnamurthi, published by MIT Press (2013) "Programming Language Foundations" by Brian A. Malloy, published by Cambridge University Press (2018) "Semantics with Applications: An Appetizer" by Hanne Riis Nielson and Flemming Nielson, published by Springer (2007) Glynn Winskel. The Formal Semantics of Programming Languages; Benjamin Pierce et al. Software Foundations (Vol. 1, 2) https://softwarefoundations.cis.upenn.edu/ F.Nielson, H-R.Nielson. Semantics with Applications. A formal introduction.
Usability and Relationship to other Modules
• In terms of career prospects, knowledge of semantics of programming languages is relevant for anyone working in the field of programming languages, compilers, programming tools, programming languages design, formal verification, and many other areas. • This course is used for developing a deeper understanding of how programming languages are designed, implemented, and used, for gaining a solid foundation in formal semantics and type systems, which can be applied to programming languages and other formal systems.
Examination Type: Module Component Examination **Component 1: Lecture** Assessment: Written examination — Duration: 60 min, Weight: 50% Scope: All theoretical intended learning outcomes of the module **Component 2: Tutorial** Assessment: Practical assessment (Programming assignments) — Weight: 50% 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%.

7.28 Internship / Startup and Career Skills

Module Name	**Module Code**	**Level (type)**	**CP**
Internship / Startup and Career Skills	CA-INT-900	Year 3 (CAREER)	15

Module Components

Number	Name	Type	CP
CA-INT-900-0	Internship	Internship	15

Module Coordinator	**Program Affiliation**	**Mandatory Status**
Sinah Vogel & Dr. Tanja Woebs (CSC Organization); SPC / Faculty Startup Coordinator (Academic responsibility)	• CAREER module for undergraduate study programs	Mandatory for all undergraduate study programs except IEM

Entry Requirements

Pre-requisites	Co-requisites	Knowledge, Abilities, or Skills
☒ at least 15 CP from CORE modules in the major	☒ None	• Information provided on CSC pages (see below) • Major specific knowledge and skills

Frequency	**Forms of Learning and Teaching**
Annually (Spring/Fall)	• Internship/Start-up • Internship event • Seminars, info-sessions, workshops and career events • Self-study, readings, online tutorials

Duration	**Workload**
1 semester	375 Hours consisting of: • Internship (308 hours) • Workshops (33 hours) • Internship Event (2 hours) • Self-study (32 hours)

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://constructor.university/student-life/career-services
- Participating in the internship events of earlier classes

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 SCS, the internship may take place at other 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 Student Career Support 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 Student Career Support.

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 Service Center (e.g. the annual Constructor 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 will 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, workspace, 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 • 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%

7.29 Bachelor Thesis and Seminar in SDT

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

Bachelor Thesis and Seminar SDT</td><td>Module Code

SDT-400</td><td>Level (type)

Year 3
(CAREER)</td><td>CP

15</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>SDT-400-S</td><td colspan="3">Thesis SDT</td><td>Thesis</td><td>12</td></tr>
<tr><td>SDT-400-T</td><td colspan="3">Thesis Seminar SDT</td><td>Seminar</td><td>3</td></tr>
<tr><td>Module Coordinator

Study Program Chair</td><td colspan="3">Program Affiliation

• All undergraduate programs</td><td colspan="2">Mandatory Status

Mandatory for all undergraduate programs</td></tr>
<tr><td rowspan="2">Entry Requirements

Pre-requisites

☒ Students must have taken and successfully passed a total of at least 30 CP from advanced modules, and of those, at least 20 CP from advanced modules in the major.</td><td rowspan="2">Co-requisites

☒ None</td><td rowspan="2">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
(Spring)</td><td colspan="2">Forms of Learning and Teaching

• Self-study/lab work (350 hours)
• Seminars (25 hours)</td></tr>
<tr><td>Duration

14-week lecture period</td><td colspan="2">Workload

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

• Identify an area or a topic of interest and discuss this with your prospective supervisor in a timely manner.
• 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 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 address a problem from their respective major subject independently using academic/scientific methods within a set time frame. Although supervised, this module requires students to be able to work independently and systematically and set their own goals in exchange for the opportunity to explore a topic that excites and interests them personally and that a faculty member is interested in supervising. Within this module, students apply their acquired knowledge about their major discipline and their learned skills and methods for conducting 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 research results.

This module consists of two components, an independent thesis and an accompanying seminar. The thesis component must be supervised by a Constructor 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 a conclusion. The seminar provides students with the opportunity to practice their ability to present, discuss, and justify their and other students' approaches, methods, and results at various stages of their research in order to improve their academic writing, receive and reflect on formative feedback, and therefore grow 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 factors;
5. apply their knowledge and understanding to a context of their choice;
6. develop, formulate, and advance solutions to problems and debates within 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 in the undergraduate program. Students apply the knowledge, skills, and competencies they have acquired and practiced during their studies, including research methods and their ability to acquire additional skills independently as and if required.

Examination Type: Module Component Examinations

Module Component 1: Thesis
Assessment type: Thesis
Scope: All intended learning outcomes, mainly 1-6.

Length: approx. 6.000 – 8.000 words (15 – 25 pages), excluding front and back matter.
Weight: 80%

Module Component 2: Seminar

Assessment type: Presentation

Duration: approx. 15 to 30 minutes
Weight: 20%

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

Completion: To pass this module, both module component examinations have to be passed with at least 45%.

Two separate assessments are justified by the size of this module and the fact that the justification of solutions to problems and arguments (ILO 6) and discussion (ILO 7) should at least have verbal elements. The weights of the types of assessments are commensurate with the sizes of the respective module components.

8 Constructor Track Modules

8.1 Methods

8.1.1 Matrix Algebra and Advanced Calculus I

Module Name	**Module Code**	**Level (type)**	**CP**
Matrix Algebra and Advanced Calculus I	CTMS-MAT-22	Year 1 (Methods)	5

Module Components

Number	Name	Type	CP
CTMS-22	Matrix Algebra and Advanced Calculus I	Lecture	5

Module Coordinator	**Program Affiliation**	**Mandatory Status**
Dr. Keivan Mallahi Karai	• CONSTRUCTOR Track Area	Mandatory for ECE, MMDA, PHDS and SDT Mandatory elective for CS and RIS

Entry Requirements

Pre-requisites	Co-requisites	Knowledge, Abilities, or Skills
☒ none	☒ none	Knowledge of pre-calculus ideas (sets and functions, elementary functions, polynomials) and analytic geometry (equations of lines, systems of linear equations, dot product, polar coordinates) at High School level. Familiarity with ideas of calculus is helpful.

Frequency	**Forms of Learning and Teaching**
Annually (Fall)	• Lectures (35 hours) • Private study (90 hours)

Duration	**Workload**
1 semester	125 hours

Recommendations for Preparation

Review of high school mathematics.

Content and Educational Aims

This module is the first in a sequence including advanced mathematical methods at the university level at a level higher than the module Calculus and Linear Algebra I. The course comprises the following topics:

- Number systems, complex numbers
- The concept of function, composition of functions, inverse functions
- Basic ideas of calculus: Archimedes to Newton
- The notion of limit for functions and sequences and series
- Continuous function and their basic properties
- Derivatives: rate of change, velocity and applications
- Mean value theorem and estimation, maxima and minima, convex functions
- Integration, change of variables, Fundamental Theorem of Calculus
- Applications of the integral: work, area, average value, centre of mass
- Improper Integrals, Mean value theorem for integrals
- Taylor series
- Ordinary differential equations, examples, solving first order linear differential equations
- Basic ideas of numerical analysis, Newton's method, asymptotic formulas
- Review of elementary analytic geometry, lines, conics
- Vector spaces, linear independence, bases, coordinates

- Linear maps, matrices and their algebra, matrix inverses
- Gaussian elimination, solution space
- Determinants

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. apply the methods described in the content section of this module description to the extent that they can
2. solve standard text-book problems reliably and with confidence;
3. recognize the mathematical structures in an unfamiliar context and translate them into a mathematical problem statement;
4. 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

Advanced Calculus, G.B. Folland (Pearson, 2002)

Linear Algebra, S. Lang (Springer Verlag, 1986)

Mathematical Methods for Physics and Engineering,

K. Riley, M. Hobson, S. Bence (Cambridge University Press, 2006)

Usability and Relationship to other Modules

- Calculus and Linear Algebra I can be substituted with this module after consulting academic advisor
- 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.

Examination Type: Module Examination

Assessment Type: Written examination

Duration: 120 min
Weight: 100%

Scope: All intended learning outcomes of the module.

