University of Luxembourg

Faculty of Science, Technology and Medicine

Bachelor in Applied Information Technology – BINFO

https://binfo.uni.lu

Programme

Academic Year 2025-2026

Programme Director: A-Prof. Volker Müller

Version: 15.02.2026

Introduction

The **Bachelor in Applied Information Technology (BINFO)** at the University of Luxembourg is a dynamic and career-focused program designed to prepare students for success in today's fast-paced digital world. Combining cutting-edge knowledge with practical, real-world experience, BINFO equips graduates with the in-demand skills that top employers are looking for. Whether aiming for a career in the private or public sector, students are empowered to make an immediate impact and thrive in the evolving field of information technology.

The BINFO program offers a **well-balanced curriculum** that integrates both theoretical foundations and practical applications of information technology. Students build a strong grounding in key IT areas, including algorithms and data structures, databases, networks, software development, and mobile and web application programming. This comprehensive approach ensures they acquire the essential theoretical knowledge needed in the field.

What sets BINFO apart is its **strong emphasis on practical, real-world experience**. The program encourages students to think critically, solve complex problems, and apply their skills in industry-relevant contexts. By focusing on hands-on learning and real-world scenarios, BINFO equips students with practical competencies and adaptable knowledge tailored to their future careers - making it a distinctive alternative to traditional computer science programs. The defined program delivers a well-rounded and engaging learning experience by offering a comprehensive education in information technology. Through a balanced blend of theoretical foundations and practical applications, it equips students with the technical expertise and hands-on skills that are highly valued by employers, ensuring a smooth transition into the professional world across diverse sectors.

BINFO has a strong commitment to preparing students for the demands of the **European labor market**. The program goes beyond technical training by integrating key soft skills essential in today's workplace. Students benefit from targeted courses in entrepreneurship, the psychology of teamwork, and real-world team projects – designed to strengthen their communication, collaboration, and problem-solving abilities. As a result, BINFO graduates emerge as adaptable, well-rounded professionals ready to thrive in dynamic, multicultural environments.

To ensure meaningful hands-on experience, the BINFO program includes a

one-semester, IT-related practical project. This allows students to apply their knowledge and skills in real-world settings. Depending on their interests, students can choose an industry-focused placement within the IT department of a Luxembourg-based company or pursue a research-oriented project within the University's Computer Science department. This practical component not only sharpens their technical and problem-solving abilities but also offers valuable insights into their preferred career paths.

Another key strength of the program is its **human-centered and international approach**. BINFO is taught in a bilingual French/English curriculum and actively promotes student mobility through at least one semester abroad. This international exposure enhances language proficiency while fostering cross-cultural understanding and adaptability. The presence of faculty and students from diverse backgrounds further enriches the learning environment, preparing graduates to thrive in today's globalized, multicultural workforce.

Admission to the BINFO program is open to all candidates holding a university entrance qualification and demonstrating sufficient language proficiency. Applicants are expected to have at least B1 level in French and B2 level in English. A strong motivation to engage with mathematics and computer science—particularly in areas related to software development—is also essential.

To promote inclusivity, the program encourages applications from students with a strong background in mathematics, even if they do not yet meet the required language levels. Recognizing the critical role of language skills, BINFO offers dedicated support through the University's Language Center, which provides tailored courses to help students strengthen their French and English proficiency and reach the necessary language standards.

In April 2021, the BINFO programme has been awarded the **accreditation** for excellent quality and programme administration by the international accrediation agency ACQUIN (https://www.acquin.org).

Important Information

Legal Basis

The legal framework for all academic programmes at the University of Luxembourg is established by the following two official documents (available only in French):

- Loi modifiée du 27 juin 2018 ayant pour objet l'organisation de l'Université du Luxembourg,
- Règlement des études du 24 september 2024.

These documents define all the regulations and procedures that apply during the study. Both documents are available on the university website at https://www.uni.lu/en/about/official-documents/.

Mobility Semester

As per the law on the university, Art 36 (6), the BINFO program mandates at least one mobility semester for Bachelor students in a foreign country. The mobility semester in BINFO can be chosen in either Semester 3 or Semester 4, providing flexibility to accommodate students' preferences and circumstances.

During the mobility semester, the standard curriculum is replaced by a customized selection of courses offered by the host institution. These courses are chosen in close collaboration between the student and the study director prior to the start of the mobility period. This collaborative process ensures that the selected courses align with both the student's academic objectives and the requirements of the BINFO program.

For further administrative details on mobility, a dedicated page is available on the Moodle platform of the University of Luxembourg. This page provides comprehensive information and guidelines regarding the mobility semester, enabling students to access all the necessary details and resources in one centralized location.

Bachelor Project

In the final Bachelor project, conducted during the 6th semester, BINFO students have the opportunity to apply both the technical and interpersonal skills acquired throughout their studies. The project spans 12 to 15 weeks and must be closely related to the field of Information Technology.

The final project is intentionally flexible, allowing students to tailor their work to their interests and career goals. Projects may involve the development of new software or the enhancement of existing systems, such as building a prototype, creating a proof-of-concept, or implementing a specific software module. Students interested in hardware can pursue projects such as designing a prototype for a feasibility study, developing a controller, or creating a data acquisition device.

Alternatively, students may focus on the management aspects of IT projects, engaging in tasks like defining technical specifications, researching state-of-the-art technologies, or evaluating hardware and software options for future systems. This variety ensures that each student can undertake a project that aligns with their strengths, interests, and professional aspirations.

The project must reflect a significant and original contribution from the student. The three key elements of a Bachelor project are:

- The Bachelor project and its expected outcomes must be clearly defined before the start of the internship and be achievable within the 12–15-week timeframe.
- The student will be guided by a local supervisor from the host institution and supported by an academic supervisor from the university, who will provide expertise on the scientific aspects of the project.
- The Bachelor project culminates in a comprehensive thesis – a detailed written report documenting the entire process. This report should cover the technical background, all IT-related components (including problem analysis, design, implementation, and results), and will be evaluated by a review committee.

The Bachelor project can be done in two different variants:

- A **profession-oriented internship** takes place within a professional institution in Luxembourg, giving students the opportunity to apply and further develop their IT expertise and soft skills in a real-world environment. This option is strongly recommended for students aiming to enter the workforce immediately after graduation, as it offers valuable practical experience, sharpens technical abilities, and cultivates essential interpersonal skills. By doing so, it significantly enhances job market readiness and eases the transition from academic study to professional life.

- A **research-oriented Bachelor project** is carried out in collaboration with a research group at the University of Luxembourg. It emphasizes the application and expansion of the student's scientific IT expertise, focusing on critical thinking, data analysis, and research methodologies within the field of Information Technology. This option is particularly recommended for students intending to continue their academic journey with a Master's program, as it strengthens their preparation for advanced studies, enhances research skills, and nurtures a passion for academic inquiry in IT.

Contact Information

A-Prof. Volker Müller (Study Director)	Maison du Nombre, E03 0335-080, Belval Phone: (+352) 46 66 44 6751 Email: volker.muller@uni.lu
Sandra Rosin (Programme Secretary)	Maison du Savoir, E06 0625-040, Belval Phone: (+352) 46 66 44 4913 Email: sandra.rosin@uni.lu

Important Information on Data Protection

Information as provided by the data protection officer of the University:

Recording of class sessions in all BINFO courses or other academic activities without authorization is prohibited. Students who wish to record a session must request and obtain explicit permission from the instructor and any *students involved in advance. If permission is granted, the publication or distribution of instructional materials is not allowed unless formally approved.*

First Semester Programme

	CM	TD	ECTS
Module 1.1 (No module compensation)			**7**
Introduction à l'informatique 🇫🇷 🇬🇧	30	15	4
Technical English 🇬🇧	30		3
Module 1.2 (Allows module compensation)			**11**
Calculus 🇬🇧 🇫🇷	30	15	4
Mathématiques discrètes 1 🇫🇷	30	30	4
Statistiques 🇫🇷 🇬🇧	15	15	3
Module 1.3 (No module compensation)			**12**
Operating Systems 1 🇬🇧	30	15	4
Programming 1 🇬🇧	45	45	8
TOTAL	**210**	**135**	**30**

Second Semester Programme

	CM	TD	ECTS
Module 2.1 (No module compensation)			**2**
Academic Writing 🇬🇧	30		2
Module 2.2 (Allows module compensation)			**10**
Linear Algebra 🇬🇧	30	15	4
Mathématiques discrètes 2 🇬🇧 🇫🇷	15	15	3
Probabilités 🇫🇷	15	15	3
Module 2.3 (No module compensation)			**10**
Algorithms 1 🇬🇧	30	30	4
Programming 2 🇬🇧	45	30	6
Module 2.4 (No module compensation)			**8**
Introduction to Graphics 🇬🇧	30	15	4
Introduction to Data Analysis with Python 🇬🇧	30	15	4
TOTAL	**225**	**135**	**30**

Third Semester Programme

	CM	TD	ECTS
Module 3.1 (No module compensation)			**9**
Web Development: Front-End	25	20	5
Programming 3	25	20	4
Module 3.2 (No module compensation)			**8**
Algorithms 2	30	15	4
Operating Systems 2	25	20	4
Module 3.3 (No module compensation)			**6**
Modelling with UML	15	15	3
Software Engineering	30		3
Module 3.4 (No module compensation)			**8**
Data Science	30	15	4
Networks 1	30	15	4
TOTAL	**210**	**120**	**31**

Fourth Semester Programme

	CM	TD	ECTS
Module 4.1 (No module compensation)			**4**
Psychologie du travail en groupe 🇫🇷	15	15	2
Droit pour informaticiens 🇫🇷 🇬🇧	30		2
Module 4.2 (No module compensation)			**12**
Interaction Design 🇬🇧	25	20	4
Software Engineering Project 🇬🇧	10	35	4
Software Testing 🇬🇧 🇫🇷	37	20	4
Module 4.3 (No module compensation)			**7**
Algorithms 3 🇬🇧	25	20	4
Networks 2 🇬🇧	15	15	3
Module 4.4 (No module compensation)			**8**
Database Systems 🇬🇧	30	15	4
Applied Data Analysis with LLMs 🇬🇧	30	15	4
TOTAL	**217**	**155**	**31**

Fifth Semester Programme

	CM	TD	ECTS
Module 5.1 (No module compensation)			**2**
Introduction à la vie professionnelle	30		2
Module 5.2 (No module compensation) **Students must choose seven courses out of all offered optional courses.**			**28**
Backend Software Development	25	20	4
Big Data	25	20	4
Business Software Systems	25	20	4
Circuits numériques	25	20	4
Cloud-Based Applications	25	20	4
Design Patterns	25	20	4
Digital Transformation	25	20	4
Generative AI	25	20	4
Introduction to IOT	25	20	4
Introduction to IT Security	25	20	4
Introduction to Machine Learning	25	20	4
IT in a Real Financial Organization	25	20	4
Java for Enterprise Applications	25	20	4
Parallel and Distributed Systems	25	20	4
TOTAL	**205**	**140**	**30**

Sixth Semester Programme

	CM	TD	ECTS
Module 6.1 (No module compensation)			
Students choose one of the two alternatives.			**30**
Research-based Bachelor Project			30
Professional Internship			30
TOTAL	**0**	**0**	**30**

Flag(s): Show the primary and possibly the secondary language used in a course.

CM: Total number of (academic) hours of lectures (*cours magistraux*).

