Advanced Software Technology

Master of Science

Subject-specific Examination Regulations for Advanced Software Technology

The subject-specific examination regulations for Advanced Software Technology are defined by this program handbook and are valid only in combination with the General Examination Regulations for Master degree programs ("General Master Policies").

This handbook also contains the program-specific Study and Examination Plan in chapter 2.2.

Upon graduation students in this program will receive a Master of Science (MSc) degree with a scope of 120 ECTS credit points (for specifics see chapter 2 and 5 of this handbook).

Valid for all students starting their studies in Fall 2025.

Version	Valid as of	Decision	Details
Fall 2025- V1.1		Feb 02, 2026	Implementation of Examination Concept according to policies.
Fall 2025-V1	Sep 01, 2025		
		June 28, 2023	Academic Senate approval of study program name change from "Data Science and Software Development" to "Advanced Software Technology
		May 24, 2023	Originally approved by Academic Senate

Contents

1 Program Overview

1.1 Concept

The Master of Science in Advanced Software Technology at Constructor University is a consecutive master program that prepares students to become the next generation of experts in the field of advanced software technology. The program offers a unique opportunity to gain a solid education in software development, data science and programming languages, which are at the forefront of digitalization and are driving the digital transformation of industry and society. The program is designed to provide students with a solid foundation in Mathematics and basic programming skills, a comprehensive understanding of the latest research and technology in these areas, as well as essential management and leadership skills, so that they can become technology leaders in research and industry.

The program offers three tracks: Data Science, Software Development, and Programming Languages, allowing students to specialize in the area of their choice. The program also includes common modules for all students, such as Architectural Strategy, Programming Languages in Software Development, Big Data Software Engineering, Capstone Project, Technological Entrepreneurship, Product Innovation & Marketing, Quality Engineering, Kotlin Ecosystem, Data Analytics, and Agile Product Development & Design.

The special modules for the Data Science track include Machine Learning Applications, Computer Vision, Machine Learning in Software Engineering, and Bayesian Methods in Machine Learning. The special modules for the Software Development track include Static Program Analysis, System Security, Cryptography, Network Security and IDE Development. For the Programming Languages track, the special modules are Advanced Functional Programming, Formal Verification, Virtual Machines in Compilers, Dependent Types, Type Theory, and Category Theory for Programmers.

The program will be taught by distinguished experts in the field from Constructor University and JetBrains, guaranteeing excellent teaching competence and hands-on experience from the forefront of the state of the art in research and industry. In addition, students will have access to real-world applications and the IT job market via JetBrains' excellent international network, and will be supported by the Constructor University Student Career Support.

The program will also make use of contemporary blended e-learning techniques, flipped classroom teaching, and team-based work on software projects, allowing for a student-centric and hands-on experience. Together with the availability of state-of-the-art software and hardware at Constructor University and the support of JetBrains, the program allows seamless collaboration among students and instructors of different institutions, and adapts to conditions that may arise from pandemic emergencies.

Students will acquire the core expertise of digital leaders, with a solid technological backbone developed along three complementary tracks, with additional core management and leadership skills. They will acquire the essential soft skills for an active digital technology leadership in the contemporary global and multiethnic society, thanks to the international environment that characterizes Constructor University and JetBrains. Overall, this education will enable them to enter research via Ph.D. programs and to succeed in the job market in high profile roles.

1.2 Qualification Aims

1.2.1 Educational Aims

The MSc Advanced Software Technology program at Constructor University aims to provide students with an in-depth understanding of the essential aspects of designing and development of software products with a focus on Data Science. The program comprises three main tracks: Data Science, Software Development, and Programming Languages Tools. Students will acquire the skills necessary to apply methods and tools to successfully and responsibly engineer software, with a special emphasis on the use of JetBrains tools.

The program seeks to expand the participant's competencies and capabilities in the subject areas of Data Science, Software Development and Programming Languages, which play a dominant role in industries and research. Each student will select one of these areas as their main specialization, and the curriculum will provide them with modern cross-disciplinary leadership and management competencies to become tomorrow's digital leaders.

Throughout the program, students will be introduced to practical and research-oriented work through a Capstone project, an elective research project, and a thesis, which will be supported by frequent individual feedback sessions and personal guidance. This will facilitate and quicken the students' career development and help them to become valuable assets in industries and research within a short period of time.

Constructor University programs are offered in a highly intercultural environment. Students will acquire intercultural competence as part of their education through everyday group work, class participation, and extracurricular activities. In this way, students will gain practical intercultural competencies and build their confidence in an English-speaking work and study environment.

To summarize, graduates of the MSc Advanced Software Technology program will have obtained the following competences and skills:

1. Subject-matter competence in a Data Science, Software Development or Programming Languages specialization

Graduates will have an in-depth knowledge of one of the fields of Data Science, Software Development or Programming Languages. They will be able to define and interpret the doctrine of the field, and will have also developed a detailed and critical understanding at the cutting edge of knowledge in the field.

2. Advanced Software Technology Competency

Graduates will have a broadened and deepened knowledge in their formal, algorithmic, and applied competencies in Advanced Software Technology. This will enable them to develop independent ideas as digital experts.

3. Learning, transfer, and research skills

The Program will enable students to apply problem solutions in new and unfamiliar situations. They will integrate learned skills in complex and multidisciplinary contexts, as it is more and more necessary in industry and research. In particular, graduates will be able to design research questions, select appropriate methods, and document and interpret research results.

4. Management and Leadership Skills

Recognizing the ever-increasing need for management and leadership skills in business, industry and research, graduates will have a broad and integrated knowledge and understanding of the fundamentals from management and leadership. Their knowledge corresponds to the standard literature in the field. In particular, they will be able to solve related problems in the field of Advanced Software Technology with professional plausibility.

5. Teamwork and communication skills

Graduates will be proficient in the specialized exchange of ideas in a group setting with the goal of collaborative development of a digital software or hardware system. This will be reinforced by effective and reflective practice of communication and collaboration on both academic and non-academic topics.

6. Personal and Professional Competence

Graduates will be able to make, justify and reflect on decisions based on theoretical and professional knowledge. They will be able to critically examine their own behavior and assess social consequences. In doing so, they will act appropriately to the situation. Thus, they will be able to develop a professional profile both in and out of academia.

1.2.2 Intended Learning Outcomes

Upon completion of this program, students will be able to

1. critically assess and creatively apply technological possibilities and innovations in the fields of data science, software development and programming languages;
2. critically assess and apply software engineering methodologies considering real life situations, organizations and industries;
3. use, adapt and improve modern techniques in data science, such as deep learning, recommender systems, computer vision, and machine learning in software engineering;
4. apply cross-disciplinary management methodologies to solve academic and professional problems in the context of software development and data science;

5. critically assess and integrate a consistent tool set of leadership abilities into a professional work environment;
6. plan, conduct and document small research projects in the context of data science, software development and programming languages;
7. independently research, document and present a scientific topic with appropriate language skills;
8. use scientific methods as appropriate in the field of data science and software engineering such as defining research questions, justifying methods, collecting, assessing and interpreting relevant information, and drawing scientifically-founded conclusions that consider social, scientific and ethical insights;
9. develop and advance solutions to problems and arguments in their subject area and defend these in discussions with specialists and non-specialists;
10. engage ethically with academic, professional and wider communities and to actively contribute to a sustainable future, reflecting and respecting different views;
11. take responsibility for their own learning, personal and professional development and role in society, evaluating critical feedback and self-analysis;
12. apply their knowledge and understanding of data science, software development, and programming languages to a professional context;
13. take on responsibility in a diverse team;
14. adhere to and defend ethical, scientific and professional standards;
15. apply data analytics techniques;
16. understand and utilize agile product development and design methodologies;
17. understand and apply principles of quality engineering.

1.3 Target Audience

The MSc Advanced Software Technology Program at Constructor University is designed for students of diverse backgrounds, with a focus on those who have completed an undergraduate program in Computer Science or a related field. The program is tailored for graduates who are interested in gaining advanced knowledge and skills in the fields of Data Science, Software Development and Programming Languages.

This program is particularly suitable for candidates who are dedicated to and interested in gaining theoretical and application-oriented knowledge in the fields of Data Science, Machine Learning, Software Engineering, Cybersecurity, Artificial Intelligence, and Programming Languages.

The program prepares students for key roles in the IT industry, as well as for entering research in the subject fields. Additionally, the program provides students with additional educational opportunities in management and leadership, which can prepare them to develop their own start-up. The program's educational approach encourages exchange and discussion within the student community, making the willingness to interact, appreciate different teaching and learning formats, accept challenges and develop professionally during study, important requirements for successful participation in the program.

1.4 Career Options

The field of Advanced Software Technology is rapidly growing and in high demand as more and more companies are recognizing the value of data-driven decision making. Graduates of the MSc Advanced Software Technology program at Constructor University will be well-equipped to enter a variety of exciting and rewarding careers in the IT industry.

Graduates of this program will be well-prepared for roles in data analysis and software development, such as data scientists, software engineers, and machine learning engineers. They will also be able to work in a wide range of industries, including finance, healthcare, education, and technology. The program's focus on advanced software technology provides students with a versatile skill set that will be highly valued by employers.

Constructor University's Student Career Services and Alumni Association, as well as the university's partnerships with leading technology companies such as JetBrains, Acronis, Alemira, Virtuozzo and Rolos, will provide students with valuable support and opportunities for professional growth. The Student Career Services offers high-quality training and coaching in application and interview preparation, effective presenting, business etiquette, and employer research, while the Alumni Association helps students establish a long-lasting worldwide network. These resources, along with the university's industry connections, will help graduates succeed in their chosen careers.

1.5 Admission Requirements

The Advanced Software Technology graduate program requires students to have completed an undergraduate program in computer science, data science, software development, information technology or another discipline with at least 60 ECTS of computer science-related topics (such as mathematics, programming, design, software architecture).

Admission to Constructor University is selective and based on a candidate’s university achievements, recommendations and self-presentation. 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:

- Letter of motivation
- Curriculum vitae (CV)
- Official or certified copies of university transcripts
- Bachelor’s degree certificate or equivalent
- Language proficiency test results (minimum score of 90 (TOEFL), 6.5 (IELTS) or 110 (Duolingo)).
- Copy of Passport
- Letter of recommendation (optional).

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-graduate.

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

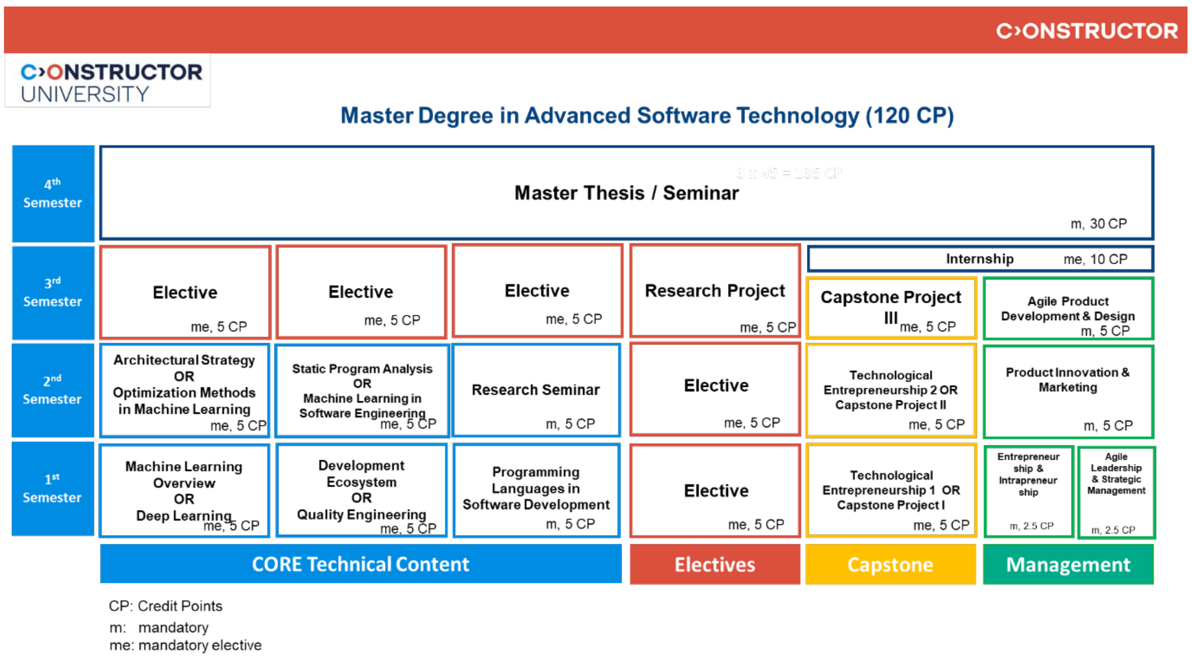
or visit our program website: Advanced Software Technology | Constructor University

For more information on Student Services please visit:
https://constructor.university/student-life/student-service

2 The Curriculum

2.1 The Curriculum at a Glance

The Advanced Software Technology graduate program is composed of foundational lectures, specialized modules, and applied project work, leading to a master thesis that can be conducted in research groups at Constructor University, at external research institutes or in close collaboration with a company. The program takes four semesters (two years). The following table shows an overview of the modular structure of the program. The program is sectioned into two areas (AST and Management modules) and the Master Thesis. All credit points (CP) are ECTS (European Credit Transfer System) credit points. In order to graduate, students need to obtain 120 CP. See Chapter 3 “Modules” of this handbook for the detailed module descriptions or refer to CampusNet.



C›ONSTRUCTOR

C›ONSTRUCTOR UNIVERSITY

Master Degree in Advanced Software Technology (120 CP)

4th Semester	Master Thesis / Seminar, 3 x 45 = 135 CP, m, 30 CP					
3rd Semester	Elective, me, 5 CP	Elective, me, 5 CP	Elective, me, 5 CP	Research Project, me, 5 CP	Internship, me, 10 CP	
					Capstone Project III, me, 5 CP	Agile Product Development & Design, m, 5 CP
2nd Semester	Architectural Strategy OR Optimization Methods in Machine Learning, me, 5 CP	Static Program Analysis OR Machine Learning in Software Engineering, me, 5 CP	Research Seminar, m, 5 CP	Elective, me, 5 CP	Technological Entrepreneurship 2 OR Capstone Project II, me, 5 CP	Product Innovation & Marketing, m, 5 CP
1st Semester	Machine Learning Overview OR Deep Learning, me, 5 CP	Development Ecosystem OR Quality Engineering, me, 5 CP	Programming Languages in Software Development, m, 5 CP	Elective, me, 5 CP	Technological Entrepreneurship 1 OR Capstone Project I, me, 5 CP	Entrepreneurship & Intrapreneurship, m, 2.5 CP; Agile Leadership & Strategic Management, m, 2.5 CP
	CORE Technical Content			Electives	Capstone	Management