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

8.1.2 Matrix Algebra and Advanced Calculus II

<table>
<tr><td colspan="3">Module Name
Matrix Algebra and Advanced Calculus II</td><td>Module Code
CTMS-MAT-23</td><td>Level (type)
Year 1
(Methods)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTMS-23</td><td colspan="3">Matrix Algebra and Advanced Calculus II</td><td>Lecture</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Keivan Mallahi Karai</td><td colspan="3">Program Affiliation
CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

Mandatory for ECE, MMDA, PHDS and SDT

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

Pre-requisites: ☒ Matrix Algebra and Advanced Calculus I
Co-requisites: ☒ none
Knowledge, Abilities, or Skills:
None beyond formal pre-requisites</td><td>Frequency

Annually
(Spring)</td><td colspan="2">Forms of Learning and Teaching
Lectures (35 hours)
Private study (90 hours)</td></tr>
<tr><td>Duration

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

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

Review the content of Matrix Algebra and Advanced Calculus I</td></tr>
</table>

Content and Educational Aims

- Coordinate systems, functions of several variables, level curves, polar coordinates
- Continuity, directional derivatives, partial derivatives, chain rule (version I)
- derivative as a matrix, chain rule (version II), tangent planes and linear approximation, gradient, repeated partial derivatives
- Minima and Maxima of functions of several variables, Lagrange multipliers
- Multiple integrals, iterated integrals, integration over standard regions, change of variables formula
- Vector fields, parametric representation of curves, line integrals and arc length, conservative vector fields
- Potentials, Green's theorem in the plane
- Parametric representation of surfaces
- Vector products and normal surface integrals
- Integral theorems by Stokes and Gauss, physical interpretations
- Basics of differential forms and their calculus, connection to gradient, curl, and divergence
- Eigenvalues and eigenvectors, diagonalisable matrices
- Inner product spaces, Hermitian and unitary matrices
- Matrix factorizations: Singular value decomposition with applications, LU decomposition, QR decomposition
- Linear constant-coefficient ordinary differential equations, application to mechanical vibrations and electrical oscillations
- Periodic functions, Fourier series

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. understand the definitions of continuity, derivative of a function as a linear transformation, multivariable integrals, eigenvalues and eigenvectors and associated notions.
2. apply the methods described in the content section of this module description to the extent that they can
3. evaluate multivariable integrals using definitions or by applying Green and Stokes theorem.
4. evaluate various decompositions of matrices
5. solve standard text-book problems reliably and with confidence;

6. recognize the mathematical structures in an unfamiliar context and translate them into a mathematical problem statement;
7. 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

Advanced Calculus, G.B. Folland (Pearson, 2002)

Linear Algebra, S. Lang (Springer Verlag, 1986)

Mathematical Methods for Physics and Engineering,

K. Riley, M. Hobson, S. Bence (Cambridge University Press, 2006)

Vector Calculus, Linear Algebra, and Differential Forms: A Unified

Approach, J.H. Hubbard, B. Hubbard (Pearson, 1998)

Usability and Relationship to other Modules

- This module can substitute Calculus and Linear Algebra II after consulting academic advisor.
- Methods of this course are applied in the module Mathematical Modeling.
- The second-semester module Linear Algebra provides a more rigorous and more abstract treatment of some of the notions discussed in this module.

Examination Type: Module Examination

Assessment Component: Written examination Length/duration: (120min)

Weight: 100 %

Scope: All intended learning outcomes of this module

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

8.1.3 Probability and Random Processes

<table>
<tr><td colspan="3">Module Name
Probability and Random Processes</td><td>Module Code
CTMS-MAT-12</td><td>Level (type)
Year 2
(Methods)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTMS-12</td><td colspan="3">Probability and random processes</td><td>Lecture</td><td>5</td></tr>
<tr><td>Module Coordinator

Dr. Keivan Mallahi Karai</td><td colspan="3">Program Affiliation

• CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

Mandatory for CS, ECE, MMDA, PHDS, RIS and SDT</td></tr>
<tr><td rowspan="2">Entry Requirements

Pre-requisites

☒ Matrix Algebra and Advanced Calculus II or Calculus and Linear Algebra II</td><td rowspan="2">Co-requisites

☒ None</td><td rowspan="2">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)</td><td colspan="2">Forms of Learning and Teaching

Lectures (35 hours)
Private study (90 hours)</td></tr>
<tr><td>Duration

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

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

- 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

- Students taking this module are expected to be familiar with basic tools from calculus and linear algebra.

Examination Type: Module Examination

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

Scope: All intended learning outcomes of this module

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

8.1.4 Statistics and Data Analytics

Module Name Statistics and Data Analytics			**Module Code** CTMS-MET-21	**Level (type)** Year 2 (Methods)	**CP** 5
Module Components Number					
CTMS-21	Statistics and Data Analytics			Lecture	5
Module Coordinator Dr. Ivan Ovsyannikov	**Program Affiliation** • CONSTRUCTOR Track Area			**Mandatory Status** Mandatory for MMDA, PHDS and SDT	
Entry Requirements Pre-requisites ☒ Probability and Random Processes	Co-requisites ☒ none	Knowledge, Abilities, or Skills Good command of basic probability	**Frequency** Annually (Spring)	**Forms of Learning and Teaching** • Lectures (35 hours) • Private Study (105 hours)	
			Duration 1 semester	**Workload** 120 hours	
Recommendations for Preparation Recap Probability and Random Processes					

Content and Educational Aims

The aims of this module is to introduce students to basic ideas and methods used for analysing large and complex datasets. While the first modern statistical toolkits date back to the beginning of the twentieth century, the advent of computer age and the availability of fast computations has lead to dramatic changes in the field.

Statistical models have found applications in many areas ranging from business and healthcare to astrophysics and speech recognition. Such models are used to make predictions, draw inferences and support policy decisions in all these areas.

This module draws on students' knowledge from the module Probability and Random Processes to help them build and analyze statistical models, ranging in their degree of sophistication from basis to more advanced ones, and apply them to real-world situations. The module will cover the following topics:

- Classical statistics: descriptive and inferential modes, parameter estimation and hypothesis testing.
- Linear regressions, multiple linear regressions
- Classification: logistic regression, generative models for classification
- Resampling methods, bootstrap
- Non-linear models, splines
- Support vector machines
- Basic ideas of deep learning

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. formulate statistical models for real world problems
2. describe statistical methods for analyzing real world problems
3. explain the importance of linear and non-linear models

4. recognize different solution methods for modeling problems
5. illustrate the use of regressions, resampling, support vector machines and other statistical tools to describe phenomena in the real world
6. Describe basic ideas of deep learning

Indicative Literature

James, Witten, Hastie, Tibshirani. An introduction to Statistical learning; second edition.

Usability and Relationship to other Modules

- This module is part of the core education in Mathematics, Modeling and Data Analytics and Physics and Data Science.
- It is also valuable for students in Computer Science, RIS, and ECE, either as part of a minor in Mathematics, or as an elective module.

Examination Type: Module Examination

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

Scope: All intended learning outcomes of this module

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

8.2 New Skills

8.2.1 Logic (perspective I)

<table>
<tr><td colspan="2">Module Name
Logic (perspective I)</td><td>Module Code
CTNS-NSK-01</td><td>Level (type)
Year 2
(New Skills)</td><td>CP
2.5</td></tr>
<tr><td colspan="5">Module Components</td></tr>
<tr><td>Number</td><td colspan="2">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTNS-01</td><td colspan="2">Logic (perspective I)</td><td>Lecture (online)</td><td>2.5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Jules Coleman</td><td colspan="2">Program Affiliation

• CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

Mandatory elective for all UG students (one perspective must be chosen)</td></tr>
<tr><td colspan="2" rowspan="2">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

Online lecture (17.5h)
Private study (45h)</td></tr>
<tr><td>Duration

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

62.5 hours</td></tr>
<tr><td colspan="5">Recommendations for Preparation</td></tr>
<tr><td colspan="5">Content and Educational Aims

Suppose a friend asks you to help solve a complicated problem? Where do you begin? Arguably, the first and most difficult task you face is to figure out what the heart of the problem actually is. In doing that you will look for structural similarities between the problem posed and other problems that arise in different fields that others may have addressed successfully. Those similarities may point you to a pathway for resolving the problem you have been asked to solve. But it is not enough to look for structural similarities. Sometimes relying on similarities may even be misleading. Once you've settled tentatively on what you take to be the heart of the matter, you will naturally look for materials, whether evidence or arguments, that you believe is relevant to its potential solution. But the evidence you investigate of course depends on your formulation of the problem, and your formulation of the problem likely depends on the tools you have available – including potential sources of evidence and argumentation. You cannot ignore this interactivity, but you can't allow yourself to be hamstrung entirely by it. But there is more. The problem itself may be too big to be manageable all at once, so you will have to explore whether it can be broken into manageable parts and if the information you have bears on all or only some of those parts. And later you will face the problem of whether the solutions to the particular sub problems can be put together coherently to solve the entire problem taken as a whole.