TD: Total number of (academic) hours of organized exercise sessions or practicals (*travaux dirigés* / *travaux pratiques*). Note that the total amount of work required might exceed this number, since additional independent work (homework, class preparation, or class wrap-up) is expected from students.

Remark: One "academic hour" shown in the table corresponds to one teaching unit (*unité d'enseignement*) of 45 minutes. The given number of academic hours is the maximal number of teaching units for a course, which can change due to the academic calendar of a specific semester (e.g. official national holidays, number of weeks with courses for the given semester).

ECTS: Number of credits in the European Credit Transfer System.

Module: Collection of courses. As described in Article 35 of the *study regulations*, a module can either apply internal "module compensation" or not. The study regulation with detailed explanations on the impact of module compensation is available on https://www.uni.lu/en/about/official-documents/.

Details for Semester 1 Courses

Module 1.1: Fundamentals

Computer Science is a modern scientific discipline centered on the study, development, and application of computing technologies. Information Technology (IT), a branch of engineering, focuses on the use of computers and digital tools to retrieve, transmit, and store information. This module aims to build a strong foundation in core concepts of computer science and IT, while also developing practical English skills - essential in the global IT industry.

After validation of this module, students are capable to:

- Practically apply the linguistic basics of the English language (grammar, domain-specific vocabulary), to allow them to follow IT-centered courses given in English.
- Answer in written English language to questions on scientific and technical topics, especially from the IT domain.
- Explain in a simple and correct way commonly used terms and abbreviations in information technology.
- Summarize some fundamental core concepts used in computer science.
- Make simple deductions on common principles in computer science, concerning both software and hardware.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

1. Introduction à l'informatique [4 ECTS]

Objectif: Rassembler tous les éléments de culture générale informatique qui seront développés dans les cours avancés, entre autres : les conversions de bases, les automates, le fonctionnement d'un réseau informatique, les outils de collaborations. Le cours s'appuie sur les contextes historiques qui ont mené aux différents points abordés et présentent les différentes notions permettant de comprendre les systèmes informatiques modernes.

Learning Outcomes: Après avoir suivi ce cours, les étudiants seront capables :

- de connaître les noms d'informaticiens célèbres et leurs contributions à l'histoire de l'informatique.
- d'expliquer de façon correcte et simple la signification des termes et abréviations communément utilisés en informatique.
- d'avoir une compréhension des concepts fondamentaux de l'informatique (logique binaire, architecture logiciel-matériel, notion de protocole, de code machine, de langage informatique).
- d'appliquer un certain nombre d'algorithmes de base à des situations concrètes.
- de connaître les principaux outils de collaboration informatique.
- de faire des raisonnements simples sur des concepts informatiques.
- de décrire des procédés et techniques informatiques.

Contenu:

- Historique des techniques de traitement de l'information.
- Conversion de bases et Calcul.
- Logique booléenne.
- Structure générale et fonctionnement des ordinateurs.
- Introduction aux automates.
- Présentation des UTMs (Universal Turing Machine).

- Introduction à l'algorithmique.
- Outils de Collaboration et évolution des langages.
- Introduction aux réseaux informatiques.
- Introduction à l'intelligence artificielle.

Enseignant(s): Dr. Mathieu Jimenez, M. Amal Akli

Prérequis: —

Langue: Français, Anglais

Modalité enseignement: Cours magistraux, travaux dirigés et travaux personnels.

Modalités d'évaluation:
Un examen écrit (100%).

Ouvrage de référence: Les références online seront publiées sur Moodle.

Notes: Les étudiants devront obligatoirement participer aux TD et remettre les travaux personnels qui leur sont demandés.

2. Technical English [3 ECTS]

Objectives: Evaluate existing working knowledge of the English language and bring it to perfection, especially in the information technology domain.

Learning Outcomes: On successful completion of this course, students are capable to:

- Use the English language in written and oral form in the context of information technology.
- Read and understand technical documents from the IT domain written in English language.
- Explain in English language the content of documents from the IT domain of medium complexity.

Description:
The course is given in English and so allows all students to practice their English language. If possible, two groups will be formed based on the students' experience with English. Special focus will be laid on the usage of the English language specifically in the context of information technology:

- Revision of the linguistic basis (grammar and specialised vocabulary).
- Investigation and analysis of thematic documents for improving the understanding of English language structures and usage of key vocabulary.

Lecturer(s): Mrs. Madeleine Hittinger

Prerequisites: Minimum level B2 (in the European system), having followed English courses of at least 120 hours in secondary school or in a language training programme.

Language: English

Evaluation:

- **First time students**: continuous evaluations during the course (50%) and a final written exam (50%).
- **Repeating students**: final written exam (100%).

Literature: Online references will be announced on the course website.

Module 1.2: Mathematics 1

Theoretical foundations of computer science are rooted in mathematics, making a solid mathematical background essential for studying both theoretical and applied aspects of the field. This foundation is also key to succeeding in more advanced, application-oriented courses in later semesters. The objective of this module is to equip students with essential mathematical knowledge and introduce the use of mathematical support tools, all through a practice-oriented approach.

After validation of this module, students are capable to:

- Explain definitions and properties of important mathematical objects like sets and functions.
- Use basic mathematical techniques for statistical analysis.
- Demonstrate the learned theories for solving simple mathematical problems.
- Apply the acquired mathematical knowledge for understanding scientific documents in computer science.
- Describe how the mathematical theories and tools can be applied to computer science problems.

Compensation

This module allows internal compensation between different courses (*Règlement des études*, Art. 35).

3. Calculus [4 ECTS]

Objectives: The course aims to ensure that all students have a solid understanding of basic calculus, including the analysis of elementary real functions of one variable and the use of finite and infinite sequences and series in computer science.

Learning Outcomes: On successful completion of this course, students are capable to:

- Solve problems in calculus of real-valued functions in one variable of different types (polynomial, logarithm, exponential, trigonometric functions).
- Determine convergence and (possibly) limits of simple sequences and series.
- Apply the most common proof techniques used in calculus to basic mathematical problems.
- Determine derivatives and antiderivatives of simple real-valued functions in one variable.

- Summarize some examples of close relationships between mathematics and computer science.

Description:

- Proofs with mathematical induction.
- Convergence of infinite sequences and their limits.
- Convergence of finite and infinite series and applications in computer science.
- Basic definitions and properties of real-valued functions in one variable.
- Elementary functions (polynomials, logarithm, exponential, trigonometric functions) and their properties.
- Derivatives and integration of real-valued functions and applications.
- Applications of real-valued functions in one variable to problems in computer science.

Lecturer(s): Prof. Franck Leprévost

Prerequisites: Mathematics of the elementary and secondary school level.

Language: English, French

Evaluation:
Final written exam (100%) for both **first time** and **repeating students**.

Literature: Franck Leprévost: *What Counts? A Hands-On Tutorial on Calculus (Revised Edition).* Ed. Amazon (2022). ISBN: 979-8352262726

4. Mathématiques discrètes 1 [4 ECTS]

Objectif: Fournir une introduction aux mathématiques discrètes, en traitant les techniques de base de la logique, des ensembles et du dénombrement, ainsi que de l'arithmétique.

Learning Outcomes: Après avoir suivi ce cours, les étudiants seront capables :

- d'appliquer les règles de logique élémentaire.
- d'utiliser les ensembles et les relations binaires.
- de représenter et de calculer avec les nombres en base quelconque.
- de calculer en arithmétique modulaire.
- de résoudre des petits problèmes de dénombrement.
- de comprendre et d'utiliser un modèle simplifié du protocole de chiffrement RSA.
- de résoudre certains problèmes élémentaires de théorie des graphes.
- d'appliquer un raisonnement par récurrence.

Contenu:

- Logique élémentaire.
- Listes et suites.
- Ensembles.
- Dénombrement.
- Arithmétique élémentaire.
- Relations binaires.
- Congruences et arithmétique modulaire.
- Graphes.

Enseignant(s): Prof. Bruno Teheux, M. Elmir Avdusinovic, M. Tobias Fischbach

Prérequis: Familiarité avec les notions mathématiques élémentaires de l'enseignement primaire et secondaire.

Langue: Français

Modalités d'évaluation:

Contrôle des connaissances pendant le semestre et examen écrit en fin de semestre. La note finale à la fin du semestre d'hiver est la meilleure valeur entre

- la note de l'examen écrit de fin de semestre,
- la moyenne pondérée entre le résultat du contrôle de mi-semestre et la note de l'examen écrit de fin de semestre (l'examen final comptant pour 3/4 des points dans cette moyenne pondérée).

Cette note peut-être légèrement améliorée d'1/2 point en fonction des résultats aux quiz et aux exercices proposés durant le semestre.

Ouvrage de référence:

- Michel Marchand, «Outils mathématiques pour l'informaticien », 2e éd. De Boeck Université, Bruxelles 2005 [ISBN 978-2804149635].
- O. Levin, Discrete Mathematics, An Open Introduction (open access book available at http://discrete.openmathbooks.org/dmoi3.html)
- Franck Leprévost, "How big is big? How fast is fast? A Hands-On Tutorial on Mathematics of Computation", available via Amazon. (2021).

5. Statistiques [3 ECTS]

Objectif: Familiariser l'étudiant(e) avec les techniques de base des statistiques descriptives.

Learning Outcomes: Après avoir suivi ce cours, les étudiants seront capables :

- de traiter des séries statistiques.
- d'appliquer la théorie statistique pour des problèmes informatiques.
- d'utiliser des outils informatique pour des calculs statistiques.

Contenu:

- Organisation des données statistiques.

- Traitement des séries statistiques.
- Paramètres caractéristiques.

Enseignant(s): Prof. Bernard Steenis

Prérequis: —

Langue: Français, Anglais

Modalité enseignement: Cours magistraux, travaux dirigés et pratiques, travaux à domicile.

Les travaux pratiques de calcul statistique seront proposés en MS-Office-Excel et OpenOffice-Calc.

Modalités d'évaluation:

- **Première participation**: Un contrôle des connaissances pratiques (50%) et un examen écrit (50%).
- **Étudiants redoublants**: examen écrit (100%).

Ouvrage de référence: Catherine Dehon, Jean-Jacques Droesbeke, Catherine Vermandele: "Éléments de statistique", éditions de l'Université de Bruxelles. ISBN 978-2340009080.

Notes: Les étudiants devront obligatoirement participer aux TD et remettre les travaux personnels qui leur sont demandés.

Module 1.3: Programming Fundamentals 1

Strong programming skills, along with a solid understanding of core operating system concepts, are essential for every applied computer scientist. This module introduces students to programming using the Java programming language and provides a foundational understanding of how operating systems manage resources and processes. These topics will be explored in greater depth in subsequent semesters.

After validation of this module, students are capable to :

- Explain fundamental concepts commonly used in operating systems and the methodology of programming, in a theoretical and practical way.
- Analyze simple problems to find a solution in form of an algorithm.
- Translate an algorithm into the Java programming language, including program testing and documentation.
- Compare basic characteristics of standard operating systems like Linux and Microsoft Windows.
- Use the acquired knowledge of this module for the autonomous consultation of technical documentation.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

6. Operating Systems 1 [4 ECTS]

Objectives: Operating systems provide an interface between the hardware and the applications on the computer. It helps with processes scheduling, user and rights management and file management. The purpose of this course is to provide a basic introduction to operating systems.

Learning Outcomes: On successful completion of this course, students are capable to:

- Explain the basic principles of OS, including process scheduling, user and rights management and file systems.