CP: Credit Points
m: mandatory
me: mandatory elective

2.2 Study and Examination Plan

MSc Degree in Advanced Software Technology
Matriculation Fall 2025

Module Code	Program-Specific Modules	Type	Assessment	Period[1]	Status[2]	Semester	CP
Semester 1							**30**
	Unit: CORE modules						**20**
MCSSE-SE-02	**Module: Quality Engineering**				**me**	**1**	**5**
MCSSE-SE-02	Quality Engineering	Lecture	Portfolio	During semester			
MCSSE-AI-01	**Module: Deep Learning**				**me**	**1**	**5**
MCSSE-AI-01	Deep Learning	Lecture	Written examination	Examination period			
MAST-101	**Module: Development Ecosystem**				**me**	**1**	**5**
MAST-101-A	Development Ecosystem	Lecture /Tutorial	Program Code	Examination period			
MAST-114	**Module: Machine Learning Overview**				**me**	**1**	**5**
MAST-114-A	Machine Learning Overview	Lecture	Program Code	During semester			
MAST-102	**Module: Programming Languages in Software Development**				**m**	**1**	**5**
MAST-102-A	Programming Languages in Software Development	Lecture/Tutorial	Program code	During semester			
	Further CORE modules				**me**	**1**	**5**
	- students choose 1 module from those listed below						
	Unit: Capstone Project						**5**
MCSSE-CAP-01	**Module: Capstone Project 1**				**me**	**1**	**5**
MCSSE-CAP-01	Capstone Project 1	Project	Project Assessment	During semester			
MAST-111	**Module: Technological Entrepreneurship 1**				**me**	**1**	**5**
MAST-111-A	Technological Entrepreneurship 1	Lecture	Project Assessment	During semester			
	Unit: Management and Leadership Modules						**5**
MCSSE-LAS-01	**Module: Entrepreneurship & Intrapreneurship**				**m**	**1**	**2.5**
MCSSE-LAS-01	Entrepreneurship & Intrapreneurship	Lecture	Presentation	During semester			
MCSSE-LAS-03	**Module: Agile Leadership and Strategic Management**				**m**	**1**	**2,5**
MCSSE-LAS-03	Agile Leadership and Strategic Management	Lecture	Presentation	During semester			
Semester 2							**30**
	Unit: CORE modules						**20**
MCSSE-SE-03	**Module: Architectural Strategy**				**me**	**2**	**5**
MCSSE-SE-03	Architectural Strategy	Lecture	Portfolio	During semester			
MAST-203	**Module: Machine Learning in Software Engineering**					**2**	**5**
MAST-203-A	Machine Learning in Software Engineering	Lecture	Written Examination	Examination period			2.5
MAST-203-B	Machine Learning in Software Engineering - Tutorial	Tutorial	Practical Assessment	During semester			2.5
MAST-110	**Module: Optimization Methods in Machine Learning**				**me**	**2**	**5**
MAST-110-A	Optimization Methods in Machine Learning	Lecture	Written Examination	Examination period			2.5
MAST-110-B	Optimization Methods in Machine Learning - Tutorial	Tutorial	Program Code	During semester			2.5
MAST-205	**Module: Static Program Analysis**				**me**	**2**	**5**
MAST-205-A	Static Program Analysis	Lecture	Oral Examination	Examination period			2.5
MAST-205-B	Static Program Analysis Tutorial	Tutorial	Practical Assessment	During semester			2.5
MAST-115	**Module: Research Seminar**				**m**	**2**	**5**
MAST-115-A	Research Seminar	Seminar	Presentation	During Semester			
	Further CORE modules				**me**	**2**	**5**
	- students choose 1 module from those listed below						
	Unit: Capstone Project						**5**
MCSSE-CAP-02	**Module: Capstone Project 2**				**me**	**2**	**5**
MCSSE-CAP-02	Capstone Project 2	Project	Project Assessment	During semester			
MAST-112	**Module: Technological Entrepreneurship 2**				**me**	**2**	**5**
MAST-112-A	Technological Entrepreneurship 2	Lecture	Project Assessment	During semester			
	Management Modules						**5**
MCSSE-MGT-02	**Module: Product Innovation & Marketing**				**m**	**2**	**5**
MCSSE-MGT-02	Product Innovation & Marketing	Lecture	Presentation	During semester			
Semester 3							**30**
	Unit: CORE modules						**20**
	Further CORE modules				**me**	**3**	**20**
	- students choose 4 modules from those listed below. One CORE module can be replaced by the Research Project module.						
	Unit: Capstone Project						**5**
MCSSE-CAP-03	**Module: Capstone Project 3**				**me**	**3**	**5**
MCSSE-CAP-03	Capstone Project 3	Project	Project Assessment	During semester			
	Unit: Management and Leadership Modules						**5**
MCSSE-MGT-01	**Module: Agile Product Development & Design**				**m**	**3**	**5**
MCSSE-MGT-01	Agile Product Development & Design	Lecture	Presentation	Examination period			
Semester 4							**30**
	Master Thesis						**30**
MAST-300	**Module: Master Thesis MSc AST**				**m**	**4**	**30**
MAST-300-T	Master Thesis AST	Thesis					
Total CP							**120**

[1] Each lecture period lasts 14 semester weeks and is followed by reading and examination days. Written examinations are centrally scheduled during weeks 15 and 16. For all

[2] m = mandatory, me = mandatory elective

Further CORE modules							
Module Code	**Program-Specific Modules**	**Type**	**Assessment**	**Period**[1]	**Status**[2]	**Semester**	**CP**
Data Science Track							
MAST-113	**Module: Industrial Machine Learning on Hadoop and Spark**				**me**	**2/4**	**5**
MAST-113-A	Hadoop and Spark Theory	Lecture	Program code	During semester			2.5
MAST-113-B	Industrial Machine Learning	Tutorial					2.5
MAST-202	**Module: Machine Learning Applications**				**me**	**3**	**5**
MAST-202-A	Machine Learning Applications	Lecture/ Tutorial	Program code	During semester			
MAST-203	**Module: Machine Learning in Software Engineering**				**me**	**2**	**5**
MAST-203-A	Machine Learning in Software Engineering	Lecture	Written examination	Examination Period			2.5
MAST-203-B	Machine Learning in Software Engineering Tutorial	Tutorial	Practical assessment	During semester			2.5
MAST-204	**Module: Bayesian Methods in Machine Learning**				**me**	**1**	**5**
MAST-204-A	Bayesian Methods in Machine Learning	Lecture	Written examination	Examination Period			2.5
MAST-204-B	Bayesian Methods in Machine Learning Tutorial	Tutorial	Program code	During semester			2.5
MAST-109	**Module: Deep Bayesian Models**				**me**	**2**	**5**
MAST-109-A	Deep Bayesian Models	Lecture	Written examination	Examination Period			2.5
MAST-109-B	Deep Bayesian Models Tutorial	Tutorial	Program code	During semester			2.5
MAST-212	**Reinforcement Learning**				**me**	**3**	**5**
MAST-212-A	Reinforcement Learning	Lecture	Written examination	Examination Period			2.5
MAST-212-B	Reinforcement Learning Tutorial	Tutorial	Program code	During semester			2.5
MDE-CO-02	**Module: Data Analytics**				**me**	**1**	**5**
MDE-CO-02	Data Analytics	Lecture	Project report	During semester			
MAST-213	**Large Scale Deep Learning Models**				**me**	**3**	**5**
MAST-213-A	Large Scale Deep Learning Models	Lecture	Written examination	Examination Period			2.5
MAST-213-B	Large Scale Deep Learning Models Tutorial	Tutorial	Program code	During semester			2.5
Software Development Track							
MAST-205	**Module: Static Program Analysis**				**me**	**2**	**5**
MAST-205-A	Static Program Analysis	Lecture	Oral examination	Examination Period			2.5
MAST-205-B	Static Program Analysis Tutorial	Tutorial	Practical assessment	During semester			2.5
MCSSE-CYB-01	**Module: Cryptography**				**me**	**1**	**5**
MCSSE-CYB-01	Cryptography	Lecture	Written examination	Examination Period			
MCSSE-CYB-02	**Module: System Security**				**me**	**2**	**5**
MCSSE-CYB-02	System Security	Lecture	Written examination	Examination Period			
MCSSE-CYB-03	**Module: Network Security**				**me**	**3**	**5**
MCSSE-CYB-03	Network Security	Lecture	Written examination	Examination Period			
MAST-103	**Module: Big Data Software Engineering**				**me**	**2**	**5**
MAST-103-A	Big Data Software Engineering	Lecture/Tutorial	Program code	During semester			
MAST-207	**Module: IDE Development**				**me**	**1**	**5**
MAST-207-A	IDE Development	Lecture/ Tutorial	Program code	During semester			
Programming Languages Track							
MAST-104	**Module: Advanced Functional Programming**				**me**	**1**	**5**
MAST-104-A	Advanced Functional Programming	Lecture	Written examination	Examination Period			2.5
MAST-104-B	Advanced Functional Programming Tutorial	Tutorial	Program code	During semester			2.5
MAST-214	**Module: Formal Verification**				**me**	**3**	**5**
MAST-214-A	Formal Verification	Lecture/Tutorial	Program code	During semester			5
MAST-106	**Module: Virtual Machines in Compilers**				**me**	**1/3**	**5**
MAST-106-A	Virtual Machines in Compilers	Lecture/Tutorial	Program code	During semester			2.5
MAST-209	**Module: Dependent Types**				**me**	**3**	**5**
MAST-209-A	Dependent Types	Lecture/ Tutorial	Practical assessments	During semester			
MAST-210	**Module: Type Theory**				**me**	**3**	**5**
MAST-210-A	Type Theory	Lecture/Tutorial	Program code	During semester			
MAST-211	**Module: Category Theory for Programmers**				**me**	**2**	**5**
MAST-211-A	Category Theory for Programmers	Lecture / Tutorial	Written examination	Examination Period			
	Research Project						**5**
MAST-201	**Module: Research Project**				**me**	**3**	**5**
MAST-201-A	Research Project	Project	Project Report	Examination period			

Figure 2: Study and Examination Plan

2.3 Core Area (30 CP)

This area is the centerpiece of the Advanced Software Technology program. The six mandatory modules in the Core Area cover essential methods of Advanced Software Technology. They provide the foundations for further, more advanced modules and applied projects by introducing the fundamental concepts, methods and technologies used in Advanced Software Technology. The modules are intensive courses accompanied by hands-on tutorials and labs.

To pursue an AST master, the following CORE modules (10 CP) need to be taken as mandatory modules (m):

- CORE Module: Research Seminar (m, 5 CP)
- CORE Module: Programming Languages in Software Development (m, 5 CP)

Further 20 CP must be taken from the following CORE modules need to be taken:

- CORE Module: Machine Learning Overview (me, 5 CP) **OR** Deep Learning (me, 5 CP)
- CORE Module: Development Ecosystem (me, 5 CP) **OR** Quality Engineering (me, 5 CP)
- CORE Module: Architectural Strategy (me, 5 CP) **OR** Optimization Methods in Machine Learning (me, 5 CP)
- CORE Module: Static Program Analysis (me, 5CP) **OR** Machine Learning in Software Engineering (me, 5 CP)

2.4 Elective Area (30 CP)

The Advanced Software Technology program attracts students with diverse career goals, backgrounds, and prior work experience. Therefore, modules in this area can be chosen freely by students depending on their prior knowledge and interests. Students can choose to strengthen their knowledge by following one of suggested focus tracks and electing the modules offered therein: Data Science, Software Development, and Programming Languages.

Students may choose any combination of the modules listed below. Each track may be followed completely and/or complemented with other modules). In addition to the modules offered within these focus tracks, 3rd year modules from the undergraduate curriculum or other graduate programs at Constructor University can be taken with the approval of the program coordinator. Please see CampusNet for current offerings.

To pursue an AST master, students choose the following Electives modules (30 CP) as mandatory elective modules (me):

Data Science Track:

- Elective Module: Machine Learning Applications (me, 5 CP)
- Elective Module: Bayesian Methods in Machine Learning (me, 5 CP)
- Elective Module: Deep Bayesian Models (me, 5 CP)
- Elective Module: Data Analytics (me, 5 CP)
- Elective Module: Reinforcement Learning (me, 5 CP)
- Elective Module: Large Scale Deep Learning Models (me, 5 CP)
- Elective Module: Industrial Machine Learning on Hadoop and Spark (me, 5CP)

Software Development Track:

- Elective Module: Big Data Software Engineering (me, 5 CP)
- Elective Module: Cryptography (me, 5 CP)
- Elective Module: System Security (me, 5 CP)
- Elective Module: Network Security (me, 5 CP)
- Elective Module: IDE Development (me, 5 CP)

Programming Language Track:

- Elective Module: Advanced Functional Programming (me, 5 CP)
- Elective Module: Virtual Machines in Compilers (me, 5 CP)
- Elective Module: Formal Verification (me, 5 CP)
- Elective Module: Dependent Types (me, 5 CP)
- Elective Module: Type Theory (me, 5 CP)
- Elective Module: Category Theory for Programmers (me, 5 CP)

2.5 Management Area (15 CP)

To equip students with market-relevant management skills they take modules in the fields of product development, marketing and change management. All modules are mandatory for the program.

To pursue an AST master, the following Management modules (15 CP) need to be taken as mandatory modules (m):

- Management Module: Agile Product Development & Design (m, 5 CP)
- Management Module: Product Innovation & Marketing (m, 5 CP)
- Management Module: Entrepreneurship & Intrapreneurship (m, 2.5 CP)
- Management Module: Agile Leadership and Strategic Management (m, 2.5 CP)

2.6 Capstone project, Research project and Master Thesis (45 CP)

To explore the full development process of a software application with relation to the areas of specialization of the program, all students take the following in their first and second semesters:

- Capstone Module: Capstone Project 1 (me, 5 CP) OR Technological Entrepreneurship 1 (me, 5CP)
- Capstone Module: Capstone Project 2 (me, 5 CP) OR Technological Entrepreneurship 2 (me, 5CP)

Alternatively, students can take the 10 ECTS **Internship** module in place of the following modules: “Capstone 3”, and one elective module.

- Capstone Module: Capstone Project 3 (me, 5 CP)
- Elective Module of choice (me, 5 CP)

Students can replace in their third semester one Elective Module with the Research Project (5 CP):

Research Project Module: Research Project (me, 5 CP)

Students with a strong drive towards academic research can replace in their third semester one Elective Module by the Research Project, which is carried out in cooperation with JetBrains. The JetBrains researcher will provide research topics for the students. In the fourth semester, students conduct research and write a master thesis guided and supported by their academic advisor.

To pursue an AST master, the following Master Thesis module need to be taken as mandatory module:

- Thesis Module: Master Thesis (m, 30 CP)

3 Advanced Software Technology Graduate Program Regulations

3.1 Scope of These Regulations

The regulations in this handbook are valid for all students who entered the Advanced Software Technology graduate program at Constructor University in Fall 2025. In case of conflict between the regulations in this handbook and the general policies for Master Studies, the latter apply (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).

Updates to Study Program Handbooks are based on the policies approved by the Academic Senate on substantial and nonsubstantial changes to study programs. Students are integrated in the decision-making process through their respective committee representatives. All students affected by the changes will be properly informed.

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

3.2 Examination Concept

According to the Policies for Bachelor and Master studies, modules generally carry at least five ECTS. Each program ensures appropriate examination frequency and organization, justified in an examination concept and regularly reviewed with student involvement.

Constructor University's examination concept follows the principle of Constructive Alignment (Biggs 1996), ensuring that learning outcomes, activities, and assessments are consistently aligned: students learn what is intended, and assessments both measure and shape learning. Where one assessment cannot cover all Intended Learning Outcomes (ILOs) complementary forms could be used (e.g., written exams plus lab reports). Module descriptions map ILOs to assessments.

In specific contexts, such as asynchronous online modules or courses emphasizing student engagement, Module Achievements or other types of formative assessments may support competence-oriented assessment.

Student feedback, embedded in the Quality Assurance System (QAS), systematically monitors workload, competence orientation, and alignment of ILOs and assessments. Student surveys and feedback are regulated in the Policy for student surveys and evaluations.

3.3 Degree

Upon successful completion of the program, students are awarded a Master of Science (M.Sc.) degree in Advanced Software Technology.

3.4 Graduation Requirements

In order to graduate, students need to obtain 120 CP. In addition, the following graduation requirements apply:

- In each module, students need to obtain a minimum amount of CP as indicated in chapter 2 of this handbook.
- Students need to complete all mandatory components of the program as indicated in chapter 2 of this handbook.

4 Advanced Software Technology Modules

4.1.1 Research Seminar

Module Name	Research Seminar
Module Code	2025-MAST-115
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory

Student Workload	
Independent Study	104
Research Group Meetings	21
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Research Seminar	MAST-115-A	Seminar	5

Module Description

The Research Seminar aims to equip students with the skills necessary to work with scientific literature, critically evaluate research findings, and present their insights. Students will select, read, and analyze scientific papers related to their field of study. They will then prepare and give presentations to their peers and discuss the topic. This process will help students refine their research interests and select a suitable thesis topic. The seminar will also introduce the best practices in academic presentation, argumentation, and peer feedback.

Usability and Relationship to other Modules

This module serves as a preparatory course for the Master thesis. It gives students the skills and confidence to engage in independent research.

Recommended Knowledge

Students should have foundational knowledge in the fields of their study, such as machine learning, programming languages, and software engineering.

Intended Learning Outcomes

No	Competence	ILO
1	Analyze	Analyze and interpret scientific papers critically.
2	Develop	Develop and deliver clear, well-structured academic presentations.
3	Engage	Engage in academic discussions and provide constructive feedback.