What you are doing is what we call engaging in computational thinking. There are several elements of computational thinking illustrated above. These include: Decomposition (breaking the larger problem down into smaller ones); Pattern recognition (identifying structural similarities); Abstraction (ignoring irrelevant particulars of the problem): and Creating Algorithms), problem-solving formulas.

But even more basic to what you are doing is the process of drawing inferences from the material you have. After all, how else are you going to create a problem-solving formula, if you draw incorrect inferences about what information has shown and what, if anything follows logically from it. What you must do is apply the rules of logic to the information to draw inferences that are warranted.</td></tr>
</table>

We distinguish between informal and formal systems of logic, both of which are designed to indicate fallacies as well as warranted inferences. If I argue for a conclusion by appealing to my physical ability to coerce you, I prove nothing about the truth of what I claim. If anything, by doing so I display my lack of confidence in my argument. Or if the best I can do is berate you for your skepticism, I have done little more than offer an ad hominem instead of an argument. Our focus will be on formal systems of logic, since they are at the heart of both scientific argumentation and computer developed algorithms. There are in fact many different kinds of logic and all figure to varying degrees in scientific inquiry. There are inductive types of logic, which purport to formalize the relationship between premises that if true offer evidence on behalf of a conclusion and the conclusion and are represented as claims about the extent to which the conclusion is confirmed by the premises. There are deductive types of logic, which introduce a different relationship between premise and conclusion. These variations of logic consist in rules that if followed entail that if the premises are true then the conclusion too must be true.

There are also modal types of logic which are applied specifically to the concepts of necessity and possibility, and thus to the relationship among sentences that include either or both those terms. And there is also what are called deontic logic, a modification of logic that purport to show that there are rules of inference that allow us to infer what we ought to do from facts about the circumstances in which we find ourselves. In the natural and social sciences most of the emphasis has been placed on inductive logic, whereas in math it is placed on deductive logic, and in modern physics there is an increasing interest in the concepts of possibility and necessity and thus in modal logic. The humanities, especially normative discussions in philosophy and literature are the province of deontic logic.

This module will also take students through the central aspects of computational thinking, as it is related to logic; it will introduce the central concepts in each, their relationship to one another and begin to provide the conceptual apparatus and practical skills for scientific inquiry and research.

Intended Learning Outcomes

Students acquire transferable and key skills in this module.

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

1. apply the various principles of logic and expand them to computational thinking.
2. understand the way in which logical processes in humans and in computers are similar and different at the same time.
3. apply the basic rules of first-order deductive logic and employ them rules in the context of creating a scientific or social scientific study and argument.
4. employ those rules in the context of creating a scientific or social scientific study and argument

Indicative Literature

Frege, Gottlob (1879), Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens [Translation: A Formal Language for Pure Thought Modeled on that of Arithmetic],Halle an der Salle: Verlag von Louis Nebert.

Gödel, Kurt (1986), Russels mathematische Logik. In: Alfred North Whitehead, Bertrand Russell: Principia Mathematica. Vorwort, S. V–XXXIV. Suhrkamp.

Leeds, Stephen. "George Boolos and Richard Jeffrey. Computability and logic. Cambridge University Press, New York and London1974, x+ 262 pp." The Journal of Symbolic Logic 42.4 (1977): 585-586.

Kubica, Jeremy. Computational fairy tales. Jeremy Kubica, 2012.

McCarthy, Timothy. "Richard Jeffrey. Formal logic: Its scope and limits. of XXXVIII 646. McGraw-Hill Book Company, New York etc. 1981, xvi+ 198 pp." The Journal of Symbolic Logic 49.4 (1984): 1408-1409.

Usability and Relationship to other Modules

Examination Type: Module Examination

Assessment Type: Written Examination

Duration/Length: 60 min
Weight: 100%

Scope: All intended learning outcomes of the module.

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

8.2.2 Logic (perspective II)

Module Name Logic (perspective II)	**Module Code** CTNS-NSK-02	**Level (type)** Year 2 (New Skills)	**CP** 2.5

Module Components

Number	Name	Type	CP
CTNS-02	Logic (perspective II)	Lecture (online)	2.5

Module Coordinator **NN**	**Program Affiliation** • CONSTRUCTOR Track Area	**Mandatory Status** Mandatory elective for all UG students (one perspective must be chosen)

Entry Requirements			**Frequency**	**Forms of Learning and Teaching**
Pre-requisites	Co-requisites	Knowledge, Abilities, or Skills	Annually (Fall)	Online lecture (17.5h) Private study (45h)
☒ none	☒ none			
			Duration 1 semester	**Workload** 62.5 hours

Recommendations for Preparation

Content and Educational Aims

The focus of this module is on formal systems of logic, since they are at the heart of both scientific argumentation and computer developed algorithms. There are in fact many kinds of logic and all figure to varying degrees in scientific inquiry. There are inductive types of logic, which purport to formalize the relationship between premises that if true offer evidence on behalf of a conclusion and the conclusion and are represented as claims about the extent to which the conclusion is confirmed by the premises. There are deductive types of logic, which introduce a different relationship between premise and conclusion. These variations of logic consist in rules that if followed entail that if the premises are true then the conclusion too must be true.

This module introduces logics that go beyond traditional deductive propositional logic and predicate logic and as such it is aimed at students who are already familiar with basics of traditional formal logic. The aim of the module is to provide an overview of alternative logics and to develop a sensitivity that there are many different logics that can provide effective tools for solving problems in specific application domains.

The module first reviews the principles of a traditional logic and then introduces many-valued logics that distinguish more than two truth values, for example true, false, and unknown. Fuzzy logic extends traditional logic by replacing truth values with real numbers in the range 0 to 1 that are expressing how strong the believe into a proposition is. Modal logics introduce modal operators expressing whether a proposition is necessary or possible. Temporal logics deal with propositions that are qualified by time. Once can view temporal logics as a form of modal logics where propositions are qualified by time constraints. Interval temporal logic provides a way to reason about time intervals in which propositions are true.

The module will also investigate the application of logic frameworks to specific classes of problems. For example, a special subset of predicate logic, based on so-called Horn clauses, forms the basis of logic programming languages such as Prolog. Description logics, which are usually decidable logics, are used to model relationships and they have applications in the semantic web, which enables search engines to reason about resources present on the Internet.

Intended Learning Outcomes

Students acquire transferable and key skills in this module.

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

1. apply the various principles of logic
2. explain practical relevance of non-standard logic
3. describe how many-valued logic extends basic predicate logic
4. apply basic rules of fuzzy logic to calculate partial truth values
5. sketch basic rules of temporal logic
6. implement predicates in a logic programming language
7. prove some simple non-standard logic theorems

Indicative Literature

Bergmann, Merry. “An Introduction to Many-Valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems”, Cambridge University Press, April 2008.

Sterling, Leon S., Ehud Y. Shapiro, Ehud Y. "The Art of Prolog", 2nd edition, MIT Press, March 1994.

Fisher, Michael. “An Introduction to Practical Formal Methods Using Temporal Logic”, Wiley, Juli 2011.

Baader, Franz. "The Description Logic Handbook: Theory Implementation and Applications", Cambridge University Press, 2nd edition, May 2010.

Usability and Relationship to other Modules

Examination Type: Module Examination

Assessment Type: Written Examination

Duration/Length: 60 min
Weight: 100%

Scope: All intended learning outcomes of the module.

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

8.2.3 Causation and Correlation (perspective I)

<table>
<tr><td colspan="3">Module Name
Causation and Correlation (perspective I)</td><td>Module Code
CTNS-NSK-03</td><td>Level (type)
Year 2
(New Skills)</td><td>CP
2.5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTNS-03</td><td colspan="3">Causation and Correlation</td><td>Lecture (online)</td><td>2.5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Jules Coleman</td><td colspan="3">Program Affiliation

• CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

Mandatory elective for all UG students (one perspective must be chosen)</td></tr>
<tr><td colspan="3" rowspan="2">Entry Requirements

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

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

Online lecture (17.5h)
Private study (45h)</td></tr>
<tr><td>Duration

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

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

Content and Educational Aims

In many ways, life is a journey. And also, as in other journeys, our success or failure depends not only on our personal traits and character, our physical and mental health, but also on the accuracy of our map. We need to know what the world we are navigating is actually like, the how, why and the what of what makes it work the way it does. The natural sciences provide the most important tool we have developed to learn how the world works and why it works the way it does. The social sciences provide the most advanced tools we have to learn how we and other human beings, similar in most ways, different in many others, act and react and what makes them do what they do. In order for our maps to be useful, they must be accurate and correctly reflect the way the natural and social worlds work and why they work as they do.