Description:

- Introduction to OS (Windows, Linux, MacOS).
- Processes: management and scheduling.
- User and Rights management.

- File Systems.
- Basic of system administration, lab exercises, virtualisation.
- Mobile OS (Android, iOS).

Lecturer(s): Dr. Christian Grévisse, Mrs. Alisa Vorokhota

Prerequisites: —

Language: English

Teaching modality: Lectures and lab exercises.

Evaluation:

- **First time students**:
  - Midterm exam (40%),
  - Final exam (60%).
- **Repeating students**: 100% written final exam.

Literature: Books and other inputs will be given in the lecture and made available on the course Moodle web page.

7. Programming 1 [8 ECTS]

Objectives: This course introduces the fundamentals of programming, together with the key concepts of object-orientation. The Java programming language will be used primarily for the code examples.

Learning Outcomes: On successful completion of this course, students are capable to:

- Understand the fundamentals of programming.
- Apply object-oriented concepts.
- Design algorithms of average complexity.
- Implement those algorithms in the Java programming language.

- Perform basic testing.

Description:
The course discusses and illustrates the following topics using a hands-on approach:

- Overview of different types of programming languages.
- Basic skills of problem-solving: from problem descriptions to algorithms.
- Data types.
- Control structures.
- Classes and objects.
- Encapsulation and access control.
- Subtyping, inheritance, and polymorphism.
- Basics of generic programming.
- Exception handling.

Lecturer(s): Prof. Steffen Rothkugel, Mr. Akif Agdag, Mr. Elmir Avdusinovic

Prerequisites: —

Language: English

Teaching modality: Interleaved sequence of lectures and supervised practical sessions.

Evaluation:
Winter semester:

- First time students:
  - closed-book written midterm exam (40%)
  - closed-book written final exam (60%)

- Repeating students: closed-book written final exam (100%)

Summer semester: closed-book written final exam (100%)

Literature:

- "The Java Language Specification, Java SE Edition", James Gosling et al, Addison-Wesley, ISBN 978-0133260229, available online at: https://docs.oracle.com/javase/specs/
- Additional material will be announced during the lecture.

Details for Semester 2 Courses

Module 2.1: Professional Preparation 1

Academic writing is essential for knowledge dissemination, the development of critical thinking skills, and the progression of both academic and professional careers. In addition, proficiency in standard and video-based presentation skills is becoming increasingly important.

After validation of this module, students are capable to:

- Write a well-structured and grammatically correct text in technical English, particularly in the IT domain.
- Give an oral presentation in English on a topic from the IT domain.
- Prepare videos for the promotion of scientific ideas.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

8. Academic Writing [2 ECTS]

Objectives: Academic writing is a formal style of writing used in universities and scholarly publications. This course teaches the fundamental basics of academic writing and presentations in the academic context. Some basics on promotional video preparation are also covered.

Learning Outcomes: On successful completion of this course, students are capable to:

- Understand the fundamentals of academic writing.
- Understand basic rules to prepare and give presentations.

- Know how to record and manipulate videos for promotion of academic ideas.
- Apply the theoretical knowledge on academic communication in practice.

Description:

- Basics of “Academic writing”.
- Using LaTeX to write academic documents.
- Using Microsoft Word to write reports.
- Skills to prepare and hold presentations.
- Guidelines on creating slides.
- Creating videos for promotion of scientific ideas.

Lecturer(s): Mr. Matthew Kahn

Prerequisites: —

Language: English

Teaching modality: Lectures and practical exercises, partly done in class, partly done as homework.

Evaluation:
A language course in which the participants begin with widely varying levels of competence must strive to balance the demonstration of minimum levels of competence in order to pass the course with the chance to earn credit for significant improvement. Therefore, this course combines both elements by dividing course work and credit awarded into minimum required output demonstrating a minimum competency and output which is above the minimum to garner more credit as measured in class points. The latter includes demonstrating marked improvement in competence.

There is a total of 20 class points potentially awarded which are divided into four categories, each receiving a possible total 25% or 5 class points each:

1. **Participation**: Attendance is 4 points out of 5 and asking good questions or other valuable class participation may be awarded with up to another point. - 25%. **5 class points**

2. **Following instructions and attention to detail:** Turning in / presenting completed assignments on time while carefully fulfilling the instructions which include format. topics. specified lengths and composition. minimum levels of clarity, conciseness and thoroughness. Late submissions will have increasing numbers of points deducted as time passes from the due date until a final date when zero points will be awarded. - 25%. **5 class points**

3. **Exemplary submissions** may be awarded additional points in this category for clarity, conciseness, thoroughness and extra utilization of the format as will those assignments awarded minimum points from category 2 which are subsequently improved and re-submitted within the specified time frame. - 25%. **5 class points.**

4. **Demonstration of competence:** Written final exam which will recapitulate various competencies developed during the course. - 25%. **5 class points** which means students cannot fail to attend class and turn in only a few assignments then pass the course by taking the final. See additional note below.

A spreadsheet combining attendance for each class and points awarded for assignments submitted will be displayed in each class and regularly posted on Moodle for students to keep track of their accumulated points throughout the semester as well as validate it within one week, pointing out any data entry errors which might occur.

Nota Bene: University students are adults, freely participating in a course of study which they are equally free to leave at any time. Each may choose to fail, merely pass or excel in this course depending on their desires and each will be awarded the appropriate class points for the work they choose to perform. If you have any questions regarding how your work will be evaluated, please ask me in class so everyone can benefit from my response and if you are still unsure, feel free to contact me at `matthew.kahn@education.lu`.

Literature: Online references will be provided on the Moodle course page.

Module 2.2: Mathematics 2

Module 2.2 aims to broaden students' mathematical foundation, equipping them with essential knowledge for advanced work in applied information technology. It focuses on fundamental concepts in probability theory and linear algebra, both critical for a wide range of practical IT applications.

After validation of this module, students are capable to:

- Explain mathematical definitions and properties of vectors, vector spaces, matrices, and probability.
- Apply these definitions and properties together with elementary proof techniques on simple problems and applications in information technology.
- Extend their acquired knowledge autonomously by consultation of periodicals or articles in the IT domain.
- Explain how these basic mathematical tools are applied in various IT applications.

Compensation

This module allows internal compensation between different courses (*Règlement des études*, Art. 35).

9. Linear Algebra [4 ECTS]

Objectives: To guarantee that all students have a sufficient knowledge of basic linear algebra needed in computer science.

Learning Outcomes: On successful completion of this course, students are capable to:

- Calculate with vectors and matrices.
- Use the Algorithm of Gauss for the solution of various problems in Linear algebra.

- Explain the relevance of determinants and Eigenvalues for some practical problems in computer science.
- Apply the learned topics to problems in IT related with basic linear algebra.

Description:

- Vectors and vector spaces.
- Linear independence of vectors and basis of a vector space.
- Matrices and matrix operations.
- Algorithm of Gauss for solving simultaneous linear equations.
- Matrix determinant and matrix inversion.
- Matrix normal forms.
- Eigenvalues and eigenspaces.

Lecturer(s): Prof. Franck Leprévost

Prerequisites: Mathematics of the elementary and secondary school level, and of the 1st BINFO semester.

Language: English

Evaluation:
One single final exam (100%) for both first-time and repeating students.

Literature: Franck Leprévost: *Order Matters! A Hands-On Tutorial on Linear Algebra (Revised Edition).* Ed. Amazon (2021). ISBN: 979-8595860642

Notes: Students are obliged to participate to the exercise sessions and do their homework.

10. Mathématiques discrètes 2 [3 ECTS]

Objectives: The course explains basic algorithms like Euclid's algorithm, modular exponentiation and the Chinese Remainder Theorem. Tests for primality are presented. In addition, operations for modular computations in Z/p^dZ and Z/mnZ are explained.

Learning Outcomes: On successful completion of this course, students are capable to:

- Explain basic algorithms for integers.
- Apply the learned techniques to common integer-related problems.
- Transfer the applied techniques to related problems for large numbers.

Description:

- Prime numbers: What? How many? What for?
- Euclid's algorithm: How?
- Euler's totient phi-function.
- $(Z/nZ)^*$ and phi(n).
- Exercises : Computing in Z/p^d Z and in Z/mnZ.
- Chinese remainder theorem.
- Modular exponentiation.
- Questions about prime numbers.
- Eratosthenes Sieve Method Digression : Gauß approximations.
- How to decide if a number is prime or not?
- Naive method.
- Fermat's test.
- Legendre symbol and Solovay-Strassen test.

Lecturer(s): Prof. Franck Leprévost

Prerequisites: Mathematics of the elementary and secondary school level, and of the 1st BINFO semester, especially *Mathématiques Discrètes 1.*

Language: English, French

Evaluation:
One single final exam (100%) for both first-time and repeating students.

Literature: Franck Leprévost: *How Big is Big? How Fast is Fast? A Hands-On Tutorial on Mathematics of Computation (Revised Edition).* Ed. Amazon (2020). ISBN: 979-8642630556

11. Probabilités [3 ECTS]

Objectif: Ce cours vise à familiariser l'étudiant avec les notions de base du calcul des probabilités. Tout événement à priori inconnu est généralement décrit par des probabilités. Des exemples classiques sont: un jet de dés ou le lancer d'une pièce. Les probabilités sont utiles dans de nombreux domaines, soit pour faire des estimations, soit pour prendre de bonnes décisions par rapport à des évènements inconnus.

Learning Outcomes: Après avoir suivi ce cours, les étudiants seront capables :

- d'expliquer les bases de la théorie des probabilités.
- de formaliser et de résoudre des problèmes avec des probabilités.
- d'utiliser des distributions discrètes et continues.

Contenu:

- Analyse combinatoire: dénombrement des possibilités, combinaisons, permutations.
- Variables aléatoires: notation, probabilités conditionnelles, théorème de Bayes, marginalisation, indépendance.
- Espérance mathématique: valeur moyenne, variance, écart-type, linéarité, corrélation.
- Distributions discrètes: épreuves de Bernoulli, loi géométrique, loi binomiale.
- Distributions continues: densités, loi uniforme, loi normale.

- La loi des grands nombres.
- Estimations.

Enseignant(s): Dr. Christian Franck

Prérequis: Statistiques

Langue: Français

Modalité enseignement: Cours magistraux, travaux dirigés et pratiques, en alternance. Les étudiants devront obligatoirement participer aux TD et remettre les travaux personnels qui leur sont demandés.

Modalités d'évaluation:

- **Première participation**: 25% de la note obtenues pour les travaux personnels réalisés à domicile, 75% de la note obtenue à l'examen final.
- **Étudiants redoublants**: Examen final (100%).

Module 2.3: Programming Fundamentals 2

Programming skills are closely tied to a solid understanding of algorithms and data structures essential for solving algorithmic problems. This module aims to strengthen programming proficiency by introducing the Swift language, emphasizing reactive programming techniques, and providing a structured exploration of algorithms and data structures used in standard computer science challenges.

After validation of this module, students are capable to:

- Explain the fundamental concepts of algorithms and reactive programming.
- Analyse a problem of medium complexity to transform it into an algorithm and implement the problem in the Java / Swift programming language, practicing software development in a group including testing and documentation.

- Explain the advantages and disadvantages of several standard data structures.
- Analyse the complexity of several algorithms operating on data structures.
- Improve their knowledge on algorithms and programming by the autonomous consultation of technical documentation.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

12. Algorithms 1 [4 ECTS]

Objectives: This course is mainly intended for deepening students' knowledge of essential linear data structures: array, linked list, associative array, hash table. The implementation of such data structures will be discussed in detail, further familiarizing students with the underlying ideas. The course focuses on the Java programming language in examples and for implementation work in the exercise sessions that complement all and each of the lectures.