Indicative Literature

- Keshav, S. (2007). How to Read a Paper. ACM SIGCOMM Computer Communication Review, 37(3), 83-84.
- Selected recent research papers from top journals and conferences in the relevant field.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Research Seminar	Presentation	30 minutes	100	45%	1-3

Module Achievement

4.1.2 Programming Languages in Software Development

Module Name	Programming Languages in Software Development
Module Code	2025-MAST-102
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Timofey Bryksin

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory

Student Workload	
Exam Preparation	20
Independent Study	70
Lecture	17.5
Tutorial	17.5
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Programming Languages in Software Development	MAST-102-A	Lecture/Tutorial	5

Module Description

The module aims to provide students comprehensive understanding of the different types of programming languages and their characteristics, to familiarize them with the syntax and semantics of a variety of programming languages, including low-level, high-level, functional, logic, concurrent, parallel, scripting, and domain-specific languages, to teach students how to analyze and compare different programming languages, and to understand the trade-offs between different language features, to train students to use different programming languages for different types of software development tasks, such as web development, data science, and mobile app development, to develop students' problem-solving skills by applying the programming languages to solve real-world problems.

Content:

- Overview of programming languages: history, classification, and trends.

- Low-level languages: assembly, machine code, and C.

- High-level languages: Java, C#, Python, JavaScript, and Kotlin.

- Functional languages: Haskell, Lisp, and Scala.

- Logic and constraint programming languages: Prolog, and MiniZinc.

- Concurrent and parallel programming languages: Erlang, and Go.

- Scripting languages: Perl, Ruby, and Shell.

- Domain-specific languages: SQL, and XML.

Recommended Knowledge

Before taking the course, it's important to have a solid understanding of at least one programming language, as the course will cover a wide range of languages and paradigms.

Usability and Relationship to other Modules

The course content is designed to provide students with a comprehensive understanding of different programming languages, including low-level, high-level, functional, logic, concurrent, parallel, scripting, and domain-specific languages. It covers the history, classification and trends of programming languages. This would give students the ability to analyze and compare different languages based on their characteristics, and choose the appropriate language for a given task.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the history and classification of programming languages, and be able to analyze and compare different languages based on their characteristics.
2	Write	Write, read, and understand code written in a variety of programming languages, including low-level, high-level, functional, logic, concurrent, parallel, scripting, and domain-specific languages.
3	Use	Use different programming languages for different types of software development tasks, such as web development, data science, and mobile app development.
4	Evaluate	Evaluate the trade-offs between different language features and choose the appropriate language for a given task.
5	Apply	Apply their knowledge of programming languages to solve real-world problems, and develop their problem-solving skills.

Indicative Literature

- Carl A. Gunter: "Introduction to the Theory of Programming Languages", Cambridge University Press, 1996.
- David A. Watt and Deryck F. Brown: "Programming Languages and Paradigms", Pearson, 2008.
- Michael L. Scott: "Programming Language Pragmatics", Morgan Kaufman Publishers, 2009.
- Robert W. Sebesta: "Concepts of Programming Languages", Addison-Wesley, 2010.
- Terrence W. Pratt and Marvin V. Zelkowitz: "Programming Languages: Design and Implementation", Prentice Hall, 2004.

Entry Requirements

Prerequisites	None

Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Programming Languages in Software Development	Program Code		100	45%	1-5

Module Achievement

4.1.3 Machine Learning Overview

Module Name	Machine Learning Overview
Module Code	2025-MAST-114
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective

Student Workload	
Interactive Learning	35
Independent Study	90
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Machine Learning Overview	MAST-114-A	Lecture	5

Module Description

This course provides a concise yet comprehensive overview of machine learning (ML) as a tool for solving practical problems in various computer science domains. It is designed for students with the foundational knowledge of ML who wish to practice in applications and deepen their understanding. The course emphasizes the practical usage of ML frameworks and tools, bridging the gap between theoretical concepts and real-world problem-solving.

Content:

- Overview of machine learning as a discipline. Quick review of foundational ML concepts (regression, classification, clustering) and ML workflows.

- Supervised Learning

- Unsupervised Learning and Dimensionality Reduction

- Deep Learning Basics

- Application in Natural Language Processing

- Applications in Computer Vision

- Evaluation and Model Tuning

Recommended Knowledge

This module will shortly introduce all core knowledge in machine learning / statistical learning at the undergraduate level. However, it is recommended to be familiar with the basic concepts of machine learning, such as supervised and unsupervised learning, and the basic types of models and algorithms used in machine learning.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand and explain the core principles behind supervised and unsupervised techniques.
2	Use	Use ML frameworks and tools.
3	Apply	Apply ML methods to solve domain-specific problems, such as in data analytics, computer vision, or natural language processing.

Indicative Literature

- S. Shalev-Shwartz, Shai Ben-David: Understanding Machine Learning, Cambridge University Press, 2014. C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
- T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edition, Springer, 2008.
- T.M. Mitchell, Machine Learning, Mc Graw Hill India, 2017.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Machine Learning Overview	Program Code		100	45%	1-3

Module Achievement

4.1.4 Development Ecosystem

Module Name	Development Ecosystem
Module Code	2025-MAST-101
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Timofey Bryksin

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	70
Lecture and Tutorial	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Development Ecosystem	MAST-101-A	Lecture/Tutorial	5

Module Description

A programming language is only the first tool you need to develop applications. After knowing the syntax and the execution environment come tooling and essential libraries. In the end, to develop a non-trivial and practical application one should know a lot about the programming language ecosystem. This course covers some software development practices in Kotlin and some must-know libraries and tools for Kotlin.

Content:

- Gradle

- Testing

- Profiling

- DSL

- Networking in JVM

- Ktor

- Reflection

- Data Science

- Interoperability

- Annotations

- IntelliJ Platform SDK

- Compose

Recommended Knowledge

- Before diving into the ecosystem, it's important to have a solid understanding of the Kotlin language itself. You can start by reading through the official Kotlin documentation and working through some of the tutorials and examples provided there.

- Expected to have practical knowledge of everything described in Kotlin documentation (https://kotlinlang.org/docs/) up to Annotations

Usability and Relationship to other Modules

Kotlin is a general-purpose programming language that is designed to be fully interoperable with Java. This means that it can be used in a wide variety of contexts, including web development, Android development, and server-side development. One of the main advantages of Kotlin is its improved readability and expressiveness over Java. It has a more compact and expressive syntax, which makes it easier to write and maintain code. Additionally, Kotlin has a number of features that make it more suitable for functional programming, such as support for lambda expressions and higher-order functions. Another advantage of Kotlin is that it is fully compatible with Java, which means that developers can easily integrate it into existing Java projects, and use Java libraries and frameworks with Kotlin. This also makes it easy for Java developers to start using Kotlin, as they can continue to use the tools and libraries that they are already familiar with. For Android development, Kotlin has become the preferred language for Android development by Google since 2019, and it is supported by Android Studio, the official IDE for Android development. This make the transition from Java to Kotlin very smooth.

Intended Learning Outcomes

No	Competence	ILO
1	Write	Write different Kotlin applications from scratch.
2	Use	Use Kotlin for web-development, data science, IntelliJ Platform plugins.
3	Deploy	Deploy and maintain Kotlin applications in production environments.
4	Understand	Understand deeply how the Kotlin compiler works and how Kotlin works with different platforms.

Indicative Literature

- Alexey Soshin: "Kotlin Cookbook", O'Reilly Media, 2018.
- Antonio Leiva: "Kotlin for Android Developers", Packt Publishing, 2017.
- Ashish Belagali and Hardik Trivedi: "Kotlin Blueprints", Packt Publishing, 2018.
- Dmitry Jemerov and Svetlana Isakova: "Kotlin in Action", Manning Publications, 2017.
- Stephen Samuel and Stefan Bocutiu: "Programming Kotlin", O'Reilly Media, 2018.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Development Ecosystem	Program Code		100	45%	1-4

Module Achievement

4.1.5 Architectural Strategy

Module Name	Architectural Strategy
Module Code	2025-MCSSE-SE-03
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Prof. Dr. Jürgen Schönwälder

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	2	Mandatory

Student Workload	
Lecture	35
Tutorial	35
Independent Study	55
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Architectural Strategy	MCSSE-SE-03	Lecture/Tutorial	5

Module Description

The course "Architectural Strategy" focuses on Software Architectures, the key element for systematically developing large and complex software systems. During the course, we study how to design, recover, analyze, and document Software Architectures and understand how the main design decisions comprising them influence the quality attributes of the resulting systems.

Students will know in the first session which assignments will be part of the portfolio examination.

Intended Learning Outcomes

No	**Competence**	**ILO**
1	Understand	Understand methods for designing large software systems.
2	Design	Design complex and large software systems using components and connectors.
3	Use	Use UML as modeling language to represent the main concepts of software systems.
4	Document	Document their main design decisions and motivate them in terms of quality attributes.

Indicative Literature

- C. Pautasso, Software Architecture, 2020 (Visual Lecture Notes).
- Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice. Addison Wesley 2013.
- R.N. Taylor, N. Medvidovic, E.M. Dashofy, Software Architecture: Foundations, Theory, and Practice, Wiley, January (2009).

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Architectural Strategy	Portfolio Assessment	Individual Assignments, Group Assignments	100	45%	1-4

Module Achievement

4.1.6 Static Program Analysis

Module Name	Static Program Analysis
Module Code	2025-MAST-205
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	70
Lecture	17.5
Tutorial	17.5
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Static Program Analysis	MAST-205-A	Lecture	2.5
Static Program Analysis Tutorial	MAST-205-B	Tutorial	2.5

Module Description

The module aims to provide students with a comprehensive understanding of the kinds of program analysis and their applications; to familiarize students with the techniques and algorithms used for type analysis, data- and control-flow analyses, intra- and interprocedural analyses, alias analysis, bounded model checking; to develop students' skills in using program analysis to detect bugs, optimize code and perform security analysis; to train students to use program analysis tools and frameworks such as Soot, LLVM, and Frama-C; to give students an opportunity to apply their knowledge of program analysis to solve real-world problems.

Content:

- Introduction to program analysis: Types of program analysis, applications, and challenges.

- Type analysis: Definition, kinds and algorithms.

- Monotone framework: Definition, kinds and algorithms.

- Interval analysis: Definition, kinds and algorithms.

- Path sensitive analysis: Definition, kinds and algorithms.

- Bounded model checking: Definition, kinds and algorithms.

- Interprocedural analysis: Definition, kinds and algorithms.

- Alias analysis: Definition, kinds and algorithms.

- Applications of program analysis: Bug detection, code optimization, and security analysis.

A single assessment type cannot sufficiently test all intended learning outcomes. The practical assessment evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

It is important to have a solid understanding of the concepts and techniques of software engineering and programming languages, as the course will cover program analysis techniques and how to use them to improve software quality and security. Understanding compilers, formal languages or semantics of programming languages would make parts of the course easier to grasp, but it is not a hard pre-requisite.

Usability and Relationship to other Modules

- This module belongs to the Software Engineering Track in the MSc AST.

- The course provides a comprehensive coverage of different types of program analysis, their applications and challenges. The course is suitable for students who want to learn about the different types of program analysis and how to use them to improve software quality and security. The course is also beneficial for students who want to pursue a career in software engineering, software testing or software security.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the different kinds of program analysis, their applications, and challenges.
2	Design	Design and implement program analysis algorithms for type analysis, data- and control-flow analyses, intra- and interprocedural analyses, alias analysis, bounded model checking.
3	Use	Use program analysis tools and frameworks such as Soot, LLVM, and Frama-C.
4	Understand	Understand the results of program analyses, and use them to improve software quality and security.
5	Apply	Apply program analysis techniques to solve real-world problems in the field of software engineering.

Indicative Literature

- "Principles of Program Analysis" by Hanne Riis Nielson, Flemming Nielson, Springer, 1999.
- “Introduction to Lattices and Order” by B.A. Davey, H.A. Priestley, Cambridge University Press, 2022.
- “Introduction to Static Analysis: An Abstract Interpretation Perspective” by Xavier Rival, Kwangkeun Yi, The MIT Press, 2020.

- "Value-Range Analysis of C Programs: Towards Proving the Absence of Buffer Overflow Vulnerabilities" by Axel Simon, Springer, 2008.
- "WYSINWYX: What You See Is Not What You Execute" by Gogul Balakrishnan, University Of Wisconsin–Madison, 2007.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Static Program Analysis	Oral Examination	45 minutes	50	45%	All theoretical ILOs of the module
Static Program Analysis Tutorial	Practical Assessment		50	45%	All practical ILOs of the module

Module Achievement

4.1.7 Quality Engineering

Module Name	Quality Engineering
Module Code	2025-MCSSE-SE-02
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Prof. Dr. Mauro Pezzé

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	1	Mandatory

Student Workload	
Lecture	35
Independent Study	90
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Quality Engineering	MCSSE-SE-02	Lecture	5

Module Description

Software quality is an essential part of the software development and cannot be guaranteed a-priori, but most be verified both during and after the development. This course introduces the main testing and analysis techniques that can be used to identify failures and verify the quality of software systems. The course introduces the general testing and analysis principles and the basic techniques, shows how to apply them to solve relevant quality problems, illustrates complementarities and differences among the different techniques, and presents the organization of a coherent quality process. The course provides the elements needed to understand principles, techniques and process that comprise the basic background of test designer, quality manager and project manager. At the end of the course, the students will be able to define and implement quality plans for complex software systems. The student will have the basic knowledge of a project and a quality manager.

Students will know in the first session which assignments will be part of the portfolio examination.

Recommended Knowledge

-Programming skills in an imperative language at CS bachelor level

- Algorithms and data structure at CS bachelor level

- Basic skills in software testing: structural testing, Junit

- Basic knowledge of software engineering and IDEs at CS bachelor level

- Discrete math at CS bachelor level

Intended Learning Outcomes

No	Competence	ILO
1	Manage	Manage a software quality process.
2	Select	Select and implement a suitable set of testing and analysis activities to certify the quality of software systems.
3	Understand	Understand the core principles of software testing and program analysis.
4	Master	Master the basic techniques underlying software testing and program analysis.
5	Choose	Choose the suitable approaches to address the different testing and analysis programs.
6	Design	Design and monitor a suitable quality process.

Indicative Literature

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Quality Engineering	Portfolio Assessment	(Individual Assignments, Group Assignments)	100	45%	1-6

Module Achievement

4.1.8 Deep Learning

Module Name	Deep Learning
Module Code	2025-MCSSE-AI-01
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Prof. Dr. Andreas Birk

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	1	Mandatory Elective
2025-DE-MSc Data Engineering	1	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	3	Mandatory Elective
2025-DE-MSc Data Engineering	3	Mandatory Elective
2025-PHDS-BSc Physics and Data Science	5	Mandatory Elective
2025-SDT-BSc Software, Data and Technology	5	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	70
Lecture	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Deep Learning	MCSSE-AI-01	Lecture	5

Module Description

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.

Recommended Knowledge

- 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 course.

- Strong knowledge and abilities in mathematics (linear algebra, calculus).

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.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand core techniques to train neural networks.
2	Select	Select from modern neural network architectures the most appropriate method (e.g. convolutional and recurrent neural networks) based on given input data.
3	Contrast	Contrast different recent unsupervised learning methods including autoencoders and generative adversarial networks.
4	Describe	Describe techniques in reinforcement learning.

Indicative Literature

- Aurélien Géron: Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, 2nd Edition, O’Reilly, 2019.
- Charu C. Aggarwal: Neural Networks and Deep Learning – A Textbook, Springer, 2018.
- Christopher M. Bishop: Pattern Recognition and Machine Learning, Springer, 2006.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville: Deep Learning, MIT Press, 2016.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Deep Learning	Written Examination	120 Minutes	100	45%	1-4

Module Achievement

4.1.9 Optimization Methods in Machine Learning

Module Name	Optimization Methods in Machine Learning
Module Code	2025-MAST-110
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Dr. Dmitry Kropotov

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	17.5
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Optimization Methods in Machine Learning	MAST-110-A	Lecture	2.5
Optimization Methods in Machine Learning – Tutorial	MAST-110-B	Tutorial	2.5

Module Description

Optimization methods underlie in solution of many problems in computer science. In machine learning, one needs to solve an optimization problem during fitting of some prediction model on data, and efficiency in solving such optimization problems directly influences practical applicability of the correspondent machine learning approaches. This module aims to provide students with theoretical knowledge and practical skills in understanding both classic and recent continuous optimization methods (including optimization of non-convex functions) and important features in application of such methods for solving machine learning and deep learning problems. The module supposes a detailed discussion of practical aspects in implementation and usage of optimization routines. As the result of the module students will be able to choose and customize optimization method with better fit for their applied problems.

Content:

- Basic optimization techniques: gradient descent and Newton method, their theoretical properties

- Practical optimization: conjugate gradient, Hessian-free optimization, Quasi-Newton optimization

- Constrained optimization: KKT theorem, primal and primal-dual methods

- Solving various optimization problems in practice: adversarial attacks for neural networks, Wasserstein distance minimization for GANs, precoding optimization for 5G cellular networks, etc.