The natural sciences and social sciences are blessed with enormous amounts of data. In this way, 99istoryy and the present are gifts to us. To understand how and why the world works the way it does requires that we are able to offer an explanation of it. The data supports a number of possible explanations of it. How are we to choose among potential explanations? Explanations, if sound, will enable us to make reliable predictions about what the future will be like, and also to identify many possibilities that may unfold in the future. But there are differences not just in the degree of confidence we have in our predictions, but in whether some of them are necessary future states or whether all of them are merely possibilities? Thus, there are three related activities at the core of scientific inquiry: understanding where we are now and how we got here (historical); knowing what to expect going forward (prediction); and exploring how we can change the paths we are on (creativity).

At the heart of these activities are certain fundamental concepts, all of which are related to the scientific quest to uncover immutable and unchanging laws of nature. Laws of nature are thought to reflect a causal nexus between a previous event and a future one. There are also true statements that reflect universal or nearly universal connections between events past and present that are not laws of nature because the relationship they express is that of a correlation between events. A working thermostat accurately allows us to determine or even to predict the temperature in the room in which it is located, but it does not explain why the room has the temperature it has. What then is the core difference between causal relationships and correlations? At the same time, we all recognize that given where we are now there are many possible futures for each of us, and even had our lives gone just the slightest bit differently than they have, our present state could well have been very different than it is. The relationship between possible pathways between events that have not materialized but could have is expressed through the idea of counterfactual.

Creating accurate roadmaps, forming expectations we can rely on, making the world a more verdant and attractive place requires us to understand the concepts of causation, correlation, counterfactual explanation, prediction, necessity, possibility, law of nature and universal generalization. This course is designed precisely to provide the conceptual tools and intellectual skills to implement those concepts in our future readings and research and ultimately in our experimental investigations, and to employ those tools in various disciplines.

Intended Learning Outcomes

Students acquire transferable and key skills in this module.

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

1. formulate testable hypotheses that are designed to reveal causal connections and those designed to reveal interesting, important and useful correlations.
2. distinguish scientifically interesting correlations from unimportant ones.
3. apply critical thinking skills to evaluate information.
4. understand when and why inquiry into unrealized possibility is important and relevant.

Indicative Literature

Thomas S. Kuhn: The Structure of Scientific Revolutions, Nelson, fourth edition 2012;

Goodman, Nelson. Fact, fiction, and forecast. Harvard University Press, 1983;

Quine, Willard Van Orman, and Joseph Silbert Ullian. The web of belief. Vol. 2. New York: Random house, 1978.

Usability and Relationship to other Modules

Examination Type: Module Examination

Assessment Type: Written Examination

Duration/Length: 60 min
Weight: 100%

Scope: All intended learning outcomes of the module

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

8.2.4 Causation and Correlation (perspective II)

Module Name			Module Code	Level (type)	CP
Causation and Correlation (perspective II)			CTNS-NSK-04	Year 2 (New Skills)	2.5
Module Components					
Number	Name			Type	CP
CTNS-04	Causation and Correlations (perspective II)			Lecture (online)	2.5
Module Coordinator Dr. Keivan Mallahi-Karai Dr. Eoin Ryan Dr. Irina Chiaburu	**Program Affiliation** • CONSTRUCTOR Track Area			**Mandatory Status** Mandatory elective for all UG students (one perspective must be chosen)	
Entry Requirements Pre-requisites ☒ none	Co-requisites ☒ none	Knowledge, Abilities, or Skills Basic probability theory	**Frequency** Annually (Spring) **Duration** 1 semester	**Forms of Learning and Teaching** Online lecture (17.5h) Private study (45h **Workload** 62.5 hours	
Recommendations for Preparation					

Content and Educational Aims

Causality or causation is a surprisingly difficult concept to understand. David Hume famously noted that causality is a concept that our science and philosophy cannot do without, but it is equally a concept that our science and philosophy cannot describe. Since Hume, the problem of cause has not gone away, and sometimes seems to get even worse (e.g., quantum mechanics confusing previous notions of causality). Yet, ways of doing science that lessen our need to explicitly use causality have become very effective (e.g., huge developments in statistics). Nevertheless, it still seems that the concept of causality is at the core of explaining how the world works, across fields as diverse as physics, medicine, logistics, the law, sociology, and history – and ordinary daily life – through all of which, explanations and predictions in terms of cause and effect remain intuitively central.

Causality remains a thorny problem but, in recent decades, significant progress has occurred, particularly in work by or inspired by Judea Pearl. This work incorporates many 20th century developments, including statistical methods – but with a reemphasis on finding the why, or the cause, behind statistical correlations –, progress in understanding the logic, semantics and metaphysics of conditionals and counterfactuals, developments based on insights from the likes of philosopher Hans Reichenbach or biological statistician Sewall Wright into causal precedence and path analysis, and much more. The result is a new toolkit to identify causes and build causal explanations. Yet even as we get better at identifying causes, this raises new (or old) questions about causality, including metaphysical questions about the nature of causes (and effects, events, objects, etc), but also questions about what we really use causality for (understanding the world as it is or just to glean predictive control of specific outcomes), about how causality is used differently in different fields and activities (is cause in physics the same as that in history?), and about how other crucial concepts relate to our concept of cause (space and time seem to be related to causality, but so do concepts of legal and moral responsibility).

This course will introduce students to the mathematical formalism derived from Pearl's work, based on directed acyclic graphs and probability theory. Building upon previous work by Reichenbach and Wright, Pearl defines a "a calculus of interventions" of "do-calculus" for talking about interventions and their relation to causation and counterfactuals. This model has been applied in various areas ranging from econometrics to statistics, where acquiring knowledge about causality is of great importance.

At the same time, the course will not forget some of the metaphysical and epistemological issues around cause, so that students can better critically evaluate putative causal explanations in their full context. Abstractly, such issues involve some of the same philosophical questions Hume already asked, but more practically, it is important to see how metaphysical and epistemological debates surrounding the notion of cause affect scientific practice, and equally if not more importantly, how scientific practice pushes the limits of theory. This course will look at various ways in which empirical data can be transformed into explanations and theories, including the variance approach to causality (characteristic of the positivistic quantitative paradigm), and the process theory of causality (associated with qualitative methodology). Examples and case studies will be relevant for students of the social sciences but also students of the natural/physical world as well.

Intended Learning Outcomes

Students acquire transferable and key skills in this module.

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

1. have a clear understanding of the history of causal thinking.
2. be able to form a critical understanding of the key debates and controversies surrounding the idea of causality.
3. be able to recognize and apply probabilistic causal models.
4. be able to explain how understanding of causality differs among different disciplines.
5. be able demonstrate how theoretical thinking about causality has shaped scientific practices.

Indicative Literature

Paul, L. A. and Ned Hall. Causation: A User's Guide. Oxford University Press 2013.

Pearl, Judea. Causality: Models, Reasoning and Inference. Cambridge University Press 2009

Pearl, Judea, Glymour Madelyn and Jewell, Nicolas. Causal Inference in Statistics: A Primer. Wiley 2016

Ilari, Phyllis McKay and Federica Russo. Causality: Philosophical Theory Meets Scientific Practice. Oxford University Press 2014.

Usability and Relationship to other Modules

Examination Type: Module Examination

Assessment: Written examination Duration/Length: 60 min

Weight: 100 %

Scope: All intended learning outcomes of the module

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

8.2.5 Linear Model and Matrices

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

Linear Model and Matrices</td><td>Module Code

CTNS-NSK-05</td><td>Level (type)

Year 3
(New Skills)</td><td>CP

5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTNS-05</td><td colspan="3">Linear Model and Matrices</td><td>Seminar
(online)</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Marc-Thorsten Hütt</td><td colspan="3">Program Affiliation
CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

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

Pre-requisites: Logic; Causation & Correlation
Co-requisites: ☒ none
Knowledge, Abilities, or Skills</td><td>Frequency

Annually

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

Online lecture (35h)

Private Study (90h)</td></tr>
<tr><td>Duration

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

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

Content and Educational Aims

There are no universal 'right skills'. But the notion of linear models and the avenue to matrices and their properties can be useful in diverse disciplines to implement a quantitative, computational approach. Some of the most popular data and systems analysis strategies are built upon this framework. Examples include principal component analysis (PCA), the optimization techniques used in Operations Research (OR), the assessment of stable and unstable states in nonlinear dynamical systems, as well as aspects of machine learning.

Here we introduce the toolbox of linear models and matrix-based methods embedded in a wide range of transdisciplinary applications (part 1). We describe its foundation in linear algebra (part 2) and the range of tools and methods derived from this conceptual framework (part 3). At the end of the course, we outline applications to graph theory and machine learning (part 4). Matrices can be useful representations of networks and of system of linear equations. They are also the core object of linear stability analysis, an approach used in nonlinear dynamics. Throughout the course, examples from neuroscience, social sciences, medicine, biology, physics, chemistry, and other fields are used to illustrate these methods.