Learning Outcomes: On successful completion of this course, students are capable to:

- Explain the concepts, properties and usage of standard linear data structures.
- Implement basic operators (like insert, search, delete) on these data structures.

Description:

- Array: basic operators, binary search, iterative sorting algorithms.
- Dynamic array, queue, stack.
- Linked list: basic operators, variants.

- Associative array: basic operators, dictionaries.
- Hash table: hash functions, basic operators, solutions to collision and overflow problems.

Lecturer(s): Prof. Denis Zampunieris

Prerequisites: *Programming 1*

Language: English

Evaluation:
Final written exam (100%) for both **first time** and **repeating students**.

Literature: "Introduction to Algorithms (3rd edition)", T.Cormen, C.Leiserson, R.Rivest & C.Stein, MIT Press, 2009, ISBN 9780262033848.

Other relevant literature and online resources will be announced on the Moodle course website.

13. Programming 2 [6 ECTS]

Objectives: The course introduces advanced notions of object-oriented programming, supporting the design and implementation of more complex software systems. Both the Java and Swift programming languages will be used to convey the relevant concepts, in a comparative fashion.

Learning Outcomes: On successful completion of this course, students are capable to:

- Analyse problem descriptions of average size.
- Assess software designs in several terms, including extensibility, reusability, and maintainability.
- Apply fundamental principles of object-orientation to come up with a sound software design.
- Create robust implementations in both the Swift and Java programming languages.

Description:

The course discusses and illustrates the following topics using a hands-on approach:

- Value and reference types.
- Functions and closures.
- Protocols and extensions.
- From object-oriented to protocol-oriented programming.
- Advanced error handling techniques.
- Advanced generic programming.
- Serialisation and persistence.

Lecturer(s): Prof. Steffen Rothkugel, Mr. Akif Agdag

Prerequisites: Programming 1

Language: English

Evaluation:
Summer semester:

- **First time students**: software project (40%) + final written exam (60%)
- **Repeating students**: final written exam (100%)

Winter semester: final written exam (100%)

Literature: Relevant literature and various resources will be announced on the Moodle course website.

Notes: Prerequisite to sitting the final exam is the completion of all intermediate deliverables.

Module 2.4: Computer Science 1

Computer science encompasses a broad spectrum of research areas that shape the technologies we use every day. This module provides a foundational introduction to key topics within Information Technology (IT), offering students a comprehensive overview of essential principles and applications. Emphasis is placed on two significant and practical domains: computer graphics and data acquisition.

Computer graphics explores the methods and tools used to create and manipulate visual content, enabling applications in gaming, simulation, design, and user interfaces. Data acquisition, on the other hand, involves the collection, processing, and analysis of data from physical environments, playing a critical role in systems such as environmental monitoring, healthcare devices, and industrial automation.

After validation of this module, students are capable to:

- Explain the concepts and fundamental processes relative to data acquisition and processing.
- Use standard Python libraries for realizing common tasks related with data processing.
- Explain basic concepts in 2D and 3D graphics.
- Use popular open-source tools to perform graphical manipulations on 2D and 3D images.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

14. Introduction to Graphics [4 ECTS]

Objectives: Computer Graphics is a very important field of computer science. Its use today spans virtually all scientific fields and is utilized for design, presentation, education and training. In this course, an introduction into basics of computer graphics theory and practice is given with an applied approach based on open-source tools like Gimp, Inkscape, and Blender.

Learning Outcomes: On successful completion of this course, students are capable to:

- Explain core techniques and data representation for the creation and manipulation of two-dimensional images.
- Apply these techniques to three-dimensional graphical objects.
- Use popular graphics-related open-source tools like Gimp, Inkscape, or Blender for simple 2D and 3D image processing tasks.
- Develop simple graphics-related programs with the help of JavaFX.

Description:

- Raster versus vector graphics.
- Two-dimensional graphics:
  - Pixel, coordinate systems, colors.
  - Basic graphical shapes: line segments, circles, ellipsis, polygons.
  - Affine transformations like translation, rotation, scaling, shearing.
  - Image manipulation with convolution filters.
  - Image manipulation based on statistical information.
- Practical introduction to JavaFX and the open source programs Gimp and Inkscape.
- Introduction to three-dimensional graphics:
  - Similarities and differences between 3D and 2D.
  - Representation of three dimensional objects, texture, material, ...
  - Introduction into light and shadow handling and ray tracing.
- Practical example for the open source 3D editor Blender.

Lecturer(s): Prof. Volker Müller

Prerequisites: —

Language: English

Teaching modality: Weekly lecture and several practical homework exercises.

Evaluation:

- **First time students:** final exam (70%), 4 graded exercises (30%)
- **Repeating students**: final exam (100%)

Literature:

- “Introduction to Computer Graphics”, David Eck, 2018, free online version available at https://math.hws.edu/graphicsbook/ .
- “Introduction to Computer Graphics - Using Java 2D and 3D”, Frank Klawonn, 2012, Springer, online version available via https://a-z.lu.
- The Java Tutorials - 2D Graphics, available at https://docs.oracle.com/javase/tutorial/2d/index.html.

Additional literature and online resources will be provided on the Moodle course website.

15. Introduction to Data Analysis with Python [4 ECTS]

Objectives: This course teaches the fundamental ideas of cleaning, manipulating, processing, and analyzing data. Students will work on data analysis problems encountered in various data-intensive applications. The course includes many in-class programming exercises where students are expected to work on various case studies. Through these exercises, the course will also serve as an introduction to data analysis and modern scientific computing using the Python programming language.

Learning Outcomes: On successful completion of this course, students are capable to:

- Understand the fundamentals of data analysis.
- Use the Python programming and its libraries NumPy, Pandas, and Matplotlib/Seaborn.

- Pose questions, collect relevant data, analyze data, interpret data and provide insights.
- Present data-driven insights using data visualization.

Description:

- Introduction: What is data analysis?
- Python basics, Build-in Data Structures, Functions, and Files.
- Development with Python: poetry and uv, code analysis, performance analysis, documentation.
- Jupyter Labs
- NumPy and Pandas basics: Arrays and Vectorized Computation.
- Data Visualization with Matplotlib and Seaborn.
- Data Acquisition, Preparation and Management.
- Data Visualization.
- Time Series.
- Introduction to Modeling Libraries in Python.
- Data Analysis Examples.

Lecturer(s): Prof. Volker Müller, Mr. Elmir Avdusinovic, Mr. Enea Mancellari

Prerequisites: —

Language: English

Teaching modality: Lectures and practical exercises

Evaluation:
First time students:

- Closed-book midterm exam (paper-based) – 25%
- Practical project – 25%

- Closed-book final written exam (paper-based) – 50%

Repeating students: final written closed-book exam (100%).

Literature:

- Python for Data Analysis, O'REILLY, ISBN-10: 1491957662
- Python Data Analysis, Steve Eddison, ISBN: 1709888989
- Introduction to Data Science with Python: Basics of Numpy and Pandas, Mark Smart, ISBN-10: 1731036841

Lectures slides and any supplemental course content will be provided on the Moodle course web site.

Details for Semester 3 Courses

Module 3.1: Programming Fundamentals 3

Web and mobile applications play a vital role in everyday life, with user-friendly interfaces being central to their effectiveness. This module builds on earlier programming experience by introducing the design and development of graphical user interfaces (GUIs). It explores key concepts such as event-driven programming and multi-threading, which are essential for creating responsive, interactive applications. Students will gain practical skills in developing intuitive user interfaces, preparing them to build modern software solutions across platforms.

After validation of this module, students are capable to:

- Develop dynamic web pages using HTML, CSS, and JavaScript.
- Use frameworks to create graphical user interfaces for desktop and mobile applications.
- Explain key programming paradigms relevant to web and mobile application development.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

16. Web Development: Front-End [5 ECTS]

Objectives: The course provides an introduction to front-end web development, from a software engineering perspective. It will cover the foundational building blocks of the Web, user interface design fundamentals, command line tools, and popular frameworks for building websites and web applications. After the course, students will be able to build the front-end of any kind of websites and web applications.

Learning Outcomes: On successful completion of this course, students are capable to:

- Identify the key components of web technologies.
- Recognize the importance of software engineering for web development.
- Understand classic and modern tools for front-end development.
- Compare alternatives between frameworks, coding standards, and design patterns.
- Design and develop front-end systems for websites and web applications.

Description:
The course provides an introduction to front-end web development, from a software engineering perspective. The course will cover the foundational building blocks of the Web, user interface design fundamentals, command line tools, and popular frameworks for building websites and web applications. After the course, students will be able to build the front-end of any kind of websites and web applications. An overview of the covered topics is given as follows:

1. HTML: The structural layer of the Web:
   - History: Web browser wars, architectures, ACID test.
   - Document types: XML, DTDs, quirks mode.
   - Markup notation: Tags, microformats, accessibility.
2. CSS: The presentation layer of the web:
   - CSS Object Model: box model, selectors, specificity levels.
   - Standards & Preprocessors: Less, Sass, BEM.
   - Frameworks: Bootstrap, Material UI.
3. JavaScript: The behavioral layer of the Web:
   - Document Object Model:

```
- parsers, implementations.
- ES5, ES6, TypeScript.
- jQuery, mootools.
```

4. UI design fundamentals:
   - Layout: navigation, forms.
   - Responsive web design: media queries.
   - Callbacks: event-driven programming.
5. Database integration:
   - Transport protocols: JSON, XML, text.
   - Serverless: RESTful APIs, SPAs.
6. Tooling:
   - Command line interface: CLI fundamentals.
   - Dependency management: bower, browserify, webpack.
   - Building systems: grunt, gulp.
   - Version control: git, subversion.
7. Web app frameworks:
   - Fundamentals: N-way binding, MVC, virtual DOM.
   - Examples: Angular, Vue, React.
8. Performance:
   - Minifiers: uglifyjs.
   - Monitoring: inspectors, debuggers.
   - Auditing: PageSpeed, Lighthouse.
9. Testing:
   - Fundamentals: unit testing, integration testing, end-to-end testing.

Lecturer(s): Dr. Remus Dobrican

Prerequisites: —

Language: English

Evaluation:

- **First time participants:**
  - Practical work in TPs: 30%
  - Participation in class: 5%
  - Final exam (multiple-choice quiz): 65%
- **Redoing session:** The final exam can be retaken in the next exam session. Grades from the coding exercises will be retained until the student passes the course.

Literature: **Reference textbook:**

1. J. Robbins. 2025. *Learning Web Design.* O'Reilly Media, 6th ed., ISBN 9781098137687.

Recommended books:

1. R. Anquetil. 2019. Fundamental Concepts for Web Development, 1st ed., ISBN 9781702250382.
2. M. Haverbeke. 2018. Eloquent JavaScript, 3rd ed., ISBN 9781593279509.
3. S. Krug. 2000. Don't Make Me Think, 3rd ed., ISBN 9781593279509.

17. Programming 3 [4 ECTS]

Objectives: The aim of this course is to familiarize students with the fundamentals of graphical user interface (GUI) programming using widely used frameworks, while focusing on essential paradigms such as event-driven programming, the Model–View–Controller design pattern, and multi-threading.

The course also provides a brief introduction to mobile development with Kotlin and extended reality, and fosters the ability to use AI coding assistants responsibly as a support tool for experimentation, explanation, and debugging.