- Convex analysis: convex sets and functions, conjugate functions and norms, projection and proximal operators

- Stochastic optimization for convex functions and neural networks

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Usability and Relationship to other Modules

- The ability to solve optimization problems and familiarity with optimization methods is fundamental for almost all advanced modules in artificial intelligence and data science. For example, the module "Reinforcement Learning" requires understanding of trust region second order optimization for TRPO algorithm and different concepts from convex analysis for GAIL algorithm. The modules "Bayesian Methods in Machine Learning" and "Deep Bayesian Models" actively use constrained optimization on simplexes, set of positively defined matrices, set of orthogonal matrices, etc. The module "Deep Learning" requires understanding of dual problems for minimizing f-divergences, Wasserstein distances, etc. This module provides students with solid understanding of optimization which is needed in many other modules of the MSc program and also for research purposes.

- This module belongs to the Data Science Track in the MSc AST

Recommended Knowledge

- Good knowledge of machine learning, calculus, numerical methods

- Good understanding of machine learning and deep learning methods and underlying optimization problems, such as linear and logistic regression, support vector machines, L1 and other sparsity-inducing norms, adversarial attacks, optimal transport, etc. Good understanding of numerical methods of linear algebra would be a big plus.

Intended Learning Outcomes

No	Competence	ILO
1	Solve	Solve analytically different constrained and unconstrained optimization problems.
2	Detect	Detect convexity of underlying functions and sets, detect standard classes of constrained optimizations, transform applied optimization problems to standard classes for further application of optimization solver.
3	Understand	Understand theoretical properties of different optimization methods, their pros and cons.
4	Understand	Understand underlying numerical methods of linear algebra and matrix computations better.
5	Implement	Implement basic and advanced optimization methods on their own.

Indicative Literature

- B. Amos, J. Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks, arXiv:1703.00443, 2017.
- J. Nocedal, S.J. Wright. Numerical Optimization, Springer, 2006.
- Nowozin et al. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. arXiv:1606.00709, 2016.
- S. Boyd, L. Vandenberghe. Convex Optimization, Cambridge University Press, 2004.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Optimization Methods in Machine Learning	Written Examination	60 Minutes	50	45%	All intended learning outcomes of the module excluding topics of theoretical assignements.
Optimization Methods in Machine Learning – Tutorial	Program Code		50	45%	All practical and remaining theoretical intended learning outcomes of the module.

Module Achievement

4.1.10 Machine Learning in Software Engineering

Module Name	Machine Learning in Software Engineering
Module Code	2025-MAST-203
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Timofey Bryksin

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	35
Lecture	35
Project	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Machine Learning in Software Engineering - Tutorial	MAST-203-B	Tutorial	2.5
Machine Learning in Software Engineering	MAST-203-A	Lecture	2.5

Module Description

Machine learning is actively used in a variety of areas, software engineering in this sense is no exception. This course offers for consideration one and a half dozen practical problems from the field of programming and software development, as well as the scope of machine learning to solve them: what data and methods are used for this, what difficulties arise, what is the current progress in these tasks and what are the problems in general now relevant in the field of machine learning in SE. The course deals with the most relevant scientific articles of recent years, and in order to receive an assessment, students must complete a group practical project on one of the proposed topics.

Content:

- machine learning problem statement

- using machine learning for prediction and estimation

- using machine learning for code synthesis problems

- using machine learning to optimize code architecture

- using machine learning to find duplicates

- using natural language processing techniques

- using machine learning to analyze code

A single assessment type cannot sufficiently test all intended learning outcomes. The practical assessment evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

- Fundamental concepts of machine learning such as supervised and unsupervised learning, and the different types of models and algorithms.

- Understanding of machine learning and deep learning approaches used for natural language processing.

- Experience in programming in Python.

Usability and Relationship to other Modules

Familiarity with basic concepts of machine learning and software engineering is fundamental for almost all advanced modules in artificial intelligence and software engineering. This module additionally introduces advanced concepts of machine learning applied to software engineering, such as applying machine learning techniques to software development, testing and maintenance that are needed in advanced AI and software engineering-oriented modules in the 2nd year of the MSc program, as well as for research purposes. This module belongs to the Data Science Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Know	Know the areas of expedient application of the machine learning method, including the development of software projects. Read their own and other people's code, and debug the program. Determine the appropriateness of applying machine learning methods for the selected task.
2	Know	Know the main approaches and methods of machine learning, understand their strengths and weaknesses, the limits of applicability. Able to measure the effectiveness of the constructed models.
3	Develop	Develop models and prototypes of applications for the selected task in common programming languages.
4	Formulate	Formulate an algorithm for solving a problem in the form of a sequence of actions based on machine learning methods. Implement algorithms for solving the selected problem in suitable programming languages and using appropriate libraries.

Indicative Literature

- "Applied Machine Learning for Software Engineering" by Markus Helfert and Michael Sheng, Springer, 2020.
- "Machine Learning for Software Developers" by David C. Anastasiu, Zoran Duric and Rishi Shah, O'Reilly Media, 2019.
- "Machine Learning for Software Engineers" by David C. Anastasiu and Zoran Duric, O'Reilly Media, 2018.
- "Machine Learning for Software Quality" by Juergen Rilling, Springer, 2020.
- "Machine Learning in Software Engineering" by Jörg Kienzle and Wojciech Cellary, Springer, 2018.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Machine Learning in Software Engineering - Tutorial	Practical Assessment		50	45%	All practical ILOs of the module
Machine Learning in Software Engineering	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module

Module Achievement

4.1.11 Machine Learning Applications

Module Name	Machine Learning Applications
Module Code	2025-MAST-202
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Kirill Krinkin

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	70
Lecture	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Machine Learning Applications	MAST-202-A	Lecture/Tutorial	5

Module Description

This module explores advanced topics in machine learning, with a focus on practical applications and emerging technologies. Students will study state-of-the-art approaches, such as generative AI and recommender systems, to gain insights into their design, implementation, and deployment. Emphasis is placed on tackling real-world challenges and critically analyzing recent advancements in ML applications. The course aims to prepare students to apply advanced ML techniques to diverse and complex problems across various domains.

Recommended Knowledge

Basic concepts of machine learning, such as supervised and unsupervised learning, and the different types of models and algorithms.

Intended Learning Outcomes

No	Competence	ILO
1	Choose	Choose appropriate algorithms for building models.
2	Design	Design and implement machine learning models for advanced applications.
3	Analyze	Analyze and evaluate the performance of complex ML systems in real-world contexts.

4	Assess	Assess recent research and developments in advanced ML topics critically.
5	Collaborate	Collaborate effectively in teams to tackle complex ML challenges.

Indicative Literature

- "Programming Collective Intelligence" by Toby Segaran, O'Reilly Media, 2007.
- "Recommender Systems" by Jannach, Dietmar and Zanker, Markus and Felfernig, Alexander and Friedrich, Gerhard and Loos, Peter. Springer, 2017.
- Selected recent papers from conferences such as NeurIPS, ICML, ICLR and ACM RecSys.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Machine Learning Applications	Program Code		100	45%	1-5

Module Achievement

4.1.12 Bayesian Methods in Machine Learning

Module Name	Bayesian Methods in Machine Learning
Module Code	2025-MAST-204
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Dmitry Vetrov

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	70
Lecture	17.5
Tutorial	17.5
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Bayesian Methods in Machine Learning - Tutorial	MAST-204-B	Tutorial	2.5
Bayesian Methods in Machine Learning	MAST-204-A	Lecture	2.5

Module Description

The module focuses on the application of Bayesian methods to machine learning, providing students with theoretical knowledge and practical skills to incorporate probabilistic modeling and Bayesian techniques in their machine learning projects. The course will cover key Bayesian concepts, Bayesian inference methods, the use of Bayesian approaches in various machine learning algorithms, and the advantages of Bayesian techniques in handling uncertainty and modeling complex data.

Content

- Introduction to Bayesian methods: Bayesian Inference, conjugate priors, exponential family of distributions, Bayesian model selection.

- Bayesian linear regression and classification: automatic choosing of relevant features.

- EM-algorithm for training models with latent variables

- Approximate Bayesian methods: Variational Inference, Markov Chain Monte Carlo (MCMC) methods, Gibbs sampling, and Metropolis Hastings algorithm.

- Bayesian approaches in various machine learning algorithms: Bayesian clustering, Bayesian topic modelling and Bayesian mixture models.

- Bayesian non-parametric methods: Gaussian processes, Dirichlet processes, and their applications in machine learning.

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

- Good understanding of the fundamental concepts of calculus, statistics, machine learning.

- Basic knowledge from optimization, machine learning, statistics.

Usability and Relationship to other Modules

- Familiarity with basic probability and statistics, as well as machine learning concepts, is fundamental for almost all advanced modules in artificial intelligence and data science. This module additionally introduces advanced concepts of Bayesian methods and their application in machine learning, which are needed in advanced AI and data science-oriented modules in the 2nd year of the MSc program and also for research purposes.

- This module belongs to the Data Science Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Apply	Apply main tools for building and training Bayesian models for classical ML problems.
2	Construct	Construct latent variable models and select the appropriate algorithms for working with them.
3	Analyze	Analyze pros and cons of approximate Bayesian inference algorithms.
4	Understand	Understand the motivation of Bayesian approach to ML and its distinctions from classical ML algorithms.

Indicative Literature

- Christopher M. Bishop: Pattern Recognition and Machine Learning. Springer, 2006.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville: Deep Learning. MIT Press, 2016.
- Kevin P. Murphy, “Probabilistic Machine Learning: Advanced Topics”, MIT Press, 2023.
- Murphy, K. P. Machine Learning: A Probabilistic Perspective. MIT Press, 2012.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Bayesian Methods in Machine Learning - Tutorial	Program Code		50	45%	All practical ILOs of the module
Bayesian Methods in Machine Learning	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module

Module Achievement

4.1.13 Deep Bayesian Models

Module Name	Deep Bayesian Models
Module Code	2025-MAST-109
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Dmitry Vetrov

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	17.5
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Deep Bayesian Models	MAST-109-A	Lecture	2.5
Deep Bayesian Models - Tutorial	MAST-109-B	Tutorial	2.5

Module Description

This course is devoted to Bayesian reasoning in application to deep learning models. Attendees would learn how to use probabilistic modeling to construct neural generative and discriminative models, how to model the uncertainty about the weights of neural networks and how to design extensions of existing deep models using their Bayesian reformulation. Selected open problems in the field of deep learning would also be discussed. The practical assignments will cover implementation of several modern Bayesian deep learning models. The learning objective of the course is to give students basic and advanced tools for inference and learning in complex probabilistic models involving deep neural networks, such as diffusion models and Bayesian neural networks.

Content:

- Scalable Bayesian Inference

- Sparsification of deep neural networks

- Variational Autoencoder (VAE) for generative modelling

- Diffusion models for generative modelling

- Generative Flow Networks

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

- Basic knowledge from optimization, deep learning, statistics.

- Good understanding of the fundamental concepts of deep learning and Bayesian inference.

Usability and Relationship to other Modules

- Familiarity with basic concepts of machine learning, probability, and statistics is fundamental for almost all advanced modules in artificial intelligence and data science. This module additionally introduces advanced concepts of deep learning, such as advanced architectures, optimization techniques, and generative models, that are needed in advanced AI and data science-oriented modules of the MSc program, as well as for research purposes.

- This module belongs to the Data Science Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Know	Knowledge about different approximate inference and learning techniques for probabilistic models.
2	Apply	Hands-on experience with modern probabilistic modifications of deep learning models.
3	Know	Knowledge about the necessary building blocks that allow to construct new probabilistic models, suitable for the desired problems.

Indicative Literature

- D. Kingma, M. Welling. Auto-Encoding Variational Bayes. arXiv:1312.6114, 2013.
- D. Molchanov et al. Variational Dropout Sparsifies Deep Neural Networks. ICML, 2017.
- E. Bengio et al. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. NeurIPS, 2021.
- J. Ho et al. Denoising Diffusion Probabilistic Models. arXiv:2006.11239, 2020.
- Kevin P. Murphy, “Probabilistic Machine Learning: Advanced Topics”, MIT Press, 2023.

Entry Requirements

Prerequisites	2025-MCSSE-AI-01 Deep Learning 2025-MAST-204 Bayesian Methods in Machine Learning

Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Deep Bayesian Models	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module
Deep Bayesian Models - Tutorial	Program Code		50	45%	All practical ILOs of the module

Module Achievement

4.1.14 Data Analytics

Module Name	Data Analytics
Module Code	2025-MDE-CO-02
Module ECTS	5
Program Owner	2025-DE-MSc (Data Engineering)
Module Coordinator	Prof. Dr. Adalbert F.X. Wilhelm

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	1	Mandatory Elective
2025-DE-MSc Data Engineering	1	Mandatory
2025-DSSB-MSc Data Science for Society and Business	1	Mandatory Elective
2025-MBA-120-MA MBA 120	1	Mandatory Elective
2025-MBA-60-MA MBA 60	1	Mandatory Elective
2025-MDDA-BSc Management, Decisions and Data Analytics	1	Mandatory Elective

Student Workload	
Independent Study	90
Lecture	17.5
Tutorial	17.5
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Data Analytics	MDE-CO-02	Lecture	5

Module Description

This module introduces concepts and methods of data analytics. The objective of the module is to present methods for gaining insight from data and drawing conclusions for analytical reasoning and decision-making. The module comprises a broad spectrum of methods for modelling and understanding complex datasets. Comprising both descriptive and predictive analytics, the standard portfolio of supervised and unsupervised learning techniques is introduced. Automatic analysis components, such as data transformation, aggregation, classification, clustering, and outlier detection, will be treated as an integral part of the analytics process.

As a central part of this module, students are introduced to the major concepts of statistical learning such as cross-validation, feature selection, and model evaluation. The course takes an applied approach and combines the theoretical foundation of data analytics with practical exposure to the data analysis process.

Recommended Knowledge

- Read the Syllabus.

- Take the free online course: Introduction to Data Science at https://cognitiveclass.ai/courses/data-science-101/

Usability and Relationship to other Modules

In this module students will learn concepts and various techniques for data analysis. They will be rigorously applied in MDE-CS-03 as well as in the applied projects MDE-DIS-02 and MDE-DIS-03, and typically also in the master thesis.

Intended Learning Outcomes

No	Competence	ILO
1	Explain	Explain advanced data analytics techniques in theory and application.
2	Apply	Apply data analytics methods to real-life problems using appropriate tools.
3	Evaluate	Evaluate and compare different data analytics algorithms and approaches.
4	Apply	Apply statistical concepts to evaluate data analytics results.

Indicative Literature

- A. Telea, Data Visualization: Principles and Practice, Wellesley, Mass.: AK Peters, 1st edition, 2008.(DV).
- G. James, D.Witten, T. Hastie, Rob Tibshirani: Introduction to Statistical Learning with R by Springer, 2013 (ISLR).
- M. Ward, G. Grinstein, D. Keim, Interactive Data Visualization: Foundations, Techniques, and Applications. AK Peters, 1st edition, 2010. (IDV)

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Data Analytics	Project Report	20 Pages	100	45%	1-4

Module Achievement

4.1.15 Reinforcement Learning

Module Name	Reinforcement Learning
Module Code	2025-MAST-212
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Dr. Dmitry Kropotov

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	17.5
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Reinforcement Learning	MAST-212-A	Lecture	2.5
Reinforcement Learning- Tutorial	MAST-212-B	Tutorial	2.5

Module Description

Unlike classic machine learning, in reinforcement learning training algorithm doesn’t have a dataset as an input. Instead, trial and error method is used: an agent should collect data by itself during interaction with outer environment and basing on this data simultaneously maximize its obtained response – reward. The module is focused on learning state-of-the-art reinforcement learning algorithms for different applied problems of discrete and continuous control based on combining classic theory with deep learning.

Content:

- Introduction to RL, Bellman equations

- Tabular RL algorithms: Cross Entropy method, Value/Policy Iteration, Q learning

- Value-based deep RL algorithms: Deep Q Network and its modifications

- Policy gradient deep RL algorithms: A2C, TRPO, PPO, DDPG, SAC

- Intrinsic motivation for environment exploration

- Learning policies from experts, Imitation learning

- Model-based RL algorithms for discrete environments: Monte Carlo Tree Search, algorithms AlphaZero and MuZero.

- Model-based RL algorithms for continuous environments: algorithms iLQR, Dreamer.