A strong emphasis of the course is on the sensible usage of linear approaches in a nonlinear world. We will critically reflect the advantages as well as the disadvantages and limitations of this method. Guiding questions are: How appropriate is a linear approximation of a nonlinear system? What do you really learn from PCA? How reliable are the optimal states obtained via linear programming (LP) techniques?

This debate is embedded in a broader context: How does the choice of a mathematical technique confine your view on the system at hand? How, on the other hand, does it increase your capabilities of analyzing the system (due to software available for this technique, the ability to compare with findings from other fields built upon the same technique and the volume of knowledge about this technique)?

In the end, students will have a clearer understanding of linear models and matrix approaches in their own discipline, but they will also see the full transdisciplinarity of this topic. They will make better decisions in their choice of data analysis methods and become mindful of the challenges when going from a linear to a nonlinear thinking.

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. apply the concept of linear modeling in their own discipline
2. distinguish between linear and nonlinear interpretation strategies and understand the range of applicability of linear models
3. make use of data analysis / data interpretation strategies from other disciplines, which are derived from linear algebra
4. be aware of the ties that linear models have to machine learning and network theory

   Note that these four ILOs can be loosely associated with the four parts of the course indicated above

Indicative Literature

Part 1:

material from Linear Algebra for Everyone, Gilbert Strang, Wellesley-Cambridge Press, 2020

Part 2:

material from Introduction to Linear Algebra (5th Edition), Gilbert Strang, Cambridge University Press, 2021

Part 3:

Mainzer, Klaus. "Introduction: from linear to nonlinear thinking." Thinking in Complexity: The Computational Dynamics of Matter, Mind and Mankind (2007): 1-16.

material from Mathematics of Big Data: Spreadsheets, Databases, Matrices, and Graphs, Jeremy Kepner, Hayden Jananthan, The MIT Press, 2018

material from Introduction to Linear Algebra (5th Edition), Gilbert Strang, Cambridge University Press, 2021

Part 4:

material from Linear Algebra and Learning from Data, Gilbert Strang, Wellesley-Cambridge Press, 2019

Usability and Relationship to other Modules

Examination Type: Module Examination

Assessment: Written examination

Duration/Length: 120 min
Weight: 100 %

Scope: All intended learning outcomes of the module

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

8.2.6 Complex Problem Solving

Module Name	Module Code	Level (type)	CP
Complex Problem Solving	CTNS-NSK-06	Year 3 (New Skills)	5

Module Components

Number	Name	Type	CP
CTNS-06	Complex Problem Solving	Lecture (online)	5

Module Coordinator	Program Affiliation	Mandatory Status
Prof. Dr. Marco Verweij	• CONSTRUCTOR Track Area	Mandatory elective

Entry Requirements			Frequency	Forms of Learning and Teaching
Pre-requisites ☒ Logic ☒ Causation and Correlation	Co-requisites ☒ none	Knowledge, Abilities, or Skills Being able to read primary academic literature Willingness to engage in teamwork	Annually (Fall)	Online Lectures (35h) Private Study (90h)
			Duration 1 semester	**Workload** 125 hours

Recommendations for Preparation

Please read: Camillus, J. (2008). Strategy as a wicked problem. Harvard Business Review 86: 99-106; Rogers, P. J. (2008). Using programme theory to evaluate complicated and complex aspects of interventions. Evaluation, 14, 29–48.

Content and Educational Aims

Complex problems are, by definition, non-linear and/or emergent. Some fifty years ago,scholars such as Herbert Simon began to argue that societies around the world had developed an impressive array of tools with which to s simple and even complicated problems, but still needed to develop methods with which to address the rapidly increasing number of complex issues. Since then, a variety of such methods has emerged. These include ‘seriou games’ developed in computer science, ‘multisector systems analysis’ applied in civil and environmental engineering, ‘robust decision-making’ proposed by the RAND Corporation, ‘design thinking’ developed in engineering and business studies, ‘structured problem solving’ used by McKinsey & Co., ‘real-time technology assessment’ advocated in science and technology studies, and ‘deliberative decision-making’ emanating from political science.

In this course, students first learn to distinguish between simple, complicated and complex problems. They also become familiar with the ways in which a particular issue can sometimes shift from one category into another. In addition, the participants learn to apply several tools for resolving complex problems. Finally, the students are introduced to the various ways in which natural and social scientists can help stakeholders resolve complex problems. Throughout the course examples and applications will be used. When possible, guest lectures will be offered by experts on a particular tool for tackling complex issues. For the written, take-home exam, students will have to select a specific complex problem, analyse it and come up with a recommendation – in addition to answering several questions about the material learned.

Intended Learning Outcomes Upon completion of this module, students will be able to: 1. Identify a complex problem; 2. Develop an acceptable recommendation for resolving complex problems. 3. Understand the roles that natural and social scientists can play in helping stakeholders resolve complex problems;
Indicative Literature Chia, A. (2019). Distilling the essence of the McKinsey way: The problem-solving cycle. Management Teaching Review 4(4): 350-377. Den Haan, J., van der Voort, M.C., Baart, F., Berends, K.D., van den Berg, M.C., Straatsma, M.W., Geenen, A.J.P., & Hulscher, S.J.M.H. (2020). The virtual river game: Gaming using models to collaboratively explore river management complexity, Environmental Modelling & Software 134, 104855, Folke, C., Carpenter, S., Elmqvist, T., Gunderson, L., Holling, C.S., & Walker, B. (2002). Resilience and sustainable development: Building adaptive capacity in a world of transformations. AMBIO: A Journal of the Human Environment 31(5): 437-440. Ostrom, E. (2010). Beyond markets and states: Polycentric governance of complex economic systems. American Economic Review 100(3): 641-72. Pielke, R. Jr. (2007). The honest broker: Making sense of science in policy and politics. Cambridge: Cambridge University Press. Project Management Institute (2021). A guide to the project management body of knowledge (PMBOK® guide). Schon, D. A., & Rein, M. (1994). Frame reflection: Toward the resolution of intractable policy controversies. New York: Basic Books. Simon, H. A. (1973). The structure of ill structured problems. Artificial Intelligence 4(3-4): 181-201. Verweij, M. & Thompson, M. (Eds.) (2006). Clumsy solutions for a complex world. London: Palgrave Macmillan.
Usability and Relationship to other Modules
Examination Type: Module Examination Assessment Type: Written examination — Duration: 120 min, Weight: 100% Scope: All intended learning outcomes of the module. Completion: To pass this module, the examination has to be passed with at least 45%.

8.2.7 Argumentation, Data Visualization and Communication (perspective I)

<table>
<tr><td colspan="3">Module Name
Argumentation, Data Visualization and Communication (perspective I)</td><td>Module Code
CTNS-NSK-07</td><td>Level (type)
Year 3
(New Skills)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTNS-07</td><td colspan="3">Argumentation, Data Visualization and Communication</td><td>Lecture (online)</td><td>5</td></tr>
<tr><td>Module Coordinator
Prof. Dr. Jules Coleman,
Prof. Dr. Arvid Kappas</td><td colspan="3">Program Affiliation
CONSTRUCTORr Track Area</td><td colspan="2">Mandatory Status
Mandatory elective for all UG students (one perspective must be chosen)</td></tr>
<tr><td colspan="3" rowspan="2">Entry Requirements
Pre-requisites: ☒ Logic
☒ Causation & Correlation
Co-requisites: ☒ none
Knowledge, Abilities, or Skills:</td><td>Frequency
Annually
(Fall)</td><td colspan="2">Forms of Learning and Teaching
Online Lectures (35h)
Private Study (90h)</td></tr>
<tr><td>Duration
1 semester</td><td colspan="2">Workload
125h</td></tr>
<tr><td colspan="6">Recommendations for Preparation</td></tr>
</table>

Content and Educational aim

One must be careful not to confuse argumentation with being argumentative. The latter is an unattractive personal attribute, whereas the former is a requirement of publicly holding a belief, asserting the truth of a proposition, the plausibility of a hypothesis, or a judgment of the value of a person or an asset. It is an essential component of public discourse. Public discourse is governed by norms and one of those norms is that those who assert the truth of a proposition or the validity of an argument or the responsibility of another for wrongdoing open themselves up to good faith requests to defend their claims. In its most general meaning, argumentation is the requirement that one offer evidence in support of the claims they make, as well as in defense of the judgments and assessments they reach. There are different modalities of argumentation associated with different contexts and disciplines. Legal arguments have a structure of their own as do assessments of medical conditions and moral character. In each case, there are differences in the kind of evidence that is thought relevant and, more importantly, in the standards of assessment for whether a case has been successfully made. Different modalities of argumentation require can call for different modes of reasoning. We not only offer reasons in defense of or in support of beliefs we have, judgments we make and hypotheses we offer, but we reason from evidence we collect to conclusions that are warranted by them.