Learning Outcomes: On successful completion of this course, students are capable to:

- Develop event-driven programs for Swing-based desktop applications and Android mobile apps in Java.
- Apply multi-threading in programming tasks to keep applications responsive.
- Explain and apply common design patterns such as Model–View–Controller in the context of interactive and mobile applications.
- Use AI tools responsibly to generate boilerplate, compare alternative solutions, and critically evaluate code.
- Extend their knowledge independently on relevant techniques in GUI and mobile programming through literature and documentation.

Description:
This course addresses both the theory and practice of GUI programming. Core topics include:

- Event-driven programming,
- Multi-threading, and
- Design patterns such as Model–View–Controller.

Practical work is based primarily on examples in:

- Java Swing for desktop applications, and
- Java-based Android for mobile applications.

The course also includes a short introduction to Kotlin as a modern alternative language for Android, and an outlook on extended reality programming with ARCore. Students are encouraged to use AI coding assistants during exercises, but always in combination with self-verification and conceptual quizzes. AI may be used to speed up routine tasks and explore alternative solutions, while the focus remains on understanding, explanation, and critical evaluation of generated code.

Lecturer(s): Dr. Christian Franck

Prerequisites: The semester one course *Programming 1* must have been passed successfully to follow this course.

Language: English

Teaching modality: The course is taught according to the flipped-classroom model. Students study theoretical material in advance, while classroom time is devoted mainly to discussions, exercises, and practical programming work.

Evaluation:
Final written exam (100%) for both **first time** and **repeating students**. AI use is not permitted during the exam.

Literature: Online documentation of the different toolkits and frameworks, other documentation provided on the Moodle course site.

Module 3.2: Computer Science 2

Understanding complex systems is essential in today's computing landscape. Advanced Operating Systems and Algorithms courses strengthen expertise in core areas vital for building scalable, secure, and efficient software. Mastery of advanced algorithms also equips students to tackle real-world problems with greater precision and performance. For those pursuing careers in systems engineering, AI, cybersecurity, or high-performance computing, this course offers the critical skills needed to innovate and lead in the tech industry.

After validation of this module, students are capable to:

- Explain the concepts used in evolutions of classical operating systems.
- Use the abstractions of such concepts in the context of application development for the covered operating systems.
- Explain both theoretical and practical aspects of essential non-linear

data structures.

- Apply more advanced data structures in application development.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

18. Algorithms 2 [4 ECTS]

Objectives: This course builds on and extends some topics covered in *Algorithms 1* and is mainly intended for deepening students' knowledge and understanding of essential non-linear data structures. Recursivity as a crucial concept both in the definition and the utilization of such data structures will be discussed in detail, further familiarizing students with the underlying idea.

Learning Outcomes: On successful completion of this course, students are capable to:

- Describe the usage of tree data structures and their main operations.
- Use these data structures in practical programming problems.

Description:

- Binary and m-way search trees, (weight-)balanced trees, B/B*-trees, tries.
- Implementation and analysis of fundamental tree operations, their applications and libraries (also in other programming languages, e.g. Java or C#).
- Algorithms and data structures for Interpreters and Virtual machines.
- Compilation schemes and some basics of optimization for Compilers.

Lecturer(s): Dr. Jean Botev

Prerequisites: *Algorithms 1*

Language: English

Teaching modality: Students must deliver all exercises and participate to all practical sessions.

Evaluation:

- **First time students**:
  - Mini-project (25%, part 1),
  - Homework assignments (25%, part 2),
  - Final written exam (50%).
- **Repeating students**: final written exam (100%).

Literature:

- Introduction to Algorithms, Thomas H. Cormen et al., MIT Press, ISBN 978-0262033848
- The Art of Computer Programming – Volume 1: Fundamental Algorithms, Donald E. Knuth, Addison Wesley, ISBN 0-201-89683-4
- Compilers: Principles, Techniques and Tools, Alfred V. Aho et al., Addison Wesley, ISBN 978-0321547989
- Programming Languages - An Interpreter-Based Approach, Samuel N. Kamin, Addison Wesley, ISBN 0-201-06824-9

19. Operating Systems 2 [4 ECTS]

Objectives: The course deepens the understanding of the concepts of operating systems together with the abstractions provided for developers.

Learning Outcomes: On successful completion of this course, students are capable to:

- Identify and explain the responsibilities of an operating system.
- Compare and evaluate the properties of different operating systems.
- Designate the abstractions operating systems provide.

- Properly use those abstractions from within applications.
- Handle heterogeneity appropriately.

Description:
Operating systems represent sophisticated runtime platforms for all types of software. Their primary purpose is to mediate between applications and different kinds of resources. The internal workings of modern operating systems as well as the abstractions they provide for application programmers will be introduced throughout this lecture. Practical experiments will illustrate the theoretical concepts in the context of the Windows and Linux operating systems. Topics covered include:

- Memory management.
- Processes and threads.
- Scheduling.
- Synchronisation.
- Resource management.
- File systems.

Lecturer(s): Prof. Steffen Rothkugel, Dr. Jean Botev

Prerequisites: The course *Operating Systems 1* (first semester) must have been passed successfully to follow this course.

Language: English

Evaluation:
Winter semester:

- **First time students**: closed-book written midterm exam (40%) + closed-book written final exam (60%)
- **Repeating students**: closed-book written final exam (100%)

Summer semester: closed-book written final exam (100%)

Literature:

- Andrew S. Tanenbaum: “Modern Operating Systems”. 4th Edition, Pearson Education, ISBN 978-1292061429
- William Stallings: “Operating Systems”. 8th Edition, Pearson Education, ISBN 978-1292061351
- Abraham Silberschatz et al.: “Operating System Concepts”. 9th Edition, Wiley & Sons, ISBN 978-1118093757

Additional material will be announced during the lecture on the Moodle course website.

Module 3.3: Software Engineering 1

Software engineering is essential for building reliable, efficient, and scalable software that underpins modern life. Quality software development should be a core focus for every programmer. It goes beyond coding - requiring structured processes and engineering techniques to improve both speed and quality. This module, the first of two, introduces key methodologies, modeling techniques, and practical tools fundamental to effective software engineering.

After validation of this module, students are capable to:

- Design and interpret UML diagrams (including use case, class, sequence, activity, and state diagrams) to model the structure and behavior of software systems, effectively communicating system requirements and architecture to stakeholders.
- Explain the core ideas of common software development methodologies, in particular for agile development.
- Explain the principles of the different activities that an IT expert should follow to realize an object-oriented software product.
- Apply the learned standard techniques and tools for software engineering in a practical example.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

20. Modelling with UML [3 ECTS]

Objectif: The goal of the course is to introduce students to the Unified Modeling Language (UML) for the analysis and design of software systems. The course will provide students with the necessary and essential theoretical and practical understanding of UML. The course focuses on the most important diagrams that can be employed to design and describe the structure and the behaviour of software systems.

Learning Outcomes: After successful completion of the course, students are capable to:

- Understand the role of UML in object-oriented design.
- Understand static and dynamic design modelling.
- Understand and explain the role and the use of the UML models and diagrams.
- Use UML modelling tools.
- Use the appropriate diagram notations and create diagrams according to the corresponding development phase.
- Identify use cases, create a use case diagram and describe different use case scenarios.
- Elaborate interaction diagrams, e.g. sequence and communication diagrams for use case scenarios.
- Elaborate class diagrams and introduce advanced associations like aggregation, composition and generalization.
- Use generative AI for UML modeling generation and analysis.

Contenu:

Introduction to UML

- A notation to support iterative and incremental Software Development.

Use case diagrams

- Actor, Use cases, Associations, Notations.

Class Diagrams and Object Diagrams

- Classes, Attributes, Operations, Associations.
- Aggregation, Composition.
- Objects and Links.

State Machine Diagrams

- States, Transitions, Events.
- Composite States.

Sequence Diagrams

- Lifelines and messages.

Activity Diagrams

- Actions, Control flows, Object flows.

The supervised exercise sessions are devoted to:

- the elaboration of UML diagrams.
- the use of UML modelling tools.

Enseignant(s): Prof. Nicolas Guelfi

Prérequis: —

Langue: Français, Anglais

Modalités d'évaluation:
First time students: Continuous assessment mid-term 'bonus points only' exam (up to 6 bonus points to add to the final exam grade), and written final examination (100%).

Repeating students: Final written exam (100%).

Ouvrage de référence:

1. Martina Seidl, Marion Scholz, Christian Huemer, Gerti Kappel, "UML @ Classroom: An Introduction to Object-Oriented Modeling". Springer 2015.
2. Craig Larman,"Applying UML and Patterns: An Introduction to object-oriented Analysis and Design and iterative development", Third Edition, Pearson Education, 2005, ISBN: 978-8-177589795.
3. Grady Booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide". Pearson Education.
4. Christoph Kecher, Alexander Salvanos, "UML 2.5, Das umfassende Handbuch", Rheinwerk computing, ISBN: 978-3836249362.
5. Laurent Debrauwer, Fien Van der Heyde, "UML 2.5 - Initiation, exemples et exercices corrigés", Editions ENI, ISBN: 978-2-409-02408-5.

21. Software Engineering [3 ECTS]

Objectives: This course prepares future software engineers to lead development projects in practical, team-based settings using established software engineering methodologies. It covers common development approaches and their core components, complemented by student-led presentations on key tools, typically open-source, essential to real-world software engineering practice.

Learning Outcomes: On successful completion of this course, students are capable to:

- Understand common software engineering processes including waterfall (linear) development, iterated and incremental approaches, and agile approaches.
- Apply the core principles of software engineering in software development processes.

- Apply Agile principles and practices such as Scrum and Kanban to plan, execute, and manage iterative software development projects in collaborative team environments.
- Practice essential software engineering tools, introduced through student group presentations.

Description:
The theoretical part of the course covers common software development methodologies - including the waterfall model, incremental, and agile approaches - and their key components. It also provides a brief overview of the principles behind requirement analysis, design, implementation, testing, and maintenance. Central to the course are student group projects, where teams deliver detailed presentations on the practical use of important software engineering tools, often open-source. The following topics will be covered in student presentations:

1. Containers in Software Engineering: **Docker** and **Podman**.
2. Distributed version control with **Git** and **GitHub**.
3. Build automation tools: **Maven** and **Gradle**.
4. Core tasks with an IDE: code refactoring, syntax checking, code documentation generation, using Debuggers.
5. Code review with **Gerrit** and with **GitHub**.
6. Static code analysis and continuous Inspection: **SonarQube**.
7. Continuous Integration with **Jenkins** and with **GitHub**.
8. Testing frameworks: **Junit**, **Selenium**, **Mockito**.
9. AI tools in Software Coding (**GitHub Copilot** and alternatives) - Opportunities and Challenges.
10. Continuous Security Monitoring.
11. Software Deployment (possibly in the cloud) and Software Monitoring.
12. Project Management with **OpenProject**, **Trello**, **Jira**.

Lecturer(s): Prof. Volker Müller

Prerequisites: Students should have some practical experience with programming in the Java language using an IDE (Eclipse, Visual Studio Code, or IntelliJ).

Language: English

Teaching modality: The course will consist of two parts, which will be done in parallel:

- The lecturer covers the theoretical foundations of software engineering, including software development methodologies.
- Student groups deliver practical presentations on tools commonly used in real-world software engineering.