- Applications of RL algorithms: playing games, robot control, finding new computational algorithms, protein structure prediction, etc.

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Usability and Relationship to other Modules

- This module is devoted to solving the most challenging problems of modern artificial intelligence. The underlying algorithms and theoretical concepts in this module are heavily based on outcomes from other modules of the MSc program like "Deep Learning", "Optimization in Machine Learning", etc.

- This module belongs to the Data Science Track in the MSc AST

Recommended Knowledge

- Good knowledge of machine learning, deep learning and optimization, good skills in learning deep neural networks.

- Students are expected to have good skills in learning deep neural networks and a good understanding of machine learning and deep learning technologies. Good understanding of advanced optimization methods would be a big plus.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand theoretical foundations of reinforcement learning algorithms and differences between various RL approaches.
2	Implement	Implement different RL algorithms for various environments and experiment with them.
3	Understand	Understand a general concept of simultaneous training of world model and control algorithm.
4	Understand	Understand how to solve most challenging problems of modern artificial intelligence using RL algorithms.

Indicative Literature

- David Silver, et al. Mastering the game of go without human knowledge. Nature 550.7676 (2017): 354.
- Fawzi et al. Discovering faster matrix multiplication algorithms with reinforcement learning. Nature 610, 47–53 (2022).
- R.S. Sutton and A.G. Barto. Reinforcement Learning: An Introduction, MIT Press, 2018.
- Y. Li. Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274. 2017.

Entry Requirements

Prerequisites	2025-MCSSE-AI-01 Deep Learning
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Reinforcement Learning	Written Examination	60 Minutes	50	45%	All theoretical ILOs of the module
Reinforcement Learning-Tutorial	Program Code		50	45%	All practical ILOs of the module

Module Achievement

4.1.16 Large Scale Deep Learning Models

Module Name	Large Scale Deep Learning Models
Module Code	2025-MAST-213
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Dr. Dmitry Kropotov

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	17.5
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Large Scale Deep Learning Models	MAST-213-A	Lecture	2.5
Large Scale Deep Learning Models-Tutorial	MAST-213-B	Tutorial	2.5

Module Description

With the development of deep learning, the size of models has grown to billions of parameters, and training data can be measured in terabytes. This led to the fact that models began to receive new properties, for example, zero-shot learning for text models. This module is aimed to show how large-scale models are trained without using labeled datasets in many domains: vision, text, audio. We also will talk about the model's architectural designs that made the training possible, what hidden abilities larger models obtain after the training and how to adapt such models to downstream tasks without additional fine-tuning.

Content:

- Introduction to the course, classical pretext tasks for self-supervised learning.

- What to do with a pretrained model? Probing, Linear classifier, Fine-tuning.

- Contrastive learning for images. Mutual information, SimCLR, MoCo, BYOL, SimSiam, SwAV.

- Masked Image Modeling. DINO, BEiT, MAE, MaskFeat. Different approaches to improving contrastive learning.

- Overview of Transformer-based language models. GPT, BERT, XLNet, RoBERTa, ALBERT, MASS, BART, ELECTRA.

- Model pre-training for the source code domain. code2vec, code2seq, CodeBERT, GraphCodeBERT, CodeT5, Codex.

- Diffusion models for NLP. Theory introduction, Multinomial Diffusion, D3PM, Diffusion-LM, DiffuSeq.

- Self-supervised learning for audio. Introduction to audio processing. CPC, Wav2Vec 2.0, HUBERT, Multi-format contrastive learning, BYOL-A.

- Self-supervised learning for graphs. Intro to representation learning on graphs.

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Usability and Relationship to other Modules

- This module covers the most recent approaches for training deep neural network models. This module is based on outcomes from other MSc modules like "Deep Learning".

- This module belongs to the Data Science Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Learn	Learn about the history of the increasing size deep learning models.
2	Deepen	Deepen their understanding of internal structure of large-scale models.
3	Learn	Learn how to use pre-trained large-scale models to solve downstream tasks.
4	Train	Train their own large-scale model.

Indicative Literature

- Scientific papers and blogs on models and algorithms from content description.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Large Scale Deep Learning Models	Written Examination	60 Minutes	50	45%	All theoretical ILOs of

					the module
Large Scale Deep Learning Models- Tutorial	Program Code		50	45%	All practical ILOs of the module

Module Achievement

4.1.17 Industrial Machine Learning on Hadoop and Spark

Module Name	Industrial Machine Learning on Hadoop and Spark
Module Code	2025-MAST-113
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Andreas Birk

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective

Student Workload	
Lecture	17.5
Tutorial	17.5
Independent Study	90
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Hadoop and Spark Theory	MAST-113-A	Lecture	2.5
Industrial Machine Learning	MAST-113-B	Tutorial	2.5

Module Description

This course provides an in-depth exploration of industrial machine learning applications using Hadoop and Spark. Topics include:

- Hadoop: HDFS, MapReduce, YARN

- Spark: DataFrames, feature engineering, ML pipelines, and real-time streaming analytics

Emphasis is placed on practical, hands-on lab sessions and a final project designed to solve real-world industrial problems.

Intended Learning Outcomes

No	**Competence**	**ILO**
1	Explain	Explain the architecture and functionality of Hadoop HDFS and its role in big data processing.
2	Gain	Gain the ability to implement MapReduce and YARN-based workflows to solve practical problems.

3	Gain	Gain the ability to utilize Spark DataFrames and ML libraries to analyze and process large datasets for collaborative filtering, image/NLP processing.
4	Evaluate	Evaluate different distributed machine learning models and their industrial applications.
5	Design	Design and implement scalable machine learning pipelines using Hadoop and Spark.
6	Communicate	Communicate technical concepts effectively in both academic and professional settings.

Indicative Literature

- Damji, J.S., et al. Learning Spark: Lightning-Fast Data Analytics. O'Reilly Media; 2nd edition, 2020.
- White, T. Hadoop: The Definitive Guide. O'Reilly Media, 2015.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Hadoop and Spark Theory	Program Code	Continuous assessment throughout the semester	100	45%	1-6
Industrial Machine Learning					

Module Achievement

4.1.18 Big Data Software Engineering

Module Name	Big Data Software Engineering
Module Code	2025-MAST-103
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Timofey Bryksin

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	52.5
Lecture	17.5
Tutorial	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Big Data Software Engineering	MAST-103-A	Lecture/Tutorial	5

<u>Module Description</u>

The module aims to provide students with the principles of building a scalable distributed software system for storing and processing big amounts of data. The course will look at the production solutions where such principles are implemented and will try to write our own distributed key-value storage.

Content:

- Data partitioning/sharding

- Data replication

- Distributed data processing

- Consistency in distributed systems

Assignments assume writing code, tests, configuration files, doing peer code reviews, deploying code in a cloud environment and running benchmarks.

<u>Recommended Knowledge</u>

- Basic knowledge Kotlin or Java

- Before taking the course, it's important to have a solid understanding of the concepts and techniques of software engineering and data science, as the course will cover big data technologies and how to use them for data analysis and modeling. Minimal knowledge of Docker, PostgreSQL and basics of working with relational databases will be a big plus.

Usability and Relationship to other Modules

The module provides a comprehensive coverage of the tools and technologies used for storing, managing, and processing big data. It also covers the important topic of data quality, governance and security. The course is suitable for students who want to learn about the challenges and opportunities of big data and how to use the technologies to process and analyze big data. The course is also beneficial for students who want to pursue a career in data science, software engineering, or big data analytics.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the general principles and challenges of building a distributed data storage system.
2	Implement	Implement data partitioning, replication and consensus algorithms in their own systems.
3	Use	Use data partitioning, replication and consensus features of the existing database systems.

Indicative Literature

- "Big Data Analytics with R and Hadoop" by Vignesh Prajapati, Packt Publishing, 2016.
- "Big Data: A Revolution That Will Transform How We Live, Work, and Think" by Viktor Mayer-Schönberger and Kenneth Cukier, Houghton Mifflin Harcourt, 2013.
- "Big Data: Understanding How Data Powers Big Business" by Bernard Marr, John Wiley & Sons, 2015.
- "Data Management for Researchers: Organize, Maintain and Share Your Data for Research Success" by Kristin Briney, CreateSpace Independent Publishing Platform, 2017.
- "Real-Time Big Data Analytics: Emerging Architecture" by Tejaswini Mandar Jog, Apress, 2016.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Big Data Software Engineering	Program Code		100	45%	1-3

Module Achievement

4.1.19 Cryptography

Module Name	Cryptography
Module Code	2025-MCSSE-CYB-01
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Prof. Dr. Jürgen Schönwälder

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	1	Mandatory Elective

Student Workload	
Lecture	35
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Cryptography	MCSSE-CYB-01	Lecture	5

Module Description

Information security requires techniques to protect information and to secure communication. Cryptography studies the design of cryptographic algorithms that can ensure confidentiality, integrity, and authenticity of data and messages exchanged in a secure communication protocol or when data is stored. This module focuses on the mathematical and algorithmic foundations of cryptography, and it covers the application of basic primitives to solve common information security challenges. Students familiar with the foundations of cryptographic algorithms will be able to judge the applicability and limitations of different cryptographic algorithms.

Recommended Knowledge

Students are expected to have a solid mathematical foundation. Students should review basic concepts of number theory, probability theory, and complexity theory in preparation for this module.

Usability and Relationship to other Modules

- The module serves as the foundational module in the cyber security specialization in CSSE. Other modules related to cyber security build on this module.

- This module belongs to the Software Engineering Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the mathematical problems on which cryptographic algorithms are built.
2	Describe	Describe pseudo random number generators and pseudo random functions.
3	Evaluate	Evaluate the strengths, weaknesses, and the applicability of cryptographic algorithms.
4	Select	Select from a set of symmetric block cipher, message integrity, and authenticated encryption algorithms.
5	Contrast	Contrast different asymmetric ciphers (finite field based, elliptic curve based, lattice based, hash based).
6	Explain	Explain the notion of quantum resistant cryptographic algorithms.
7	Analyze	Analyze the properties of cryptographic protocols such as key exchange mechanisms.
8	Apply	Apply techniques to analyze cryptographic protocols and their implementations.
9	Explain	Explain homomorphic encryption schemes and differential privacy.

Indicative Literature

- Bruce Schneier: Applied Cryptography, 20th Anniversary Edition, Wiley, 2015.
- Dan Boneh, Victor Shoup: A Graduate Course in Applied Cryptography, version 0.5, online, 2020.
- Simon Singh: The Code Book: Science of Secrecy from Ancient Egypt to Quantum Cryptography, Anchor Books, 2000.
- Wm. Arthur Conklin, Gregory White: Principles of Computer Security, 5th Edition, McGraw-Hill, 2018.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Cryptography	Written Examination	120 Minutes	100	45%	1-9

Module Achievement

4.1.20 System Security

Module Name	System Security
Module Code	2025-MCSSE-CYB-02
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Prof. Dr. Jürgen Schönwälder

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	2	Mandatory Elective

Student Workload	
Lecture	35
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
System Security	MCSSE-CYB-02	Lecture	5

Module Description

This module focuses on system level security aspects of computing systems. The module starts with investigating attacks on the microarchitecture of computing systems, such as attacks to gain information from side channels targeting caches. It then introduces trusted execution environments that use hardware isolation mechanisms to provide protected storage for keys and to bootstrap the integrity of bootloaders and the loaded operating systems. Students learn about the different levels of isolation that can be achieved using various types of hypervisors or sandboxing mechanisms. Techniques that can be used to protect a system against misbehaving code and malware are introduced. Students will gain knowledge how protected data storage components can be provided at the system level and how systems can offer support for collections of (distributed) authentication mechanisms. Finally, the module will discusses how authorization mechanisms are realized in the different system software components and how they can be used to define effective security policies.

Recommended Knowledge

Students are expected to be familiar with program execution at the system and machine level. Students should have a good understanding of computer architecture and operating systems at the level of typical undergraduate modules covering these topics. Students who have not taken an undergraduate course on computer architecture or operating systems yet may consider taking a remedial course or an online course to obtain a fundamental understanding how computer systems function.

Usability and Relationship to other Modules

- The module serves as a mandatory elective module in the cyber security specialization. Parts of the module require an understanding of cryptographic algorithms.

- This module belongs to the Software Engineering Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Describe	Describe microarchitectural attacks and computer components and suitable counter measures.
2	Illustrate	Illustrate trusted execution environments and how they can be used to bootstrap security.
3	Compare	Compare the isolation achieved by hypervisors and operating system mechanisms.
4	Assess	Assess application layer isolation and sandboxing mechanisms.
5	Explain	Explain how systems can identify misbehaving code and protect themselves against malware.
6	Outline	Outline how protected data storage can be implemented.
7	Recommend	Recommend authentication methods suitable for different kinds of applications.
8	Compose	Compose authorization mechanisms to define effective security policies.

Indicative Literature

- Swarup Bhunia: Hardware Security: A Hands-on Learning Approach, Morgan Kaufmann, 2018.
- William Stallings, Lawrie Brown: Computer Security: Principles and Practice, 4th edition, Pearson, 2018.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
System Security	Written Examination	120 Minutes	100	45%	1-8

Module Achievement

4.1.21 Network Security

Module Name	Network Security
Module Code	2025-MCSSE-CYB-03
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Prof. Dr. Jürgen Schönwälder

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	3	Mandatory Elective

Student Workload	
Lecture	35
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Network Security	MCSSE-CYB-03	Lecture	5

Module Description

Computer networks such as the Internet connect millions of computing systems, enable a fast exchange of information, and provide the technological basis on which large parts of the modern online economy are built. Computer networks, however, also expose an infrastructure that can be used by criminals or nation states to attack computing systems, to control the flow of messages, or to distribute malicious programs to potentially large numbers of targeted systems. This module educates students about how computer networks can be used to obtain information about remote systems, to manipulate the flow of data traffic, to disrupt access to remote services, or to control malicious software using botnets and distributed command and control channels. The module also covers technologies that help to protect the integrity of computer networks and that provide generic security services that can be used by applications requiring secure communication.

Recommended Knowledge

Students are expected to have a general understanding of computer networks, as provided by typical undergraduate modules on computer networks. Students who have not taken an undergraduate course on computer networks yet may consider taking a remedial course or an online course to obtain a fundamental understanding how computer networks function.

Usability and Relationship to other Modules

- The module serves as a mandatory elective module in the cyber security specialization. It builds on the cryptography module, which provides the necessary knowledge of cryptographic primitives that are used to protect data exchanged over computer networks and to authenticate communicating peers.

- This module belongs to the Software Engineering Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Describe	Describe techniques to obtain information about networked computing systems.
2	Contrast	Contrast mechanisms in the different network protocol layers for traffic manipulation and redirection.
3	Explain	Explain how distributed denial of service attacks are executed and how botnets are constructed.
4	Evaluate	Evaluate security mechanisms such as firewalls and anomaly / intrusion detection systems.
5	Analyze	Analyze generic security protocols such as IPsec, TLS, SSH and how they have evolved.
6	Compare	Compare protocols aiming to secure the network infrastructure (name resolution, routing).
7	Evaluate	Evaluate the security properties of modern software-defined network architectures.
8	Design	Design scalable solutions for protecting communication in distributed applications.

Indicative Literature

- Chris McNab, Network Security Assessment, O'Reilly, 2017.
- James Forshaw: Attacking Network Protocols, A Hacker's Guide to Capture, Analysis, and Exploitation, no starch press, 2017.
- William Stallings: Cryptography and Network Security: Principles and Practice, 7th edition, Pearsons, 2018.

Entry Requirements

Prerequisites	2025-MCSSE-CYB-01 Cryptography
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Network Security	Written Examination	120 Minutes	100	45%	1-8

Module Achievement

4.1.22 IDE Development

Module Name	IDE Development
Module Code	2025-MAST-207
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Timofey Bryksin

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	70
Lecture	17.5
Tutorial	17.5
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
IDE Development	MAST-207-A	Lecture and Seminar	5

Module Description

The module is designed to introduce students to a modern approach to creating integrated development tools. The course covers the main IDE modules: lexer, parser, code analyzer, local and global caches, code navigation, code modification and refactorings. In addition to the theoretical review, students will gradually develop their own IDE during the course. The course was created and implemented with the support of JetBrains. It is an elective module .

Content:

- Development tools. An introduction to the history and architecture of the IDE

- Data structures for working with text. Text editor and document markup

- Virtual file system, the concepts of the PSI model and the design model

- Introduction to the theory of formal languages

- Lexical analysis

- Parsing, abstract syntax trees

- Semantic analysis, symbol tables and link resolution.