Reasoning can be informal and sometimes even appear unstructured. When we recognize some reasoning as unstructured yet appropriate what we usually have in mind is that it is not linear. Most reasoning we are familiar with is linear in character. From A we infer B, and from A and B we infer C, which all together support our commitment to D. The same form of reasoning applies whether the evidence for A, B or C is direct or circumstantial. What changes in these cases is perhaps the weight we give to the evidence and thus the confidence we have in drawing inferences from it.

Especially in cases where reasoning can be supported by quantitative data, wherever quantitative data can be obtained either directly or by linear or nonlinear models, the visualization of the corresponding data can become key in both, reasoning and argumentation. A graphical representation can reduce the complexity of argumentation and is considered a must in effective scientific communication. Consequently, the course will also focus on smart and compelling ways for data visualization - in ways that go beyond what is typically taught in statistics or mathematics lectures. These tools are constantly developing, as a reflection of new software and changes in state of the presentation art. Which graph or bar chart to use best for which data, the use of colors to underline messages and arguments, but also the pitfalls when presenting data in a poor or even misleading manner. This will also help in readily identifying intentional mis-representation of data by others, the simplest to recognize being truncating the ordinate of a graph in order to exaggerate trends. This frequently leads to false arguments, which can then be readily countered.

There are other modalities of reasoning that are not linear however. Instead they are coherentist. We argue for the plausibility of a claim sometimes by showing that it fits in with a set of other claims for which we have independent support. The fit is itself the reason that is supposed to provide confidence or grounds for believing the contested claim.

Other times, the nature of reasoning involves establishing not just the fit but the mutual support individual items in the evidentiary set provide for one another. This is the familiar idea of a web of interconnected, mutually supportive beliefs. In some cases, the support is in all instances strong; in others it is uniformly weak, but the set is very large; in other cases, the support provided each bit of evidence for the other is mixed: sometimes strong, sometimes weak, and so on.

There are three fundamental ideas that we want to extract from this segment of the course. These are (1) that argumentation is itself a requirement of being a researcher who claims to have made findings of one sort or another; (2) that there are different forms of appropriate argumentation for different domains and circumstances; and (3) that there are different forms of reasoning on behalf of various claims or from various bits of evidence to conclusions: whether those conclusions are value judgments, political beliefs, or scientific conclusions. Our goal is to familiarize you with all three of these deep ideas and to help you gain facility with each.

Intended Learning Outcomes

Students acquire transferable and key skills in this module.

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

1. distinguish among different modalities of argument, e.g. legal arguments, vs. scientific ones.
2. construct arguments using tools of data visualization.
3. communicate conclusions and arguments concisely, clearly and convincingly.

Indicative Literature

- Tufte, E.R. (1985). The visual display of quantitative information. The Journal for Healthcare Quality (JHQ), 7(3), 15.
- Cairo, A (2012). The Functional Art: An introduction to information graphics and visualization. New Ridders.
- Knaflic, C.N. (2015). Storytelling with data: A data visualization guide for business professionals. John Wiley & Sons.

Usability and Relationship to other Modules

Examination Type: Module Examination

Assessment Type: Written Examination

Duration/Length: 120 (min)
Weight: 100%

Scope: All intended learning outcomes of the module

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

8.2.8 Argumentation, Data Visualization and Communication (perspective II)

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

Argumentation, Data Visualization and Communication (perspective II)</td><td>Module Code
CTNS-NSK-08</td><td>Level (type)
Year 3
(New Skills)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTNS-08</td><td colspan="3">Argumentation, Data Visualization and Communication</td><td>Lecture (online)</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Jules Coleman,
Prof. Dr. Arvid Kappas</td><td colspan="3">Program Affiliation

• CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

Mandatory elective for all UG students (one perspective must be chosen)</td></tr>
<tr><td colspan="3" rowspan="2">Entry Requirements

Pre-requisites: ☒ Logic; ☒ Causation & Correlation

Co-requisites: ☒ none

Knowledge, Abilities, or Skills: ability and openness to engage in interactions media literacy, critical thinking and a proficient handling of data sources own research in academic literature</td><td>Frequency

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

• Online Lecture (35 hours)
• Tutorial of the lecture (10 hours)
• Private study for the lecture (80 hours)</td></tr>
<tr><td>Duration

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

125 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation</td></tr>
<tr><td colspan="6">Content and Educational Aims
Humans are a social species and interaction is crucial throughout the entire life span. While much of human communication involves language, there is a complex multichannel system of nonverbal communication that enriches linguistic content, provides context, and is also involved in structuring dynamic interaction. Interactants achieve goals by encoding information that is interpreted in the light of current context in transactions with others. This complexity implies also that there are frequent misunderstandings as a sender's intention is not fulfilled. Students in this course will learn to understand the structure of communication processes in a variety of formal and informal contexts. They will learn what constitutes challenges to achieving successful communication and to how to communicate effectively, taking the context and specific requirements for a target audience into consideration. These aspects will be discussed also in the scientific context, as well as business, and special cases, such as legal context – particularly with view to argumentation theory.

Communication is a truly transdisciplinary concept that involves knowledge from diverse fields such as biology, psychology, neuroscience, linguistics, sociology, philosophy, communication and information science. Students will learn what these different disciplines contribute to an understanding of communication and how theories from these fields can be applied in the real world. In the context of scientific communication, there will also be a focus on visual communication of data in different disciplines. Good practice examples will be contrasted with typical errors to facilitate successful communication also with view to the Bachelor's thesis.</td></tr>
</table>

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. analyze communication processes in formal and informal contexts.
2. identify challenges and failures in communication.
3. design communications to achieve specified goals to specific target groups.
4. understand the principles of argumentation theory.
5. use data visualization in scientific communications.

Indicative Literature

- Joseph A. DeVito: The Interpersonal Communication Book (Global edition, 16th edition), 2022
- Steven L. Franconeri, Lace M. Padilla, Priti Shah, Jeffrey M. Zacks, and Jessica Hullman: The Science of Visual Data Communication: What Works Psychological Science in the Public Interest, 22(3), 110–161, 2022
- Douglas Walton: Argumentation Theory – A Very Short Introduction. In: Simari, G., Rahwan, I. (eds) Argumentation in Artificial Intelligence. Springer, Boston, MA, 2009

Examination Type: Module Examination

Assessment Type: Digital submission of asynchronous presentation, including reflection

Duration/Length: Asynchronous/Digital submission
Weight: 100%

Scope: All intended learning outcomes of the module

Module achievement: Asynchronous presentation on a topic relating to the major of the student, including a reflection including concept outlining the rationale for how arguments are selected and presented based on a particular target group for a particular purpose. The presentation shall be multimedial and include the presentation of data

The module achievement ensures sufficient knowledge about key concepts of effective communication including a reflection on the presentation itself

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

8.2.9 Agency, Leadership, and Accountability

<table>
<tr><td colspan="3">Module Name
Agency, Leadership, and Accountability</td><td>Module Code
CTNS-NSK-09</td><td>Level (type)
Year 3
(New Skills)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTNS-09</td><td colspan="3">Agency, Leadership, and Accountability</td><td>Lecture</td><td>5</td></tr>
<tr><td>Module Coordinator

Prof. Dr. Jules Coleman</td><td colspan="3">Program Affiliation
CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

Mandatory elective</td></tr>
<tr><td rowspan="2">Entry Requirements

Pre-requisites

☒ none</td><td rowspan="2">Co-requisites

☒ none</td><td rowspan="2">Knowledge, Abilities, or Skills</td><td>Frequency

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

Online Lectures (35h)
Private Study (90h)</td></tr>
<tr><td>Duration</td><td colspan="2">Workload

125 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation</td></tr>
<tr><td colspan="6">Content and Educational Aims

Each of us is judged by the actions we undertake and held to account for the consequences of them. Sometimes we may be lucky and our bad acts don't have harmful effects on others. Other times we may be unlucky and reasonable decisions can lead to unexpected or unforeseen adverse consequences for others. We are therefore held accountable both for choices and for outcomes. In either case, accountability expresses the judgment that we bear responsibility for what we do and what happens as a result. But our responsibility and our accountability in these cases is closely connected to the idea that we have agency.

Agency presumes that we are the source of the choices we make and the actions that result from those choices. For some, this may entail the idea that we have free will. But there is scientific world view that holds that all actions are determined by the causes that explain them, which is the idea that if we knew the causes of your decisions in advance, we would know the decision you would make even before you made it. If that is so, how can your choice be free? And if it is not free, how can you be responsible for it? And if you cannot be responsible, how can we justifiably hold you to account for it?

These questions express the centuries old questions about the relationship between free will and a determinist world view: for some, the conflict between a scientific world view and a moral world view.