Evaluation:

- **First time students**:
  - Presentation of the group project (20%).
  - Deliverables of the group project (video 20%, paper 20%).
  - Final written exam (40%).
- **Repeating students**: final written exam (100%).

Literature: Genuine literature will be provided during the course on the Moodle course website.

Module 3.4: Computer Science 3

Data science and networking are core areas of information technology. This module introduces both fields, laying the foundation for future specialization. It covers widely used Internet networking protocols and provides a practical introduction to data science techniques for analyzing large datasets.

After validation of this module, students are capable to:

- Explain the principles of widely used Internet technologies and protocols.
- Apply standard techniques for acquiring and processing large datasets.
- Use their knowledge of networks and data science to independently understand technical documentation and publications.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

22. Data Science [4 ECTS]

Objectives: The successful candidate will understand the basic theoretical concepts of data-centric aspects and will be able to work on data-centric problems.

Learning Outcomes: On successful completion of the course the student should be able to:

- Explain and apply basic theoretical concepts on selected aspects of data processing.
- Develop appropriate solutions for data-centered problems.
- Consolidation of the acquired competences in the subject area through a research project.

Description:

In this course, the term ‘data’ is seen centric, and we will look at data from different perspectives. We will discuss the following selected aspects of data:

- Data Preparation and Preprocessing,
- Data Statistics,
- Data Security,
- Data Privacy,

- Data Management
- Big and Small Data,
- Data Retrieval,
- Data Visualization, and
- Data Analytics.

Lecturer(s): Prof. Christoph Schommer

Prerequisites: —

Language: English

Evaluation:
First time students: 50% written assignment or oral interview and 50% practical.

Repeating students: 100% exam

Literature:

- Elmasri, Navathe: Fundamentals of Database Systems. Pearson Addison Wesley. 2006.
- Han, Kamber: Data Mining – Concepts and Techniques. Morgan Kaufmann. 2011.
- Manning, Raghavan, Schütze: Introduction to Information Retrieval. Cambridge University Press.
- Ware: Information Visualization. Morgan Kaufmann. 2012.
- Witten, Kamber: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann.
- Aggarwal, Yu: Privacy-Preserving Data Mining – Models and Algorithms. Springer. 2008.
- Marz: Big Data: Principles and best practices of scalable realtime data systems. Manning. 2015.

In addition, different articles, reports, and journals contributions will be referenced on the course Moodle page.

23. Networks 1 [4 ECTS]

Objectives: The course aims to provide an introduction to the TCP/IP networking protocols and architectures, including practical application of network analysis tools.

Learning Outcomes: On successful completion of this course, students are capable to:

- Explain the different protocol layers of the TCP/IP protocol.
- Develop a computer program in the Java programming language that realizes TCP/IP connections.
- Explain different protocols for routing.
- Use some network analysis tools like Wireshark for traffic analysis.

Description:

The class uses the layout of networking classes introduced by J.Kurose and K.Ross. This model is strongly adhered to by many universities in the USA and Europe.

The class will follow a top-down approach to introducing the TCP/IP protocols. It will introduce the functioning and protocol elements of popular protocols (HTTP/SMTP/SIP) and continue with the routing algorithms in TCP/IP networks.

A selected set of TCP/IP specific mechanism (retransmission, state machine) will be also shown in class. The class will address also Layer 2 protocols and show the cross-layer interactions between protocols. Multimedia protocols, the QoS related parameters and security protocols for TCP/IP networks are also covered in the class. The class will also address basic TCP/IP programming. Students will be introduced to single/multithreaded client server programming using Java. At the end of the class, the student is expected to master network analysis tools (like TcPDump, Wireshark), understand the basic routing protocols (RIP, OSPF, BGP) and be able to write simple UDP/TCP client server applications.

Lecturer(s): Prof. Radu State, Dr. Stefan Hommes

Prerequisites: Java or Python programming

Language: English

Teaching modality: Students have to attend the supervised practical sessions and have to hand in the deliverables they will be asked for.

Evaluation:

First time students: Wireshark exam (25%), written exam (75%).

Repeating students: Wireshark exam (25%), written exam (75%).

Literature: J. Kurose and K. Ross, "Computer Networking: A Top-Down Approach", ISBN-13: 978-0132856201.

Details for Semester 4 Courses

Module 4.1: Professional Preparation 2

In the fourth semester, students begin focused preparation for entering the professional world by exploring practical topics essential to businesses. This module deepens their understanding of key organizational and legal issues in the workplace and emphasizes the value of teamwork. It also features active involvement from professionals within Luxembourg's business community.

After validation of this module, students are capable to:

- Explain the principal functions of an enterprise (purchasing, logistics, sale, marketing, production, finances, human resources, strategy).
- Identify these functions in case studies presented by external speakers (employees).
- Analyse group behavior and group dynamics.
- Negotiate in conflicts inside a team of IT workers.
- Explain key legal aspects relevant to information technology.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

24. Psychologie du travail en groupe [2 ECTS]

Objectif: Comprendre et se familiariser avec les différents aspects du travail en groupe (psychologie et dynamique affective des groupes, gestion, négociation et techniques de résolution des conflits) tant dans le monde physique de l'entreprise que dans le monde numérique et les relations de travail médiées par les TIC. Apports théoriques, conceptuels et mise en pratique par une étude de cas en groupe (dossier de groupe).

Learning Outcomes: Après avoir suivi ce cours, les étudiants seront capables :

- de comprendre et analyser les différents aspects psychosociologiques du travail en groupe.
- de l'appliquer à une étude de cas réelle, de travail en groupe, en vue de s'approprier la démarche critique-réflexive comme méthode et posture éthique.

Contenu:

Partie I. Des biais algorithmiques aux biais cognitifs

1. C'est quoi un biais algorithmique ?
2. La notion de biais cognitif. a. **Travaux en groupe durant le cours (durée 1 séance 1/2 à 2 séances).** b. **Travail de recherche documentaire par les étudiants, en groupe,** pour s'approprier la notion de biais cognitif, avec utilisation comparée de **différentes IA** (ChatGPT, Deepseek, Copilot, Gemini, Claude, etc.) et **analyse critique** de la réponse (vérification de la qualité des informations et sources académiquesproposées par l'IA, en termes de fiabilité + pertinence). c. **objectif visé : améliorer la qualité de son prompt** pour améliorer la qualité de la réponse de l'IA
3. Les principaux biais cognitifs (présentation/restitution en groupe des principaux biais cognitifs) après la recherche documentaire.
4. Comment lutter contre les biais cognitifs ?

Partie II. La prise de décision en groupe

1. Les enjeux de la prise de décision.
2. L'effet *Groupthink* en psychologie.
3. La notion de « décision absurde » en sociologie.

4. Etude de cas réels documentés (*Navette Challenger*, *OceanGate*) : travaux en groupe durant 1 séance.
5. Comment éviter les « décisions absurdes » dans les groupes ?

Partie III. La dynamique des groupes

1. **Caractéristiques d'un groupe en psychologie sociale**: dynamique et dimension affective.
2. **Leader vs Chef**: La notion d'influence ; phénomènes de leadership et charisme; le leader comme « entrepreneur d'identité ».
3. **Le pouvoir**: Pouvoir et soumission/obéissance ; contre-pouvoir.
4. **L'efficacité d'un groupe au travail**: notion d'efficacité : approche quanti/quali ; rôle de la tâche ; phénomène de paresse sociale (Social Loafing) ; approche cognitive (modèles mentaux, mémoire transactive) ; intelligence collective : vers la coopération/coopétition ?
5. **La recherche du consensus dans le groupe**: notion de consensus ; influence desminorités actives.

Partie IV. Gestion des conflits

1. Approche nouvelle des conflits.
2. Causes/origines des conflits.
3. Types de conflits.
4. Méthodes de gestion des conflits/ Approche éthique.

Exercices pratiques EN COURS (exercices très similaires - sans piège - dans Exam N°1) : apprentissage de l'écoute active/empathie (technique d'écoute non-directive et de reformulation de C. ROGERS (2019) pour la régulation/négociation des conflits) (étalement de l'apprentissage sur 5 séances, d'où la **présence nécessaire à tous les cours**).

Préparation du dossier en GROUPE: analyse d'un **conflit éditorial dans l'écriture collaborative de Wikipédia** (apports théoriques et méthodologiques

: comprendre la **gouvernance** de Wikipédia, notion de **biens communs**, vigilance participative et régulation, **gestion des conflits**).

Enseignant(s): Mme Corinne Martin

Prérequis: —

Langue: Français

Modalité enseignement: Les étudiants devront effectuer des recherches / travaux (travail personnel / en groupe) pendant et en dehors des cours.

Modalités d'évaluation:

Cours et Examen en français :

- EXAM 1 : devoir individuel sur table : coefficient 40 % (questions de cours et études de cas de conflits en groupe/ de reformulation en réunion, etc. ; en français)
- EXAM 2 : dossier à réaliser en petit groupe (3-4 pers.) étude de cas avec cahier des charges : coefficient 30 % (dossier en français) ; analyse d'un cas de conflit éditorial de groupe, dans l'écriture collaborative en ligne de Wikipédia
- EXAM 3 : Divers exercices notés durant les differentes seances du module (= 30% de la note globale)
- Rattrapage en janvier : 1 dossier à rendre (étude de cas avec cahier des charges) : 100 %

Ouvrage de référence:

- Anzieu D., Martin J.-Y., 2007, *La dynamique des groupes restreints*, Paris, PUF Quadrige.
- Aubert Nicole, 2010, *Le culte de l'urgence*, Paris, Flammarion.
- Audebert Patrick, 2005, *Bien négocier*, Paris, Ed. Organisation.
- Augustinova M., Oberlé D., 2013, *Psychologie sociale du groupe au travail,* Bruxelles, De Boeck.
- Bellenger L., *Réussissez toutes vos négociations*, Paris, ESF, 2009.
- Blanchet A., Trognon A., 2008, *La psychologie des groupes*, Paris, A. Colin.

- Cardon D., Levrel J., 2009, « La vigilance participative. Une interprétation de la gouvernance de Wikipédia », *Réseaux*, 2, no 154, pp. 51-89.
- Cardon D., 2013, Dans l'esprit du PageRank. Une enquête sur l'algorithme de Google, *Réseaux* (dossier *« Politique des algorithmes. Les métriques du Web »*, 1, no 177, pp. 63-95.
- Cardon D., 2015, *A quoi rêvent les algorithmes. Nos vies à l'heure des big data*, Paris, Seuil.
- Casilli A., 2019, *En attendant les robots. Enquête sur le travail du clic*, Paris, Seuil.
- Casilli A., 2022, Plateformes numériques, p. 185-201, *in* Didier Fassin (dir.), *La société qui vient*, Paris, Seuil.
- Falgas J., Robert P., 2025, Comment le discours médiatique sur l'IA empêche d'envisager d'autres possibles, *The Conversation.*
- Fischer G. N., 2020, *Les concepts fondamentaux de la psychologie sociale*, Malakoff, Dunod (6e éd. actualisée).
- Hadji C., 2025, Ce que ChatGPT change à l'évaluation des élèves, *The Conversation.*
- Hellrieger D., Slocum J.W., 2006, *Management des organisations*, Bruxelles, De Boeck Université, 2e éd. (chapitre 16 : la gestion des conflits).
- Jean A., 2021, Les algorithmes font-ils la loi ?, Paris, Éd. L'Observatoire.
- Jean A., 2019, *De l'autre côté de la machine. Voyage d'une scientifique au pays des algorithmes*, Paris, Éd. L'Observatoire
- Magakian J.-L., Barmeyer C., Bouziat X., Hounounou A., Le Loarne S., 2003, *50 fiches pour aborder la gestion stratégique des ressources humaines*, Paris, Bréal.
- Morel C., 2014, *Les décisions absurdes I. Sociologie des erreurs radicales et persistantes*, Paris, Gallimard.
- Morel C., 2014, *Les décisions absurdes II. Comment les éviter ?* Paris, Gallimard.
- Morel C., 2018, *Les décisions absurdes III. L'enfer des règles, les pièges relationnels*, Paris, Gallimard.