- Introduction to type systems and type inference.

- Introduction to static analysis.

- Abstract interpretation, control flow analysis and data flow analysis

- Interprocedural analysis and call graph

- Help with typing and code completion. Search and navigation through the code.

- Modification of the abstract syntax tree. Code generation based on the abstract syntax tree. Auto-formatting. Automatic refactoring.

- Debugger and debugging symbols, expression evaluation during debugging.

- Instrumentation, profiling and tracing

Recommended Knowledge

The students should have a strong foundation in programming concepts and practices.

Usability and Relationship to other Modules

- The course provides a comprehensive coverage of the tools and technologies used for developing integrated development environments (IDEs). This can include topics such as IDE architecture, plugin development, debugging, code refactoring, version control integration and software testing. The course is suitable for students who want to learn about the challenges and opportunities of IDE development and how to use the technologies to develop IDEs. The course is also beneficial for students who want to pursue a career in software development, software engineering or software testing.

- This module belongs to the Software Engineering Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Demonstrate	Demonstrate a thorough understanding of the algorithms, data structures, and methods underlying the operation of modern IDEs and static analysis tools.
2	Conduct	Conduct research in the field of IDE development by identifying, analyzing, and developing new specific algorithms necessary to solve problems that arise during the development process.
3	Apply	Apply practical skills to address applied problems that emerge during the development of an IDE, such as designing user interfaces, optimizing performance, and implementing advanced features.
4	Evaluate	Evaluate and compare various IDEs and static analysis tools, considering factors such as usability, efficiency, and extensibility.
5	Collaborate	Collaborate effectively with a team to design, implement, and refine an IDE or a static analysis tool, leveraging version control systems, project management tools, and communication skills.

Indicative Literature

- "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin.
- "Refactoring: Improving the Design of Existing Code" by Martin Fowler.
- "The Pragmatic Programmer: From Journeyman to Master" by Andrew Hunt and David Thomas.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
IDE Development	Program Code		100	45%	All theoretical ILOs of the module

Module Achievement

4.1.23 Advanced Functional Programming

Module Name	Advanced Functional Programming
Module Code	2025-MAST-104
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Anton Podkopaev

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	70
Lecture	17.5
Tutorial	17.5
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Advanced Functional Programming - Tutorial	MAST-104-B	Tutorial	2.5
Advanced Functional Programming	MAST-104-A	Lecture	2.5

Module Description

The module aims to provide students with a thorough understanding advanced concepts and design patterns in functional programming, interacting with the external world using functional programming, being able to profile and debug functional programs, understanding and implementing persistent data structures in functional programming, understanding the principles of functional programming and be able to use them to solve complex problems.

Content:

- Advanced functional programming concepts and design patterns
- Interacting with the external world in functional programming
- Profiling and debugging functional programs
- Persistent data structures and their implementation in functional programming
- Type-level programming and meta-programming in functional programming
- Hands-on experience with the Haskell programming language

A single assessment type cannot sufficiently test all intended learning outcomes. The program code evaluates practical skills, whereas the written examination assesses understanding of theoretical knowledge, core principles, and analytical reasoning.

Recommended Knowledge

The basics of functional programming, including higher-order functions, recursion, and immutability

Usability and Relationship to other Modules

Familiarity with the basics of functional programming, and the Haskell programming language is fundamental for almost all advanced modules in computer science that rely on functional programming. This course introduces advanced concepts of functional programming such as persistent data structures, type-level programming, and meta-programming, which are needed in advanced programming-oriented modules such as functional software design, functional programming languages, and formal verification. Understanding the principles of functional programming and the Haskell programming language will enable students to apply these concepts in various fields such as computer science, finance, and data analysis. Additionally, the course provides a solid ground to use functional programming principles in mainstream programming languages such as Scala, F#, or OCaml. This module belongs to the Programming Languages Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand advanced concepts and design patterns in functional programming, and be able to apply them to design and implement functional programs.
2	Interact	Interact with the external world using functional programming techniques, such as IO Monad, and other related concepts.
3	Profile	Profile and debug functional programs and identify performance bottlenecks.
4	Understand	Understand and implement persistent data structures in functional programming, such as persistent arrays and persistent linked lists.
5	Learn	Learn how to use type-level programming and meta-programming in functional programming, such as type-level programming with type families, GADT, and other related topics.
6	Develop	Develop hands-on experience with the Haskell programming language and be able to apply functional programming concepts in other mainstream programming languages.

Indicative Literature

- Hudak, Paul. "The Haskell school of expression: learning functional programming through multMedia." (1999).
- Löh, Andres. "Functional pearl: The monad-reader pattern." (2009).
- Marlow, Simon. Parallel and Concurrent Programming in Haskell. O'Reilly Media, Inc., 2013.
- O'Sullivan, Bryan, John Goerzen, and Don Stewart. Real World Haskell. O'Reilly Media, Inc., 2008.

- Röjemo, András, and Erik Hesselink. "Functional pearl: Implicit configurations." (2010).
- Thompson, Simon. Haskell: The Craft of Functional Programming. Addison-Wesley, 1999.
- Wadler, Philip, and Stephanie Weirich. "The essence of functional programming." (2002).

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Advanced Functional Programming - Tutorial	Program Code		50	45%	All practical ILOs of the module
Advanced Functional Programming	Written Examination	120 Minutes	50	45%	All theoretical ILOs of the module

Module Achievement

4.1.24 Virtual Machines in Compilers

Module Name	Virtual Machines in Compilers
Module Code	2025-MAST-106
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Kirill Krinkin

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective

Student Workload	
Exam Preparation	20
Independent Study	70
Lecture	17.5
Tutorial	17.5
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Virtual Machines in Compilers	MAST-106-A	Lecture/Tutorial	5

Module Description

Modern compilers often use an intermediate target such as a virtual machine (JVM, .NET) or an intermediate representation (LLVM). In this module, students will learn how to construct compilers that make use of such virtual machines. Considerable attention is paid to issues related to the theoretical foundations and practical methods for generating efficient code targeting virtual machines that takes into account the details of the machine in question.

Content:

- Introduction- Why do we need virtual machines?

- Structure of a compiler

- One frontend, multiple backends: intermediate representations in production compilers

- Virtual machine performance

- Kotlinc as a case study of a modern compiller

Recommended Knowledge

The basics of computer systems and operating systems.

Usability and Relationship to other Modules

- Familiarity with a mainstream programming language such as Kotlin. Having followed a basic course on compilers is a plus.

- This module belongs to the Programming Languages Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Know	Know the main examples of popular virtual machines; main considerations to keep in mind when targeting virtual machines; main features of the implementation of existing VMs.
2	Create	Create a compiler that targets a virtual machine.
3	Have	Have the skills (gain experience) of building secure and reliable virtual machines; application of implementation algorithms for JIT compilers, physical memory managers.

Indicative Literature

- “Programming for the Java Virtual Machine”, Joshua Engel, 1999.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Virtual Machines in Compilers	Program Code		100	45%	1-3

Module Achievement

4.1.25 Formal Verification

Module Name	Formal Verification
Module Code	2025-MAST-214
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective

Student Workload	
Class Attendance	17.5
Tutorial	17.5
Independent Study	70
Exam Preparation	20
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Formal Verification	MAST-214-A	Lecture/Tutorial	5

Module Description

Formal verification is used to prove stronger code correctness guarantees than can be done using most of type systems. In this module, students will learn how to use Dafny, a modern formal verification language, to reason about imperative code. Students will learn about Hoare logic, the theory underlying most formal verification, and will gain a deep understanding of how Dafny uses an SMT solver to help with verification.

Content:

- Introduction to Dafny: preconditions, postconditions, invariants

- Advanced Dafny: triggers, performance considerations

- SMT solvers: theory and implementation

- Hoare logic

Recommended Knowledge

Students should be familiar with an imperative programming language. A course on type theory is recommended.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the principles of Hoare logic and how these are used for the design of a verification language like Dafny.
2	Understand	Understand the role of SMT solvers in formal verification.
3	Use	Use functional requirements to formulate and verify specifications on code.
4	Debug	Debug verification failures by formulating lemmas, strengthening preconditions and invariants, and guiding the solver.

Indicative Literature

- Leino, K. Rustan M. “Program Proofs.”, MIT Press, 2023.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Formal Verification	Program Code		100	45%	1-4

Module Achievement

4.1.26 Dependent Types

Module Name	Dependent Types
Module Code	2025-MAST-209
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Anton Podkopaev

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective

Student Workload	
Independent Study	90
Lecture	17.5
Tutorial	17.5
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Dependent Types	MAST-209-A	Lecture/Tutorial	5

Module Description

In this course, we'll learn the basics of program verification, specification, and formal theorem proving, We will also talk about the theoretic foundation of dependently typed systems. By the end of the course, students will be able to formulate and prove correctness properties of functional programs, algorithms, and simple maths theorems.

Content:

- Simple types

- Subtypes and Recursive Types

- Polymorphic types

- Type systems of higher orders

Recommended Knowledge

To master the module students need the knowledge gained from studying the module s "Formal languages", and "Functional programming".

Usability and Relationship to other Modules

- Familiarity with basic concepts of programming languages and formal methods is fundamental for almost all advanced modules in computer science and software engineering. This module additionally introduces advanced concepts of type systems, type inference, and type-based program analysis that are needed in advanced programming languages-oriented modules in the 2nd year of the MSc program, as well as for research purposes.

- This module belongs to the Programming Languages Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the relationship between logic and functional programming.
2	Know	Know various type-theoretic constructions occurring in dependently typed languages.
3	Formulate	Formulate and prove simple theorems.
4	Prove	Prove correctness of various algorithms.

Indicative Literature

- "Advanced Topics in Types and Programming Languages" by Benjamin C Pierce MIT Press, 2005.
- "Introduction to the Theory of Programming Languages" by Michael JC Gordon Cambridge University Press, 1996.
- "Types and Programming Languages" by Benjamin C Pierce MIT Press, 2002.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Dependent Types	Practical Assessment		100	45%	1-4

Module Achievement

4.1.27 Type Theory

Module Name	Type Theory
Module Code	2025-MAST-210
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective

Student Workload	
Independent Study	90
Lecture/Tutorial	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Type Theory	MAST-210-A	Lecture/Tutorial	5

Module Description

The module aims to provide students with theoretical knowledge and practical skills in understanding the basic principles and concepts of homotopy type theory, developing the ability to express homotopy-theoretic concepts in the language of homotopy type theory, learning the relationship of homotopy type theory with logic, set theory, and group theory, understanding how homotopy type theory can be applied in programming languages, developing the ability to use homotopy type theory to prove theorems in geometry and topology.

Content:

- Introduction to type theory and its extensions
- Fundamentals of homotopy theory and its relationship to geometry and topology
- Logic, set theory, and group theory in the context of homotopy type theory
- Concepts from homotopy theory expressed in the language of homotopy type theory

Recommended Knowledge

Students are expected to be familiar with imperative and functional programming languages.

Usability and Relationship to other Modules

- Familiarity with statically typed programming languages, both imperative and functional, is expected. This module introduces a variety of topics related to type theory that will prepare students both for practical applications, and provide a baseline for courses such as formal verification and dependent type theory.

- This module belongs to the Programming Languages Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the fundamental concepts and principles of type theory and its relationship to logic, set theory, and category theory.
2	Express	Express concepts in the language of type theory and use this language to express properties of the type system.
3	Understand	Understand the connection between type theory and programming languages and be able to apply type theory in the context of programming languages.
4	Write	Write and understand typing derivations, type checkers, and type inference engines.
5	Understand	Understand the basic concepts of model theory and domain theory and how they relate to type theory.
6	Understand	Understand the basic concepts of functional programming and how they relate to type theory.
7	Understand	Understand how the concepts of type theory can be used to reason about and reason with the properties of programs and systems.
8	Communicate	Communicate effectively and express your understanding of type theory in written and oral form.

Indicative Literature

- Awodey, Steve. Category theory. Vol. 48. Oxford: Clarendon Press, 2006.
- Homotopy Type Theory: Univalent Foundations of Mathematics. The Univalent Foundations Program, Institute for Advanced Study, 2013.
- Martin-Löf, Per. "Intuitionistic type theory." Bibliopolis (1984): 343-441.
- Pierce, Benjamin. “Types and Programming Languages.” MIT Press, 2002.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Type Theory	Program Code		100	45%	1-8

Module Achievement

4.1.28 Category Theory for Programmers

Module Name	Category Theory for Programmers
Module Code	2025-MAST-211
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory Elective

Student Workload	
Independent Study	90
Lecture/Tutorial	35
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Category Theory for Programmers	MAST-211-A	Lecture/Tutorial	5

Module Description

The module is aimed at developing students' theoretical knowledge and practical skills related to the use of functional programming languages. The course introduces the basic concepts of category theory, such as category, functor and monad. Students will learn to understand commutative diagrams. The course will help you better understand modern programming languages such as Agda, Coq and Idris. To master the module , students need to have knowledge of set theory, algebra, and topology.

Content:

- Introduction to category theory and its basic structures
- Fundamental concepts and theorems of category theory
- Relationship with functional programming and type theory
- Introduction to toposes and their internal language

Recommended Knowledge

Good understanding of the fundamental concepts of mathematics, such as set theory, logic and functions.

Usability and Relationship to other Modules

- Familiarity with basic concepts of mathematics and programming is fundamental for almost all advanced modules in computer science and software engineering. This module additionally introduces advanced concepts of category theory and its application to functional programming and type systems that are needed in advanced programming languages-oriented modules in the 2nd year of the MSc program, as well as for research purposes.

- This module belongs to the Programming Languages Track in the MSc AST

Intended Learning Outcomes

No	Competence	ILO
1	Know	Know the basics of category theory.
2	Understand	Understand categorical models of lambda calculus and simple type theory.
3	Understand	Understand the relationship between, logic, type theory, and category theory.
4	Work	Work within the internal language of a topos.

Indicative Literature

- "Categories and Types in Logic Language and Physics" by Bob Coecke Aleks Kissinger and Mehrnoosh Sadrzadeh Cambridge University Press, 2018.
- "Categories for the Working Mathematician" by Saunders Mac Lane Springer, 1971.
- "Category Theory for Programmers" by Bartosz Milewski, self-published, 2018.
- "Conceptual Mathematics: A First Introduction to Categories" by F William Lawvere and Stephen Hoel Schanuel Cambridge University Press, 1997.
- "The Joy of Cats" by Barry Mazur and Emily Riehl American Mathematical Society, 2020.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Category Theory for Programmers	Written Examination	120 Minutes	100	45%	1-4

Module Achievement

4.1.29 Capstone Project I

Module Name	Capstone Project I
Module Code	2025-MCSSE-CAP-01
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Prof. Dr. Manuel Oriol

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	1	Mandatory Elective

Student Workload	
Lecture	35
Tutorial	35
Project (Group-Based and Independent Work)	55
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Capstone Project 1	MCSSE-CAP-01	Project	5

Module Description

This series of Capstone modules gives the possibility of experiencing knowledge and expertise learned in the master by a posteriori analysis, transformational adaptation and coherent planning hands-on practice. The series spans over three modules during which students develop a complete product from scratch. The project starts with an ideation process, creation of clickable demos and initial requirements. It continues with the practical creation of a software architecture and development of the solution. It then finishes with application of artificial intelligence and cybersecurity. During the project, students are going through various steps during which they are encouraged to talk directly to potential real-world customers and users, thus gathering an understanding of what real users and customers for their project might want.

The project is organized in tribes (20-30 people) in charge of exactly one project. The tribes are then further split in agile teams working with the advice of the instructors and the assistants (impersonating the business owners and product owners). The teams can be geographically distributed and work with an up-to-date environment supported with open-source IDEs and engineering tools. Few lectures indicate the best practices to follow and the interim goals. Periodic meetings with the instructor and teaching assistants steer the process towards the overall goal.

This instance is the first semester of the Capstone project that focuses on ideation and requirements elicitation.

Recommended Knowledge

- Programming skills in an imperative language at CS bachelor level.

- Algorithms and data structure at CS bachelor level

- Train and advance programming, read about agile development, watch videos on ideation processes and read books on team and teamwork.

Usability and Relationship to other Modules

It is highly recommended to take the three Capstone Project modules in their numerical order to gain the full experience of the project.

Intended Learning Outcomes

No	Competence	ILO
1	Create	Create and propose mocks.
2	Perform	Perform requirements elicitation.
3	Prototype	Prototype
4	Approach	Approach customers and users.
5	Specify	Specify user stories.
6	Organize	Organize themselves through collaborative tools.
7	Understand	Understand team dynamics and resolve most interpersonal issues.