But we do not always act as individuals. In society we organize ourselves into groups: e.g. tightly organized social groups, loosely organized market economies, political societies, companies, and more. These groups have structure. Some individuals are given the responsibility of leading the group and of exercising authority. But one can exercise authority over others in a group merely by giving orders and threatening punishment for non-compliance.

Exercising authority is not the same thing as being a leader? For one can lead by example or by encouraging others to exercise personal judgment and authority. What then is the essence of leadership?

The module has several educational goals. The first is for students to understand the difference between actions that we undertake for which we can reasonably held accountable and things that we do but which we are not responsible for. For example, a twitch is an example of the latter, but so too may be a car accident we cause as</td></tr>
</table>

a result of a heart attack we had no way of anticipating or controlling. This suggests the importance of control to responsibility. At the heart of personal agency is the idea of control. The second goal is for students to understand what having control means. Some think that the scientific view is that the world is deterministic, and if it is then we cannot have any personal control over what happens, including what we do. Others think that the quantum scientific view entails a degree of indeterminacy and that free will and control are possible, but only in the sense of being unpredictable or random. But then random outcomes are not ones we control either. So, we will devote most attention to trying to understand the relationships between control, causation and predictability.
But we do not only exercise agency in isolation. Sometimes we act as part of groups and organizations. The law often recognizes ways in which groups and organizations can have rights, but is there a way in which we can understand how groups have responsibility for outcomes that they should be accountable for. We need to figure out then whether there is a notion of group agency that does not simply boil down to the sum of individual actions. We will explore the ways in which individual actions lead to collective agency.
Finally we will explore the ways in which occupying a leadership role can make one accountable for the actions of others over which one has authority.

Intended Learning Outcomes
Students acquire transferable and key skills in this module.

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

1. understand and reflect how the social and moral world views that rely on agency and responsibility are compatible, if they are, with current scientific world views.
2. understand how science is an economic sector, populated by large powerful organizations that set norms and fund research agendas.
3. identify the difference between being a leader of others or of a group – whether a research group or a lab or a company – and being in charge of the group.
4. learn to be a leader of others and groups. Understand that when one graduates one will enter not just a field of work but a heavily structured set of institutions and that one's agency and responsibility for what happens, what work gets done, its quality and value, will be affected accordingly.

Indicative Literature
Hull, David L. "Science as a Process." Science as a Process. University of Chicago Press, 2010;
Feinberg, Joel. "Doing & deserving; essays in the theory of responsibility." (1970).

Usability and Relationship to other Modules

Examination Type: Module Examination

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

Scope: All intended learning outcomes of the module

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

8.2.10 Community Impact Project

<table>
<tr><td colspan="3">Module Name
Community Impact Project</td><td>Module Code
CTNC-CIP-10</td><td>Level (type)
Year 3
(New Skills)</td><td>CP
5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTNC-10</td><td colspan="3">Community Impact Project</td><td>Project</td><td>5</td></tr>
<tr><td>Module Coordinator

CIP Faculty Coordinator</td><td colspan="3">Program Affiliation

• CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

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

Pre-requisites: ☒ at least 15 CP from CORE modules in the major
Co-requisites: ☒ None
Knowledge, Abilities, or Skills: Basic knowledge of the main concepts and methodological instruments of the respective disciplines</td><td>Frequency

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

• Introductory, accompanying, and final events: 10 hours
• Self-organized teamwork and/or practical work in the community: 115 hours</td></tr>
<tr><td>Duration

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

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

Develop or join a community impact project before the 5th or 6th semester based on the introductory events during the 4th semester by using the database of projects, communicating with fellow students and faculty, and finding potential companies, organizations, or communities to target.</td></tr>
<tr><td colspan="6">Content and Educational Aims

CIPs are self-organized, major-related, and problem-centered applications of students' acquired knowledge and skills. These activities will ideally be connected to their majors so that they will challenge the students' sense of practical relevance and social responsibility within the field of their studies. Projects will tackle real issues in their direct and/or broader social environment. These projects ideally connect the campus community to other communities, companies, or organizations in a mutually beneficial way.
Students are encouraged to create their own projects and find partners (e.g., companies, schools, NGOs), but will get help from the CIP faculty coordinator team and faculty mentors to do so. They can join and collaborate in interdisciplinary groups that attack a given issue from different disciplinary perspectives.
Student activities are self-organized but can draw on the support and guidance of both faculty and the CIP faculty coordinator team.</td></tr>
<tr><td colspan="6">Intended Learning Outcomes

The Community Impact Project is designed to convey the required personal and social competencies for enabling students to finish their studies at Constructor as socially conscious and responsible graduates (part of the Constructor mission) and to convey social and personal abilities to the students, including a practical awareness of the societal context and relevance of their academic discipline.

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

1. understand the real-life issues of communities, organizations, and industries and relate them to concepts in their own discipline;</td></tr>
</table>

2. enhance problem-solving skills and develop critical faculty, create solutions to problems, and communicate these solutions appropriately to their audience;
3. apply media and communication skills in diverse and non-peer social contexts;
4. develop an awareness of the societal relevance of their own scientific actions and a sense of social responsibility for their social surroundings;
5. reflect on their own behavior critically in relation to social expectations and consequences;
6. work in a team and deal with diversity, develop cooperation and conflict skills, and strengthen their empathy and tolerance for ambiguity.

Indicative Literature

Usability and Relationship to other Modules

- Students who have accomplished their CIP (6th semester) are encouraged to support their fellow students during the development phase of the next year's projects (4th semester).

Examination Type: Module Examination

Project, not numerically graded (pass/fail)
Scope: All intended learning outcomes of the module

8.3 Language and Humanities Modules

8.3.1 Languages

The descriptions of the language modules are provided in a separate document, the "Language Module Handbook" that can be accessed from the Constructor University's Language & Community Center internet sites (https://constructor.university/student-life/language-community-center/learning-languages).

8.3.2 Humanities

8.3.2.1 Introduction to Philosophical Ethics

<table>
<tr><td colspan="3">Module Name
Introduction to Philosophical Ethics</td><td>Module Code
CTHU-HUM-001</td><td>Level (type)
Year 1</td><td>CP
2.5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTHU-HUM-001</td><td colspan="3">Introduction to Philosophical Ethics</td><td>Lecture (online)</td><td>2.5</td></tr>
<tr><td>Module Coordinator

Dr. Eoin Ryan</td><td colspan="3">Program Affiliation

• CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

Mandatory elective</td></tr>
<tr><td colspan="3" rowspan="2">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

Online lectures (17.5 h)
Private Study (45h)</td></tr>
<tr><td>Duration

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

62.5 hours</td></tr>
<tr><td colspan="6">Recommendations for Preparation</td></tr>
<tr><td colspan="6">Content and Educational Aims
The nature of morality – how to lead a life that is good for yourself, and how to be good towards others – has been a central debate in philosophy since the time of Socrates, and it is a topic that continues to be vigorously discussed. This course will introduce students to some of the key aspects of philosophical ethics, including leading normative theories of ethics (e.g. consequentialism or utilitarianism, deontology, virtue ethics, natural law ethics, egoism) as well as some important questions from metaethics (are useful and generalizable ethical claims even possible; what do ethical speech and ethical judgements actually do or explain) and moral psychology (how do abstract ethical principles do when realized by human psychologies). The course will describe ideas that are key factors in ethics (free will, happiness, responsibility, good, evil, religion, rights) and indicate various routes to progress in understanding ethics, as well as some of their difficulties.</td></tr>
</table>

Intended Learning Outcomes
Upon completion of this module, students will be able to 1. describe normative ethical theories such as consequentialism, deontology and virtue ethics. 2. discuss some metaethical concerns. 3. dnalyze ethical language. 4. highlight complexities and contradictions in typical ethical commitments. 5. indicate common parameters for ethical discussions at individual and social levels. 6. analyze notions such as objectivity, subjectivity, universality, pluralism, value.
Indicative Literature
Simon Blackburn, Being Good (2009) Russ Shafer-Landay, A Concise Introduction to Ethics (2019) Mark van Roojen, Metaethicas: A Contemporary Introduction (2015)
Usability and Relationship to other Modules
Examination Type: Module Examination Assessment Type: Written Examination — Duration/Length: 60 min; Weight: 100% Scope: All intended learning outcomes of the module. Completion: To pass this module, the examination has to be passed with at least 45%.