- Robert P., 2020, *L'impensé numérique, interprétations critiques et logiques pragmatiques de l'impensé*, (Dir) T. 2, Paris, Éd. des archives contemporaines.
- Rogers C.R., 2019, *La relation d'aide et la psychothérapie*, Paris, ESF.

25. Droit pour informaticiens [2 ECTS]

Objectif: Sensibiliser l'étudiant aux aspects juridiques liés à l'informatique.

Learning Outcomes: Après avoir suivi ce cours, les étudiants seront capables :

- de réfléchir sur des questions juridiques de l'informatique.
- de décrire les aspects différents de la criminalité informatique.

Contenu:

- Introduction générale au droit.
- Propriété intellectuelle et informatique.
- Contrats et informatique (développement, maintenance, etc.).
- Communications électroniques et preuve.
- Informatique et vie privée.
- Aspects de droit de la responsabilité.
- Criminalité informatique.

Enseignant(s): M. Florian Poncin, M. Vincent Semidei

Prérequis: —

Langue: Français, Anglais

Modalité enseignement: Les étudiants devront remettre les travaux personnels qui leur sont demandés.

Modalités d'évaluation:
Un examen écrit (100%) pour **étudiants débutants** / **redoublants**.

Ouvrage de référence: Les références seront annoncées lors de la cours.

Module 4.2: Software Engineering 2

Practical experience is essential for software developers to confidently navigate the development process. This module provides students with the opportunity to apply their knowledge in a 12-week, group-based software development project, fostering the integration of technical skills and soft competencies developed in earlier semesters. Key topics such as effective UI design, which enhances user experience, and comprehensive software testing, which ensures interface reliability across diverse use cases, will also be explored in this module.

After validation of this module, students are capable to:

- Apply and combine the theories learned in the courses *Modelling with UML* and *Software Engineering* in a practical development project.
- Reflect on their experiences made in a 12 week practical software development project.
- Apply the techniques and basic concepts of verification of computer systems on the basis of testing, as part of the software engineering process.
- Explain the concepts and the procedures for user interface engineering, including "intelligent" interaction with users in a computer system.
- Extend their knowledge on aspects of software engineering by autonomously consulting technical documentation and journals.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

26. Interaction Design [4 ECTS]

Objectives: The overall objective of this course is to help deepen students' knowledge and skills in user interface and interaction design.

Learning Outcomes: On successful completion of this course, students are capable to:

- Describe, explain, and use a standard analysis and design process, and standard analysis and design terminology.
- Recognize several basic design patterns and common software techniques, and be able to use them in product design.
- Generate alternative solutions for analysis and design problems, evaluate them, choose a good alternative, and explain and defend this choice.

Description:
The Interaction Design course follows the following core topics. Many of these combine lectures with studio time in the classroom.

1. Introduction to Interaction Design.
2. Understanding Users.
3. Needs, Requirements and Hierarchical Task Analysis.
4. Prototyping.
5. Conceptual Design.
6. Physical Design.
7. Evaluation.

Lecturer(s): Prof. Steve Frysinger

Prerequisites: —

Language: English

Teaching modality: Students must deliver all exercises and participate in all practical sessions.

Evaluation:
First time students: two exams during the semester (30% each), a group final project (30%), and participation in the classroom (10%).

Repeating students: final exam (100%).

Literature: Interaction Design: Beyond Human-Computer Interaction (2nd Edition), by Preece, Rogers, and Sharp. Wiley, 2007.

27. Software Engineering Project [4 ECTS]

Objectives: Prepare the future software engineer to run a software engineering project in a team following a development method. Teaching is centered on the realization of a group project (including homework) in which the students will engineer a software system of low complexity that should educate them concerning problems, principles, methods and techniques of software engineering.

The main knowledge fields are: requirements analysis, GUI prototyping, Software engineering environments, software processes, project management, and teamwork.

Learning Outcomes: On successful completion of this course, students are capable to:

- Describe experiences made during the running of a development project in a team.
- Prove experience with several practically relevant tools used in software engineering.
- Reflect on the practical difficulties during software development in a team and how to tackle these problems.

Description:
The detailed description of the planned project will be given in the course. The course will be mostly based on active student participation in a project related with software engineering. The following parts are foreseen:

- Introduction into important aspects and tools used in Software Engineering (similar to the respective course in semester 3).
- Running a software development in practice in a team composed of 3-4 students.
- Two sprints are done. After each sprint, a presentation on the current status of the project with discussion must be given before the other students.
- Each group must also present the final result in front of the audience and develop two videos: a promotion video, and a video with all technical details.

Lecturer(s): Prof. Volker Müller

Prerequisites: *Software Engineering*

Language: English

Evaluation:
Proven usage of important tools in software engineering as demanded in project description (30%), active participation during the sprints and their evaluations, including the intermediate presentations (30%), result of the software development process (40%).

Literature:

- Course material (student presentations) from course *Software Engineering* (Semester 3).
- Additional literature will be announced in class.

28. Software Testing [4 ECTS]

Objectives: The objective of the course is to present the three classical stages for software testing, namely unit, integration and system testing. The course is focusing on OO Java programming, as a basis for unit testing. The concepts, methods and techniques seen during the course are illustrated by recent testing frameworks widely used in the industry.

Learning Outcomes: On successful completion of this course, students are capable to:

- Reflect on the classical stages of software testing.
- Describe the different techniques and issues for the different stages.
- Apply the learned techniques for software testing in practical scenarios.

Description:
The course will be divided into 2 parts:

1. **Part I: Software Testing principles and practice**
    - (1) Software testing : presents the overall view of the testing process, the definitions and the main issues related to the software life-cycle. Some classical testing techniques are presented.
    - (2) Unit, (3) integration and (4) system testing are the three remaining modules, going in depth into the issues and techniques specific to each of these life cycle stages.
2. **Part II. Towards certification: Certified Tester – Foundation Level (CTFL)**
    - Nowadays, software testing is identified as a key role in a company that requires specific knowledge on which we can get a certification.
    - The goal of this part is to go through the ISTQB (International Software Testing Qualifications Board) standard and prepare the first level of qualification.
    - The student will then be free to go through the official ISTQB certification exam to be certified (`https://www.istqb.org/`).

Lecturer(s): Prof. Yves Le Traon, Dr. Seung Yeob Shin, Mr. Dietmar Gehring

Prerequisites: —

Language: English, French

Teaching modality: Students must deliver all exercises and homeworks, and participate to all practical sessions.

Evaluation:

First time students: The grading will be based on a combination of two written exams (Part I: 60-70% and Part II: 30-40%). The practical exams may be taken into account for the final grade.

Repeating students: one written exam (100%).

Literature:

- Introduction to Software Testing - Paul Ammann and Jeff Offutt - ISBN-13: 9780521880381 – Cambridge Press, 2008.
- Foundations of Software Testing - Aditya Mathur - Addison-Wesley Professional – 2007.
- Software testing techniques - B. Beizer - Van Nostrand Reinhold, 1992.
- Le test des logiciels - S. Xanthakis et Co - Editions Hermes, 2000.

Notes: The course includes the preparation for the CTFL certification. This preparation counts for 12 hours, included in the total of 37 hours for CM.

Module 4.3: Computer Science 4

This module builds on two key topics from previous semesters, offering a deeper exploration of algorithms through a practical project while further advancing students' understanding of specialized algorithms, particularly those applicable to real-world scenarios. Additionally, it provides more in-depth coverage of advanced network-related technologies.

After validation of this module, students are capable to:

- Explain more advanced technologies in the field of networking.
- Use and explain more advanced algorithms for practical applications.
- Extend their knowledge on algorithms and networking by autonomously consulting technical journals and articles.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

29. Algorithms 3 [4 ECTS]

Objectives: This course builds on and extends topics covered in "Algorithms 1" and "Algorithms 2", primarily aiming at a practical point of view. Based on real-world examples using different programming languages, selected data structures and algorithms will be discussed. Their functional as well as non-functional properties such as performance, memory consumption and concurrency issues will be investigated, guiding the selection process of different alternative approaches.

Learning Outcomes: On successful completion of this course, students are capable to:

- Use popular data structures in practice with Java.
- Describe how generics work in the Java programming language.
- Explain exact algorithms, heuristics, and AI-based Optimization.

Description:
After a thorough discussion of Java generics, data structures and algorithms provided by the Java Collections Framework will be closely examined. C++ templates will be introduced. The basics concepts of the C++ Standard Template Library will be covered, elaborating on its general design and performing a comparative analysis. Additional kinds of algorithms will be investigated, including exact algorithms as well as heuristics for optimisation and AI.

Lecturer(s): Prof. Steffen Rothkugel, Dr. Matthias Brust, Mr. Akif Agdag

Prerequisites: Algorithms 1, Algorithms 2.

Language: English

Evaluation:

Summer semester:

- **First time students**: one practical project (30%) + final written exam (70%),
- **Repeating students**: final written exam (100%).

Winter semester: final written exam (100%)

Literature:

- Introduction to Algorithms, Thomas H. Cormen et al., MIT Press, ISBN 978-0262033848
- Java Generics, Maurice Naftalin and. Philip Wadler, O'Reilly Media, ISBN 978-0596527754
- Generic Programming and the STL, Matthew H. Austern, Addison-Wesley Professional, ISBN 978-0201309560

30. Networks 2 [3 ECTS]

Objectives: Study data link and network layer protocols, wireless and mobile networks. Understand security in computer networks. Intensive hands-on experience with networking in the labs.

Learning Outcomes: On successful completion of this course, students are capable to:

- Explain the basics of data link and network link layer protocols.
- Apply practical experience gained about networking in a lab environment.
- Describe security-related aspects of networking.

Description:

- Network layer: Routing algorithms.
- Link Layer and LAN, Ethernet, ARP.

- Organization of internet standards and RFCs.
- Wireless and mobile networks CDMA, GSM, 3GPP.
- Multimedia networking RTSP.
- Security in Computer Networks: PGP, IPSec, SSL/TLS.

Lecturer(s): Prof. Bernard Steenis

Prerequisites: Networks 1

Language: English

Teaching modality: Participation in TP is obligatory and will be the main part of the final grade.

Evaluation:
First time students: practicals and homeworks (100%).

Repeating students: written exam (100%).

Literature: Kurose / Ross textbook and slides: https://www-net.cs.umass.edu/kurose-ross-ppt-6e/

Module 4.4: Data Science

Databases and data science play a critical role in today's information-driven world. Databases provide the foundational infrastructure for storing, organizing, and managing large volumes of structured data efficiently and securely. Meanwhile, data science builds on this foundation to extract meaningful insights from both structured and unstructured data using statistical analysis, machine learning, and data visualization techniques. Together, these fields enable informed decision-making, drive innovation across industries, and support the development of intelligent systems that can adapt and respond to complex real-world challenges. As data continues to grow in volume and importance, proficiency in both databases and data science is increasingly essential for modern technology professionals.