Indicative Literature

- • Online resources on team dynamics: - https://www.challengeapplications.com/stages-of-team-development - https://agilescrumguide.com/blog/files/tag-5-stages-of-team-development.html
- Bertrand Meyer: Agile! The good, the Hype and the Ugly. Springer, 2024
- Patrick Lencioni: The Five Dysfunctions of a Team. Jossey-Bass, 2022.
- Timothy M. Franz: Group dynamics and Teams interventions. Wiley-Blackwell, 2012

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Capstone Project 1	Project Assessment		100	45%	1-7

Module Achievement

4.1.30 Capstone Project II

Module Name	Capstone Project II
Module Code	2025-MCSSE-CAP-02
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Prof. Dr. Manuel Oriol

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	2	Mandatory Elective

Student Workload	
Lecture	35
Tutorial	35
Project (Group-Based and Independent Work)	55
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Capstone Project II	2025-MCSSE-CAP-02	Project	5

Module Description

This series of courses gives the possibility of experiencing knowledge and expertise learned in the master by a posteriori analysis, transformational adaptation and coherent planning hands-on practice. The course series spans over three courses during which students develop a complete product from scratch. The project starts with an ideation process, creation of clickable demos and initial requirements. It continues with the practical creation of a software architecture and development of the solution. It then finishes with application of artificial intelligence and cybersecurity. During the project students are going through various steps during which they are encouraged to talk directly to potential real-world customers and users, thus gathering an understanding of what real users and customers for their project might want.

The project is organized in tribes (20-30 people) in charge of exactly one project. The tribes are then further split in agile teams working with the advice of the instructors and the assistants (impersonating the business owners and product owners). The teams can be geographically distributed and work with an up-to-date environment supported with open source IDEs and engineering tools. Few lectures indicate the best practices to follow and the interim goals. Periodic meetings with the instructor and teaching assistants steer the process towards the overall goal.

This instance is the second semester of the capstone project that focuses on architecture and base implementation.

Recommended Knowledge

- Programming skills in an imperative language at CS bachelor level.

- Algorithms and data structure at CS bachelor level

- Train and advance programming, read about agile development, watch videos on ideation processes and read books on team and teamwork.

Usability and Relationship to other Modules

It is highly recommended to take the three Capstone Project modules in their numerical order to gain the full experience of the project.

Intended Learning Outcomes

No	Competence	ILO
1	Describe	Describe and defend a software architecture.
2	Code	Code in groups
3	Code	Code as a large team.
4	Integrate	Integrate independent works.
5	Use	Use a source code versioning system.
6	Specify	Specify user stories
7	Hold	Hold practical discussions with stakeholders.
8	Organize	Organize themselves through collaborative tools.
9	Understand	Understand team dynamics and resolve most interpersonal issues.

Indicative Literature

- Bertrand Meyer: Agile! The good, the Hype and the Ugly. Springer, 2024
- Online resources on team dynamics: - https://www.challengeapplications.com/stages-of-team-development - https://agilescrumguide.com/blog/files/tag-5-stages-of-team-development.html
- Online resources on team dynamics: - https://www.challengeapplications.com/stages-of-team-development - https://agilescrumguide.com/blog/files/tag-5-stages-of-team-development.html
- Patrick Lencioni: The Five Dysfunctions of a Team. Jossey-Bass, 2022.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Capstone Project II	Project Assessment		100	45%	1-8

Module Achievement

4.1.31 Capstone Project III

Module Name	Capstone Project III
Module Code	2025-MCSSE-CAP-03
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Prof. Dr. Manuel Oriol

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective
2025-CSSE-MSc Computer Science & Software Engineering	3	Mandatory Elective

Student Workload	
Lecture	35
Tutorial	35
Project (Group-Based and Independent Work)	55
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Capstone Project III	MCSSE-CAP-03	Project	5

Module Description

This series of courses gives the possibility of experiencing knowledge and expertise learned in the master by aposteriori analysis, transformational adaptation and coherent planning hands-on practice. The course series spans over three courses during which students develop a complete product from scratch. The project starts with an ideation process, creation of clickable demos and initial requirements. It continues with the practical creation of a software architecture and development of the solution. It then finishes with application of artificial intelligence and cybersecurity. During the project students are going through various steps during which they are encouraged to talk directly to potential real-world customers and users, thus gathering an understanding of what real users and customers for their project might want.

The project is organized in tribes (20-30 people) in charge of exactly one project. The tribes are then further split in agile teams working with the advice of the instructors and the assistants (impersonating the business owners and product owners). The teams can be geographically distributed and work with an up-to-date environment supported with open source IDEs and engineering tools. Few lectures indicate the best practices to follow and the interim goals. Periodic meetings with the instructor and teaching assistants steer the process towards the overall goal.

This instance is the third semester of the Capstone Project that focuses on integrating artificial intelligence, cybersecurity, and develops best practices.

Recommended Knowledge

- Programming skills in an imperative language at CS bachelor level

- Algorithms and data structure at CS bachelor level

- Train and advance programming, read about agile development, watch videos on ideation processes and read books on team and teamwork.

Usability and Relationship to other Modules

It is highly recommended to take the three Capstone Project modules in their numerical order to gain the full experience of the project.

Intended Learning Outcomes

No	Competence	ILO
1	Know	Know practical cybersecurity.
2	Hold	Hold practical discussions with stakeholders.
3	Practice	Practice of machine learning.
4	Work	Work with continuous improvements tools.
5	Organize	Organize themselves through collaborative tools.
6	Understand	Understand team dynamics and resolve most interpersonal issues.

Indicative Literature

- Bertrand Meyer: Agile! The good, the Hype and the Ugly. Springer, 2024
- Online resources on team dynamics: - https://www.challengeapplications.com/stages-of-team-development - https://agilescrumguide.com/blog/files/tag-5-stages-of-team-development.html.
- Patrick Lencioni: The Five Dysfunctions of a Team. Jossey-Bass, 2022.
- Timothy M. Franz: Group dynamics and Teams interventions. Wiley-Blackwell, 2012

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Capstone Project III	Project Assessment		100	45%	1-6

Module Achievement

4.1.32 Technological Entrepreneurship 1

Module Name	Technological Entrepreneurship 1
Module Code	2025-MAST-111
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory Elective

Student Workload	
Class Attendance	35
Tutorial	35
Independent Study	55
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Technological Entrepreneurship 1	MAST-111-A	Lecture	5

Module Description

The goal of this course is to go together from a product idea to testing this hypothesis on real users. During the course, students must refine their idea and justify it, first of all, for themselves: evaluate competitors, evaluate the development, create a financial model, roughly understand their Go-to-market. Understand what a prototype or MVP is and whether someone needs it or whether it is worth abandoning the idea in this formulation (pivot).

In the first part of the course, students will have to choose an area of interest where they will formulate a hypothesis for testing - a startup idea. For this purpose, a number of workshops will be organized, where students will be told current news from different markets. These workshops will help students assess their strengths and desire to work in a particular topic. Also, these workshops will present unsolved problems and issues that can be taken as a basis and try to come up with a solution.

Also, during the workshops, students will practice the skill of coming up with ideas. According to the developed methodology, students will come up with different startup ideas and learn to quickly defend them.

The educational outcome of the first part of the program is the skill of rapid preliminary assessment and defense of one's idea. The outcome of the semester is startup ideas that students will develop in the next part of the program.

Usability and Relationship to other Modules

The educational outcome of the first part of the program is the skill of rapid preliminary assessment and defense of one's idea. The outcome of the semester is startup ideas that students will develop in the next part of the program.

Recommended Knowledge

-As preparation, start reading the press about startup news in America and Europe.

- View startups that applied to and/or just successfully completed the best global accelerators.

- View the top startups in the area of interest that have raised investments over the past year.

Intended Learning Outcomes

No	Competence	ILO
1	Formulate	Formulate a product hypothesis.
2	Defend	Defend your idea with evidence and clear reasoning.
3	Validate	Validate your hypothesis.
4	Demonstrate	Demonstrate a basic understanding of markets where solutions can be offered.
5	Navigate	Navigate market information quickly.
6	Collaborate	Collaborate effectively in teams-based activities.
7	Find	Find co-founder.
8	Pitch	Pitch the idea clearly and persuasively.

Indicative Literature

- Eric Ries: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.
- Jason Fried, David Heinemeier Hansson: Rework.
- Jim Collins: Good to Great: Why Some Companies Make the Leap...And Others Don't.
- Peter Thiel: Zero to One: Notes on Startups, or How to Build the Future.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Technological Entrepreneurship 1	Project Assessment		100	45%	1-8

Module Achievement

4.1.33 Technological Entrepreneurship 2

Module Name	Technological Entrepreneurship 2
Module Code	2025-MAST-112
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory Elective

Student Workload	
Class Attendance	35
Tutorial	35
Independent Study	55
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Technological Entrepreneurship 2	MAST-112-A	Lecture	5

Module Description

The goal of this course is to go together from a product idea to testing this hypothesis on real users. During the course, students must refine their idea and justify it, first of all, for themselves: evaluate competitors, evaluate the development, create a financial model, roughly understand their Go-to-market. Understand what a prototype or MVP is and whether someone needs it or whether it is worth abandoning the idea in this formulation (pivot). To get to the second semester, students must pass the project idea defense stage.

The task of the second semester is to test the hypothesis for its validity. First, students will have to understand who their user or customer of the product is. Then, assess the market size and understand to what extent existing solutions cover the pain of users. Understand what the competitive advantage will be and why their team will be able to implement this idea.

Students will have to conduct research on the market and their users. Learn to collect feedback and quickly change their hypothesis, according to feedback from the market. Students will also need to come up with the cheapest way to test this hypothesis. Understand what the MVP can be and how to assemble this minimum product in no more than 3-4 working weeks.

Students will also have to develop a financial and business model. Understand whether investments will be needed to implement the project and the payback period. Students will have to defend the Go To Market strategy and try to make the first sales or pre-sales.

At the end of the course, students will have to learn how to design a pitch deck. They will learn how to make a good presentation and learn how to talk about their project.

Intended Learning Outcomes

No	Competence	ILO
1	Explain	Explain a Good product and how your startup take off.
2	Analyze	Analyze the market size and develop the right strategy for a startup.
3	Evaluate	Evaluate who your user really is and whether they have a demand for such a product.
4	Determine	Determine how to quickly test hypotheses and what role a prototype plays in a startup.
5	Describe	Describe how to build a business and a financial model for a startup.
6	Demonstrate	Demonstrate the ability to start making your first sales.
7	Communicate	Communicate how to do marketing without a budget or with a very small budget.
8	Communicate	Communicate how to develop a startup during the first years.
9	Communicate	Communicate how to organize workflow at startup team.
10	Articulate	Articulate where to secure investments and when they are not needed at all.
11	Learn	Learn how to make a pitch deck that will convince investors.
12	Understand	Understand how to communicate with a co-founder and when to hire a team.

Indicative Literature

- https://jetbrains.notion.site/Startup-School-2024-Guide-57249773057743b487459afaeb98ae97

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Technological Entrepreneurship 2	Project Assessment		100	45%	1-12

Module Achievement

4.1.34 Internship

Module Name	Internship
Module Code	2025-MAST-INT-01
Module ECTS	10
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective

Student Workload	
Internship	230
Report Preparation	20
Total Hours	**250**

Module Components	**Number**	**Type**	**CP**
Internship	MAST-INT-01-A	Internship	10

Module Description

The internship module provides students with the opportunity to gain hands-on experience in an industrial or applied research setting relevant to their field of study. Its primary goals are to explore potential directions for the Master's thesis, apply and deepen skills acquired in coursework, and develop a better understanding of real-world challenges in software engineering and computer science. A minimum of 230 working hours (i.e., approx. 6 weeks of full-time occupation) is required for the successful completion of this module. An internship should be approved by the Study Program Coordinator (SPC) based on a prior evaluation of the planned tasks. It is typically scheduled during the summer between the second and third semesters or the third semester. Students submit a short reflective report, connecting their internship experience with their academic learning and professional goals. At the end of the course, students will have to learn how to design a pitch deck. They will learn how to make a good presentation and learn how to talk about their project.

Intended Learning Outcomes

No	**Competence**	**ILO**
1	Apply	Apply their skill, knowledge, and tools to real-world problems.
2	Demonstrate	Demonstrate professional work attitude and business etiquette.
3	Collaborate	Collaborate effectively in a professional environment.
4	Improve	Improve reporting skills.
5	Explain	Explain Real-world problems and suitable approaches in their field of study.
6	Engage	Engage ethically with academic or professional communities.

Indicative Literature

- https://jetbrains.notion.site/Startup-School-2024-Guide-57249773057743b487459afaeb98ae97

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Internship	Internship Report or Business Plan and Reflection	2000 words/ Completion: Pass/Fail	100	45%	1-2

Module Achievement

4.1.35 Research Project

Module Name	Research Project
Module Code	2025-MAST-201
Module ECTS	5
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	3	Mandatory Elective

Student Workload	
Project (Independent Work)	104
Research Group Meetings	21
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Research Project	MAST-201-A	Project	5

<u>Module Description</u>

The competencies and knowledge earned in the first two semesters are deepened by developing a small research project. Students will be exposed to state-of-the-art research with the goal of reproducing results of recent research papers or extending ideas presented in recent research papers. Students will learn how to organize and execute a research project and how to present the results in the format of a typical research paper. Students are expected to participate in the meetings of the research group in which they are doing their research projects. The module is conducted together with JetBrains which provides research topics for the students.

<u>Intended Learning Outcomes</u>

No	**Competence**	**ILO**
1	Understand	Understand state-of-the-art research papers in a chosen field of specialization.
2	Plan	Plan a research project to reproduce research results or to extend ideas of recent research results.
3	Explain	Explain research questions and choose suitable methodologies to address them.
4	Document	Document a research project in the style of a typical scientific paper.

Indicative Literature

- Recent publications provided by the research project supervisors.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Research Project	Project Report	5,000 Words	100	45%	1-4

Module Achievement

4.1.1 Master Thesis AST

Module Name	Master Thesis AST
Module Code	2025-MAST-300
Module ECTS	30
Program Owner	2025-AST-MSc (Advanced Software Technology)
Module Coordinator	Prof. Dr. Alexander Omelchenko

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	4	Mandatory

Student Workload	
Independent Study	750
Total Hours	**750**

Module Components	**Number**	**Type**	**CP**
Master Thesis AST	MAST-300-T	Thesis	30
	MAST-300-T		

Module Description

The aim of this module is to train students to motivate, design, carry out and document a research project in one of the areas represented by the research groups of the faculty of AST. Some familiarity with the requisite Advanced Software Technology techniques will typically have been acquired in one of the preceding Advanced Projects. The thesis topic is determined in mutual agreement with the module instructor. They may arise from the ongoing research in the instructorâ€™s own research group, but it is also possible for a student to adopt a topic of his/her own choice provided the instructor agrees to supervise it. The thesis work comprises the full cycle of a scientific research endeavor: (i) identifying a relevant open research question, (ii) carrying out a literature survey to put the planned work in its context and relate it to the state of the art (SoA), (iii) formulate a concrete research objective, (iv) design a research plan including a statement of criteria to evaluate the success of the project, (v) carry out the plan (with the possibility to change the original plan when motivated), (vi) document the results, (vii) analyze the results with respect to the SoA, the original objective, and the success criteria, and (viii) document all of this in a thesis report. All of this work should be done with as much self-guidance as can be reasonably expected. The instructor will likely give substantial guidance for (i) and (iii), whereas the other aspects will be addressed with larger degrees of self-guidance. A research proposal document summarizing (i) â€“ (iv) is expected as an interim result and milestone (target size: 10 pages). In the first weeks of the course, an intense taught tutorial on scientific working and writing is held. The subsequent weeks follow a seminar style where students present and discuss literature as well as their own results to date. The project consists of the proposal, a thesis report (target size: 30â€“60 pages, and an oral presentation at the end of the course.

Recommended Knowledge

- Read the Syllabus

- Proficiency in the area of the chosen thesis topic.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understanding, at a professional level, of a circumscribed segment of the hosting group's research area.
2	Able	Able to apply specific and selected AST techniques, as required for the project, at a professional level.
3		General professional skills
4	Design	Designing and carrying out the full cycle of a scientific research project in a professional manner.
5	Formulate	Formulating a research proposal such that that it could serve as a funding proposal.
6	Write	Writing a research thesis such that it could be submitted to a scientific publication venue, or as a project report to a funding agency or industrial client.
7	Present	Presentation of project results for specialists and non-specialists.