8.3.2.2 Introduction to the Philosophy of Science

<table>
<tr><td colspan="2">Module Name
Introduction to the Philosophy of Science</td><td>Module Code
CTHU-HUM-002</td><td>Level (type)
Year 1</td><td>CP
2.5</td></tr>
<tr><td colspan="5">Module Components</td></tr>
<tr><td>Number</td><td colspan="2">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTHU-HUM-002</td><td colspan="2">Introduction to the Philosophy of Science</td><td>Lecture (online)</td><td>2.5</td></tr>
<tr><td>Module Coordinator

Dr. Eoin Ryan</td><td colspan="2">Program Affiliation

• CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

Mandatory elective</td></tr>
<tr><td colspan="2" rowspan="2">Entry Requirements

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

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

Online lectures (17.5h)
Private Study (45h)</td></tr>
<tr><td>Duration

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

62.5 hours</td></tr>
<tr><td colspan="5">Recommendations for Preparation</td></tr>
<tr><td colspan="5">Content and Educational Aims

This humanities module will introduce students to some of the central ideas in philosophy of science. Topics will include distinguishing science from pseudo-science, types of inference and the problem of induction, the pros and cons of realism and anti-realism, the role of explanation, the nature of scientific change, the difference between natural and social sciences, scientism and the values of science, as well as some examples from philosophy of the special sciences (e.g., physics, biology).
The course aims to give students an understanding of how science produces knowledge, and some of the various contexts and issues which mean this process is never entirely transparent, neutral, or unproblematic. Students will gain a critical understanding of science as a human practice and technology; this will enable them both to better understand the importance and success of science, but also how to properly critique science when appropriate.</td></tr>
<tr><td colspan="5">Intended Learning Outcomes

Upon completion of this module, students will be able to
1. understand key ideas from the philosophy of science.
2. discuss different types of inference and rational processes.
3. describe differences between how the natural sciences, social sciences and humanities discover knowledge.
4. identify ways in which science can be more and less value-laden.
5. illustrate some important conceptual leaps in the history of science.</td></tr>
<tr><td colspan="5">Indicative Literature

Peter Godfrey-Smith, Theory and Reality (2021)

James Ladyman, Understanding Philosophy of Science (2002)

Paul Song, Philosophy of Science: Perspectives from Scientists (2022)</td></tr>
</table>

Usability and Relationship to other Modules
Examination Type: Module Examination Assessment Type: Written Examination — Duration/Length: 60 min Weight: 100% Scope: All intended learning outcomes of the module. Completion: To pass this module, the examination has to be passed with at least 45%.

8.3.2.3 Introduction to Visual Culture

<table>
<tr><td colspan="3">Module Name
Introduction to Visual Culture</td><td>Module Code
CTHU-HUM-003</td><td>Level (type)
Year 1</td><td>CP
2.5</td></tr>
<tr><td colspan="6">Module Components</td></tr>
<tr><td>Number</td><td colspan="3">Name</td><td>Type</td><td>CP</td></tr>
<tr><td>CTHU-HUM-003</td><td colspan="3">Introduction to Visual Culture</td><td>Lecture (online)</td><td>2.5</td></tr>
<tr><td>Module Coordinator

Dr. Irina Chiaburu</td><td colspan="3">Program Affiliation

• CONSTRUCTOR Track Area</td><td colspan="2">Mandatory Status

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

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

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

Online lectures (17.5 h)
Private Study (45h)</td></tr>
<tr><td>Duration

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

62.5 h</td></tr>
<tr><td colspan="6">Recommendations for Preparation</td></tr>
</table>

Content and Educational Aims

Of the five senses, the sense of sight has for a long time occupied the central position in human cultures. As John Berger has suggested this could be because we can see and recognize the world around us before we learn how to speak. Images have been with us since the earliest days of the human history. In fact, the earliest records of human history are images found on cave walls across the world. We use images to capture abstract ideas, to catalogue and organize the world, to represent the world, to capture specific moments, to trace time and change, to tell stories, to express feelings, to better understand, to provide evidence and more. At the same time, images exert their power on us, seducing us into believing in their 'innocence', that is into forgetting that as representations they are also interpretations, i.e., a particular version of the world.

The purpose of this course is to explore multiple ways in which images and the visual in general mediate and structure human experiences and practices from more specialized discourses, e.g., scientific discourses, to more informal and personal day-to-day practices, such as self-fashioning in cyberspace. We will look at how social and historical contexts affect how we see, as well as what is visible and what is not. We will explore the centrality of the visual to the intellectual activity, from early genres of scientific drawing to visualizations of big data. We will examine whether one can speak of visual culture of protest, look at the relationship between looking and subjectivity and, most importantly, ponder the relationship between the visual and the real.

Intended Learning Outcomes

Upon completion of this module, students will be able to

1. understand a range of key concepts pertaining to visual culture, art theory and cultural analysis
2. understand the role visuality plays in development and maintenance of political, social, and intellectual discourses
3. think critically about images and their contexts
4. reflect critically on the connection between seeing and knowing

Indicative Literature

Berger, J., Blomberg, S., Fox, C., Dibb, M., & Hollis, R. (1973). Ways of seeing.
Foucault, M. (2002). The order of things: an archaeology of the human sciences (Ser. Routledge classics). Routledge.
Hunt, L. (2004). Politics, culture, and class in the French revolution: twentieth anniversary edition, with a new preface (Ser. Studies on the history of society and culture, 1). University of California Press.
Miller, V. (2020). Understanding digital culture (Second). SAGE.
Thomas, N. (1994). Colonialism's culture: anthropology, travel and government. Polity Press.

Usability and Relationship to other Modules

Examination Type: Module Examination

Assessment: Written examination

Duration/Length: 60 min.
Weight: 100%

Scope: all intended learning outcomes

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

9 Appendix

9.1 Intended Learning Outcomes Assessment-Matrix

Software, Data and Technology BSc					Programming in Python and C++	Analysis	Core Algorithms and Data Structures	Development in JVM Languages	Linear Algebra	Operating Systems	Software Engineering and Design	Computer Architecture	Advanced Algorithms and Data Structures	Machine Learning	Functional Programming	Scientific Data Analysis	Databases	Discrete Mathematics	Artificial Intelligence	Specialization modules	Bachelor Thesis	CT Methods	CT New Skills	CT Language and Humanities
Semester					1	1	2	2	2	3	4	3	3	4	3	3	4	4	4	5-6	6	1-4	3-6	1-2
Mandatory/ mandatory elective					m	m	m	m	me	m	m	m	m	m	me	me	me	me	me	me	m	me	me	me
Credits					7,5	7,5	7,5	7,5	7,5	7,5	7,5	5,0	5,0	5,0	5,0	5,0	5,0	5,0	5,0	15,0	15	20	20,0	5,0
	Competencies*																							
Program Learning Outcomes	**A**	**E**	**P**	**S**																				
work professionally in the field of software, Data and Technology and enter graduate programs related to these fields	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
apply fundamental concepts of mathematics, statistics, and computer science while solving data-related problems;	x	x				x	x		x				x	x		x		x	x	x	x	x	x	
analyze at multiple levels of abstraction and use appropriate mathematical and computational methods to model and analyze real-world problems;	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
develop, analyze and implement algorithms using modern software engineering methods and programming languages;	x	x			x		x	x					x		x	x	x		x	x	x			
understand the characteristics of a range of computing platforms and their advantages and limitations;	x	x								x	x	x					x			x	x			
choose from multiple programming paradigms, languages and algorithms to solve a given problem adequately;	x	x			x		x	x		x	x	x	x		x		x			x	x			
apply the necessary mathematical methods, such as linear algebra, analysis, calculus, and discrete mathematics;	x					x	x		x				x	x	x			x		x	x	x		
recognize the context in which data science and software systems operate, including interactions with people and the physical world;	x	x	x	x	x			x		x	x	x		x	x	x	x		x	x	x	x	x	
describe the state of published knowledge in the field of software, Data and Technology and in a chosen specialization (Machine Learning, Software Development, Programming Languages);	x	x	x		x			x		x	x	x		x	x	x	x	x	x	x	x	x		
analyze and model real-life scenarios in organizations and industries using contemporary techniques of data science and software development, also taking methods and insights of other disciplines into account;	x	x	x		x		x	x		x	x	x	x	x	x	x	x	x	x	x	x	x		
appropriately communicate solutions of problems in software, Data and Technology in both spoken and written form to specialists and non-specialists;	x	x	x	x						x										x	x		x	x
draw scientifically founded conclusions that consider social, professional, scientific, and ethical aspects;	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
work effectively in a diverse team and take responsibility in a team;	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, reflecting on their practice and evaluating critical feedback;																					x	x	x	x
adhere to and defend ethical, scientific, and professional standards;	x	x	x	x																	x	x	x	
Assessment Type																								
Oral examination																								
Final written exam					x	x	x	x	x	x	x	x	x	x	x		x	x	x			x	x	
Project											x										x		x	
Essay																								
Lab report																								
Poster presentation																								
Presentation																					x		x	
Practical assessment					x		x	x				x	x		x		x							
Portfolio																x								
Various																								x
Module achievements																							x	x

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

Figure 4: Intended Learning Outcomes Assessment Matrix