Indicative Literature

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Master Thesis AST	Thesis	30-60 pages	75	Ø 45% of component 1 and 2	
	Oral Examination	20 minutes	25		Mainly presentation of project results but the presenta

					tion touches all ILOs.

Module Achievement

5 Management Modules

5.1.1 Agile Product Development & Design

Module Name	Agile Product Development & Design
Module Code	2025-MCSSE-MGT-01
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Prof. Dr. Jürgen Schönwälder

Study Semester		
Program	**Semester**	**Status**
2025-CSSE-MSc Computer Science & Software Engineering	1	Mandatory
2025-AST-MSc Advanced Software Technology	3	Mandatory

Student Workload	
Lecture	80
Independent Study	45
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Agile Product Development & Design	MCSSE-MGT-01	Lecture	5

Module Description

This course is focused on key aspects of agile product and service development and design process.

State-of-the-art user centered design methods will be at the core of the course.

The overall goal of this module is to help managers without a business degree to learn, understand and practice agile customer- and data-driven innovation processes in the information age. This module helps students to understand today's real-life challenges in a complex world, with wicked problems and with multiple stakeholder interests, where unpredictable is common, and where managers need to focus on achieving goals rather than repetitive tasks.

Students learn to develop and present innovative user-centered and theory-oriented solutions for real-world challenges in an IT-driven world.

This course is strongly based on the agile paradigm of user-centeredness, user-centered design and the ideas of the Service Dominant Logic. Service-dominant (S-D) logic is a meta-theoretical framework for explaining value co-creation, through exchange, among configurations of actors.

Major challenges and concerns will be reflected:

- the role of the customer and data in a transformed business world

- new theories, concepts, and approaches (such as service dominant logic, customer integration, gamification, new service models)

- new methods and management techniques in (service) innovation (Design Thinking)

- new methods in handling business processes: (agile) business process management - BPM

- ethics and security issues.

The module will enable students to collaborate across disciplines with experts from various areas.

Intended Learning Outcomes

No	Competence	ILO
1	Develop	Develop practical knowledge and management skills, and mind sets to master the challenges from an agile business environment.
2	Understand	Understand (routine) business processes in various context and how to adapt business processes to an agile business environment (agile Business Process Management).
3	Summarize	Summarize and classify the new data- and customer-driven technologies in a business context.
4	Understand	Understand the ideas of the “service dominant logic” as a business opportunity, such as user-centricity, value in use, value in interaction, business service ecosystems.
5	Apply	Apply innovative creativity methods and processes for product and software development (Design Thinking).
6	Adapt	Adapt to a new working culture based on a user-centricity, empathy, and playful testing of new products and services.

Indicative Literature

- Brenner, W., Uebernickel, F., Abrell, T. (2016). Design Thinking as Mindset, Process, and Toolbox, in: Brenner, W., Uebernickel, F. (Eds.), Design Thinking for Innovation. Springer International Publishing, pp. 3–21. https://doi.org/10.1007/978-3-319-26100-3_1. Brown, T. (2008). Design Thinking. Harvard Business Review. 86, 84–92. Available at: https://hbr.org/2008/06/design-thinking.
- Daniel Paschek, D., Frank Rennung, F., Trusculescu, A., Draghici,A. (2016). Corporate Development with Agile Business Process Modeling as a Key Success Factor, Procedia Computer Science, Vol 100, Pages 1168-1175, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.09.273.
- Vargo, S.L., & Lusch, R. (2004). Evolving to a New Dominant Logic for Marketing. Journal of Marketing, Vol. 68(1), 1 – 17. Vargo SL, Akaka MA, Vaughan CM. (2017). Conceptualizing Value: A Service-ecosystem View. Journal of Creating Value. 3(2):117-124. https://doi.org/10.1177%2F2394964317732861. Lusch, R.F., Nambisan, S. (2015). Service Innovation: A Service-Dominant Logic Perspective. MIS Quarterly. Vol. 39 No.1 , pp. 155-175. https://doi.org/10.25300/MISQ/2015/39.1.07.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Agile Product Development & Design	Presentation	30 minutes	100	45%	1-6

Module Achievement

5.1.2 Product Innovation & Marketing

Module Name	Product Innovation & Marketing
Module Code	2025-MCSSE-MGT-02
Module ECTS	5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Dr. PingPing Meckel

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	2	Mandatory
2025-CSSE-MSc Computer Science & Software Engineering	2	Mandatory

Student Workload	
Lecture	80
Independent Study	45
Total Hours	**125**

Module Components	**Number**	**Type**	**CP**
Product Innovation & Marketing	MCSSE-MGT-02	Lecture	5

Module Description

This course focuses on key strategic aspects of the innovation and commercialization process. The course draws on insights from a variety of fields – in particular, product management, innovation, marketing, and strategic management – in order to (i) develop a holistic, state-of-the art understanding of this process, (ii) to nurture the underlying mindset that spans technology and market elements, and (iii) to provide students with concrete tools that help them in navigating the journey from product idea to market success. The course will take both the perspective of established companies as well as of new ventures.

Intended Learning Outcomes

No	**Competence**	**ILO**
1	Understand	Understand the innovation process, particularly in technology domains.
2	Understand	Understand the commercialization process, particularly in technology domains.
3	Analyze	Analyze how value can be created and appropriated through innovation.
4	Understand	Understand and apply tools, methods and concepts to manage the commercialization process.

Indicative Literature

- Kotler, P. et al. (2024). Principles of Marketing, Global Edition. 19th ed. Harlow: Pearson Education Limited.
- Schilling, M.A. (2019). Strategic Management of Technological Innovation. McGraw-Hill.
- Tidd, J. and Bessant, J. (2021). Managing Innovation: Integrating Technological, Market and Organizational Change. 7th ed. Hoboken: Wiley.

Entry Requirements

Prerequisites	2025-MCSSE-LAS-01 Entrepreneurship and Intrapreneurship
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	**Examination Type**	**Duration or Length**	**Weight (%)**	**Minimum for Pass**	**ILOs**
Product Innovation & Marketing	Presentation	30 minutes	100	45%	1-4

Module Achievement

5.1.3 Entrepreneurship and Intrapreneurship

Module Name	Entrepreneurship and Intrapreneurship
Module Code	2025-MCSSE-LAS-01
Module ECTS	2.5
Program Owner	2025-CSSE-MSc (Computer Science & Software Engineering)
Module Coordinator	Dr. PingPing Meckel

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory
2025-CSSE-MSc Computer Science & Software Engineering	1	Mandatory

Student Workload	
Lecture	17.5
Independent Study	45
Total Hours	**62.5**

Module Components	**Number**	**Type**	**CP**
Entrepreneurship and Intrapreneurship	MCSSE-LAS-01	Lecture	2.5

Module Description

The module introduces students to the themes which are relevant to clearly develop corporate innovation and entrepreneurship as an activity. It introduces entrepreneurial thinking styles that are important to develop radical forms of innovation in companies. This is about a way of thinking, reasoning and acting that is opportunity obsessed and holistic in approach. It is first and foremost a process that has an intention to create, enhance, realize, and renew value, not just for owners, but for all participants and stakeholders in either a new or existing organization. Today, entrepreneurship has evolved beyond the classic start-up notion to include companies and organizations of all types, old and new; small and large; fast and slow growing; private, not-for-profit, and public.

This focus on “entrepreneurship as a process” has become a fundamental part for three main reasons. The first is the growing recognition of the critical importance of entrepreneurial activities in the economy and the society at large. As such, having an insight in the specific challenges and solutions that characterize entrepreneurship has broader implications for any 21st century graduate. The second reason is that many graduates eventually find themselves occupying a position as entrepreneur, or are associated with one as their financier, partner, supplier or customer. This requires an action-oriented approach and approaching the phenomenon from multiple angles. Finally, given the specific challenges entrepreneurs often face in terms of uncertainty and resource scarcity, solutions applied by expert entrepreneurs can be of value to any professional that finds him/herself in similar situations in organizations seeking growth, renewal or even survival.

The module focuses on the tasks and skills that entrepreneurs typically complete/use in their journey towards success. With this in mind, this module aims to provide students with insight into the approach entrepreneurs use to identify opportunities and build new ventures; the analytical skills that are needed to implement this approach; and the background knowledge and managerial skills that are needed for dealing with issues involved in starting, growing, and harnessing the value of new ventures. First and foremost, however, entrepreneurship is about action. Hence our approach is based on the primary objective of having students experience entrepreneurship.

This module is intentionally designed as a 2.5 CP module to enhance Employability by exposing students to a variety of professionally relevant topics rather than requiring in-depth specialization.

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand the essence of entrepreneurship.
2	Assess	Assess and develop a business case.
3	Analyze	Analyze and identify new venture opportunities in a more systematic way.
4	Understand	Understand the importance of a business model for new venture creation.
5	Evaluate	Evaluate the viability of a new venture idea.
6	Understand	Understand how to finance a new venture.
7	Create	Create and present a business case for a new venture.

Indicative Literature

- Greene, F. J. (2020). Entrepreneurship: Theory and Practice. London: Macmillan Education Ltd.
- Jones, O., Meckel, P., and Taylor, D. (2021). Creating Communities of Practice: Entrepreneurial Learning in a University-Based Incubator. Cham: Springer International Publishing AG. https://ebookcentral.proquest.com/lib/constructor-university/detail.action?docID=6467881.
- Rae, D. (2015). Opportunity-Centred Entrepreneurship. 2nd ed. London: Palgrave.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Entrepreneurship and Intrapreneurship	Presentation	30 minutes	100	45%	1-7

Module Achievement

5.1.4 Agile Leadership and Strategic Management

Module Name	Agile Leadership and Strategic Management
Module Code	2025-MCSSE-LAS-03
Module ECTS	2.5
Program Owner	2025-MBA-120-MA (MBA 120)
Module Coordinator	Prof. Dr. Adalbert F.X. Wilhelm

Study Semester		
Program	**Semester**	**Status**
2025-AST-MSc Advanced Software Technology	1	Mandatory
2025-MBA-120-MA MBA 120	1	Mandatory
2025-CSSE-MSc Computer Science & Software Engineering	3	Mandatory

Student Workload	
Lecture	17.5
Independent Study	45
Total Hours	**62.5**

Module Components	**Number**	**Type**	**CP**
Agile Leadership and Strategic Management	MCSSE-LAS-03	Lecture	2.5

<u>Module Description</u>

This module focuses on key strategic aspects of the leadership and strategy development processes, specifically strategic problems solving, alignment, engagement and coping with black swans and paradigm shifts. The module draws insights from a variety of fields such as business strategy, problem solving, strategic communication, strategic planning, and strategic resilience.

To build a holistic understanding, the module deals with the following topics:

- The strategic process: from analysis, definition, planning and evaluation

- Hypothesis driven problem solving

- Pyramid principle strategic communication

- Antifragile strategies

The module assessment will consist of three presentations. Students will know in the first session which topics need to be covered in their presentations.

This module is intentionally designed as a 2.5 CP module to enhance Employability by exposing students to a variety of professionally relevant topics rather than requiring in-depth specialization

Intended Learning Outcomes

No	Competence	ILO
1	Understand	Understand and analyze business strategies.
2	Understand	Understand and analyze strategic statements and levels of ambition.
3	Understand	Understand opportunities and threats on the external environment.
4	Evaluate	Evaluate sources of competitive advantage as well as strategic strengths and weaknesses.
5	Analyze	Analyze core challenges of agile leadership and strategy development.
6	Develop	Develop and communicate strategic initiatives.
7	Apply	Apply this knowledge to real-world strategic planning processes.

Indicative Literature

- Sola, D. & Couturier, J, 2013, How To Think Strategically, FT Publishing International.

Entry Requirements

Prerequisites	None
Co-requisites	None
Additional Remarks	None

Assessment and Completion

Module Component	Examination Type	Duration or Length	Weight (%)	Minimum for Pass	ILOs
Agile Leadership and Strategic Management	Presentation	30 minutes	100	45%	1-7

Module Achievement

6 Appendix

6.1 Intended Learning Outcomes Assessment-Matrix

MSc Advanced Software Technology					Programming Languages in Software Development	Quality Engineering	Development ecosystem	Deep Learning	Machine Learning Overview	Data Analytics	Architectural Strategy	Optimization Methods for Machine Learning	Machine Learning in Software Engineering	Static Program Analysis	Big Data Software Engineering	Research Seminar	Machine Learning Applications	Bayesian Methods in Machine Learning	Deep Bayesian Models	Reinforcement Learning	Large Scale Deep Learning Models	Cryptography	System Security	Network Security	IDE Development	Advanced Functional Programming	Industrial Machine Learning on Hadoop and Spark	Formal Verification	Virtual Machines in Compilers	Dependent Types	Type Theory	Category Theory for Programmers	Agile Product Development & Design	Product Innovation & Marketing	Agile Leadership and Strategic Management	Entrepreneurship & Intrapreneurship	Internship	Master's Thesis	Research Project	Capstone Project 1	Technological Entrepreneurship 1	Capstone Project 2	Technological Entrepreneurship 2	Capstone Project 3
Semester					1	1	1	1	1	1	2	2	2	2	2	2	3	1	2	3	3	1	2	3	1	1	2	3	3	3	3	3	3	2	3	3	3	4	3	1	1	2	2	3
Mandatory/ optional					m	me	me	me	me	me	me	me	me	me	me	m	me	me	me	me	me	me	me	me	me	me	me	me	me	me	me	me	m	me	me	me	me	m	me	me	me	me	me	me
Credits					5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	2,5	2,5	10	30	5	5	5	5	5	5
	ompetencie																																											
Program Learning Outcomes	**A**	**E**	**P**	**S**																																								
critically assess and creatively apply technological possibilities and innovations in the fields of data science, software development and programming languages	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x
critically assess and apply software engineering methodologies considering real life situations, organizations and industries	x	x			x	x	x	x	x	x	x			x	x	x						x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x
use, adapt and improve modern techniques in data science, such as deep learning, recommender systems, computer vision, and machine learning in software engineering	x	x						x	x	x		x	x		x	x	x	x	x	x	x						x										x	x	x	x	x	x	x	x
apply cross-disciplinary management methodologies to solve academic and professional problems in the context of software development and data science	x	x	x													x																	x	x	x	x	x	x	x	x	x	x	x	x
critically assess and integrate a consistent tool set of leadership abilities into a professional work environment	x	x	x													x																	x	x	x	x	x	x	x	x	x	x	x	x
plan, conduct and document small research projects in the context of data science, software development and programming languages	x	x	x			x	x	x	x						x	x																	x	x	x	x	x	x	x	x	x	x	x	x
independently research, document and present a scientific topic with appropriate language skills	x	x	x	x												x																	x	x	x	x	x	x	x	x	x	x	x	x
use scientific methods as appropriate in the field of data science and software engineering such as defining research questions, justifying methods, collecting, assessing and interpreting relevant information, and drawing scientifically-founded conclusions that consider social, scientific and ethical insights	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
develop and advance solutions to problems and arguments in their subject area and defend these in discussions with specialists and non-specialists		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
engage ethically with academic, professional and wider communities and to actively contribute to a sustainable future, reflecting and respecting different views		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
take responsibility for their own learning, personal and professional development and role in society, evaluating critical feedback and self-analysis		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
apply their knowledge and understanding of data science, software development, and programming languages to a professional context	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
take on responsibility in a diverse team		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
adhere to and defend ethical, scientific and professional standards	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
use and understand the Kotlin ecosystem	x				x		x							x		x						x	x	x	x					x	x	x					x	x	x	x	x	x	x	x
apply data analytics techniques	x								x	x			x		x	x	x	x																			x	x	x	x	x	x	x	x
understand and utilize agile product development and design methodologies		x	x	x												x																	x	x	x	x	x	x	x	x	x	x	x	x
understand and apply principles of quality engineering	x					x					x					x												x									x	x	x	x	x	x	x	x
Assessment Type																																												
Written examination								x				x	x					x	x	x	x	x	x	x	x		x			x		x	x													
Term paper																																												
Essay																																												
Project report										x																											x		x					
Poster presentation																																												
Laboratory Report																																												
Program code					x		x		x			x			x		x	x	x	x	x				x	x	x																	
Oral examination														x																														
Presentation																																	x	x	x	x								
Practical Assessments													x	x															x	x														
Project Assessments																																								x	x	x	x	x
Portfolio Assessments						x					x																																	
Master Thesis																																						x						
Module achievements																																												

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