After validation of this module, students are capable to:

- Understand fundamental database concepts, including data models, schemas, and database management systems (DBMS).
- Write basic Structured Query Language (SQL) commands to create, read, update, and delete data in a relational database.
- Demonstrate the ability to use a DBMS to create and manage databases effectively.
- Understand basic concepts of transactions, concurrency, and database security.
- Develop problem-solving skills related to data organization and retrieval in practical scenarios.
- Apply descriptive statistics and exploratory data analysis techniques to summarize and interpret datasets.
- Explain the basic mathematical foundations of machine learning algorithms and LLMs.
- Extend their knowledge on databases and mathematical foundations of machine learning by consultation of journals and scientific articles.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

31. Database Systems [4 ECTS]

Objectives: This course provides a solid foundation in database concepts, focusing primarily on relational database systems. Students will learn how to design, implement, and query relational databases using SQL. The course also includes a brief introduction to non-relational database models such as NoSQL, key-value stores, and document databases to offer a broader perspective on data storage technologies.

Learning Outcomes: On successful completion of this course, students will be able to:

- Explain the basic principles and architecture of database systems.
- Design normalized relational database schemas using ER modeling.
- Write effective SQL queries for data manipulation and retrieval.
- Understand and apply concepts of data integrity, constraints, and transactions.
- Compare relational databases with key types of non-relational databases.
- Identify appropriate database types for different data scenarios.

Description:
The course covers the following topics using a hands-on approach:

- Introduction to Databases and Data Models.
- Relational Database Concepts.
- Entity-Relationship (ER) Modeling.
- Relational Schema Design and Normalization.
- Structured Query Language (SQL).
- Constraints, Indexes, and Transactions.
- Accessing Databases from Programming Languages.
- Overview of alternative database types (NoSQL, NewSQL, Object-Oriented Databases, Graph Databases, ...).
- Use Cases and Comparative Analysis of Database Types.

The course is organised in the form of a seminar. The topics are worked on and presented by students, followed by a discussion and accompanied by practical tasks.

Lecturer(s): Prof. Steffen Rothkugel

Prerequisites: —

Language: English

Evaluation:

First time students: presentation (20%), discussion (30%), practical part (10%), closed-book written exam (40%).

Repeating students: closed-book written final exam (100%).

Literature:

- Database System Concepts. Abraham Silberschatz, Henry Korth, S. Sudarshan, McGraw-Hill Education, 7th edition, 978-1260084504
- Fundamentals of Database Systems. Ramez Elmasri, Shamkant B. Navathe. Pearson, 7th edition, 978-0-13-397077-7
- Database Systems: The Complete Book. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, Pearson Prentice Hall, 2nd edition, 978-1-29202-447-9
- Database Management Systems. Raghu Ramakrishnan, Johannes Gehrke, McGraw-Hill Education, 3rd edition, 978-0072465631

Additional references will be announced in the class.

32. Applied Data Analysis with LLMs [4 ECTS]

Objectives:

1. Solidify statistical foundations: Ensure students are comfortable with descriptive statistics, sampling theory, hypothesis testing and confidence intervals, enabling them to choose appropriate statistical techniques for data problems.
2. Introduce core machine-learning concepts: Provide a practical overview of supervised and unsupervised learning, regression and classification algorithms, clustering and dimensionality reduction, and evaluation methods like cross-validation.
3. Explain the basics of neural-networks and transformer architectures: Introduce deep learning fundamentals, culminating in the transformer model that underpins modern LLMs.

4. Demystify large language models: Explain how LLMs are trained, their capabilities and limitations, and the significance of Karpathy's Software 3.0 vision. Promote critical thinking: highlight issues of bias, fairness and hallucinations in LLMs.

5. Develop data analysis and coding proficiency with LLM assistance: Strengthen skills for data analysis and visualization, learn how to use an LLM to generate and debug the code needed for data analysis and visualization.

Learning Outcomes: At the end of the course, students will be able to:

- Compute and interpret summary statistics (means, medians, quantiles, variances, interquartile ranges) and visualize data distributions.
- Use statistical inference methods, including confidence intervals, hypothesis tests and bootstrapping, to draw conclusions from samples.
- Formulate and understand the concept of data correlation.
- Apply linear regression to curve-fitting problems. Understand the concept of overfitting.
- Apply statistical tests to compare different design candidates.
- Understand and apply sampling and bootstrapping techniques.
- Data transformation: data cleaning and dimension reduction.
- Understand how statistics can be used to set engineering simulation parameters with respect to target confidence level.
- Differentiate between supervised and unsupervised machine learning, select appropriate algorithms (e.g., regression, classification, clustering) and justify their choices.
- Understand and experiment with neural networks and deep learning: Gain some understanding of how neural networks function, including layers, activation functions, and training through back-propagation, and be able to apply simple deep-learning models to data-analysis tasks with the help of an LLM.

- Evaluate and compare machine-learning models effectively: Understand and use common evaluation techniques (such as accuracy, precision, recall, F1-score and cross-validation) to assess model performance, compare different algorithms and select appropriate models for a given task.
- Describe the basics of the architecture of transformers and understand how LLMs leverage transformer models.
- Design and refine prompts to program an LLM, illustrating the Software 3.0 paradigm where we "program" (mostly) by giving instructions in plain language to an AI rather than writing traditional code.
- Critically evaluate LLM outputs, recognizing biases and hallucinations, and apply strategies such as retrieval augmented generation to mitigate the problem of wrong outputs.

Description:

The course covers the following topics:

- Overview of the use of statistics in data analysis, machine learning and engineering.
- Descriptive statistics and exploratory data analysis: population, sample, mean, median, variance, empirical distribution histogram, cumulative distribution function, quantile, box plots, density plots, scatterplot and coefficient of correlation.
- Inferential statistics: sampling, sampling bias and bootstrap.
- Confidence intervals and margin of errors.
- Statistical significance and p-values. Hypothesis tests: one way versus two two ways tests.
- Introduction to Machine Learning. Supervised and unsupervised learning. Overview of regression and classification problems. Model Evaluation and Selection.
- Introduction to Neural Networks.
- Transformers and the Attention Mechanism. Large Language Models and Software 3.0.
- Prompt Engineering and LLM-Assisted Data Analysis.

- Ethical Use of LLMs and Bias Mitigation.

Lecturer(s): Prof. Nicolas Navet

Prerequisites: —

Language: English

Evaluation:

- **First time students:** final written exam (70%) and take-home assignment (30%).
- **Repeating students:** final written exam (100%).

Literature:

- "Practical Statistics for Data Scientists", P. Bruce, A. Bruce, P.Gedeck, second edition, O'Reilly, 2020.
- "Software is changing (again)" Andrej Karpathy, https://youtu.be/LCEmiRjPEtQ

More literature will be announced during the course via the Moodle course website.

Details for Semester 5 Courses

Module 5.1: Professional Preparation 3

Proper preparation for students' initial steps into their professional careers is crucial. This preparation includes not only IT-related aspects but also detailed guidance on the administrative steps required to start a career. This module equips students for these tasks, with a focus on preparing for an internship in a partner institution during the final semester of the BINFO program or a future professional career.

After validation of this module, students are capable to:

- Understand legal and other important aspects of work contracts.
- Prepare effective job application materials, including resumes, cover letters, and professional portfolios.
- Gain skills in interview techniques, networking, and career planning to successfully navigate the job market.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

33. Introduction à la vie professionnelle [2 ECTS]

Objectif: La fin des études et l'entrée dans la vie professionnelle marquent un tournant essentiel dans la vie. Le cours vise à aider l'étudiant à passer de la vie universitaire à la vie active de manière professionnelle. Cette transition sera facilitée par la présentation de la réalité du monde du travail et des différentes démarches pour décrocher un emploi épanouissant.

Learning Outcomes: Après avoir suivi ce cours, les étudiants seront capables :

- de comprendre les spécificités du marché du travail luxembourgeois.
- d'avoir une vision globale de l'entreprise.
- d'être bien préparé dans leur recherche d'emploi.
- de bien intégrer leur future entreprise.

Contenu:

1. Paysage économique luxembourgeois:
   - Caractéristiques du marché du travail luxembourgeois.
   - Aperçu des principaux secteurs d'activités.
2. Fonctionnement d'une entreprise:
   - Missions des principaux départements.
   - Impact de la transformation digitale des entreprises.
3. Techniques de recherche d'emploi:
   - Description du processus de recrutement d'une entreprise.
   - Identification des attentes du recruteur.
   - Rédaction de la lettre de motivation et du CV.
   - Préparation de l'entretien d'embauche.
4. Le droit du travail:
   - Composantes du contrat de travail.
   - Différents types de contrat de travail.
   - Période d'essai et fin du contrat.
5. L'entrepreneuriat:
   - Aperçu des différents types d'entreprises.
   - Vade-mecum des démarches à effectuer.

Enseignant(s): Mme Céline Hieronimus

Prérequis: —

Langue: Français

Modalité enseignement: Les étudiants sont invités à préparer l'interview des experts invités.

Modalités d'évaluation:
Étudiants débutants et redoublants: Un examen écrit final (100%).

Ouvrage de référence:

- La gestion des talents - C. Dejoux, M. Thévenet (Editions Dunod)
- Les 100 schémas du management – D.Autissier, K.Johnson, L.Giraud (Editions Eyrolles)
- L'art de la reconnaissance au travail – L.Becker (Editions InterEditions)
- Ressources Humaines – J-M Peretti (Editions Vuibert)
- La boîte à outils des soft skills – N. Van Laethem, J-M Josset (Editions Dunod)
- Management : L'essentiel des concepts et pratiques – S. Robbins, D. DeCenzo, M. Coulter, C-C Rüling (Editions Pearson)

Module 5.2: Specialisation

This module provides a portfolio of optional courses, enabling students to choose seven electives aligned with their personal specialization interests. The courses focus on in-demand topics, primarily within software development and artificial intelligence, reflecting current industry needs.

After validation of this module, students are capable to:

- Extend their knowledge by autonomously consulting publicly available technical documentation and tutorials on advanced topics of software engineering or other currently practically interesting topics.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

34. Backend Software Development [4 ECTS]

Objectives: This course covers advanced topics in back-end Web development, including API design, implementation using different languages and frameworks, and integration of databases. Proper use of asynchronous programming together with concurrency handling to tackle scalability is discussed extensively based on several case studies.

Learning Outcomes: On successful completion of this course, students are capable to:

- Assess Web development technologies and frameworks.
- Design and develop scalable back-ends for Web apps.
- Properly use asynchronous programming while dealing with concurrency issues.

Description:
The course covers the following topics using a hands-on approach:

- Basics:
  - Asynchronous and reactive programming.
  - RESTful API design and use.
  - OpenAPI.
  - GraphQL.
  - Interaction between front- and back-end.
- JavaScript-based Web development:
  - Node.js.
  - Express.
- Python-based Web development:

  - Flask.

- Databases and persistence:
  - MySQL.
  - MongoDB.

Lecturer(s): Prof. Steffen Rothkugel, Mr. Akif Agdag

Prerequisites: —

Language: English

Evaluation:
Winter Semester:

- First time students: continuous evaluation (30%), three closed-book written exams (70%).
- Repeating students: closed-book written final exam (100%).

Summer Semester:

- Closed-book written final exam (100%)

Literature: References will be announced in the class.

35. Big Data [4 ECTS]

Objectives: The course is about (classical and new) techniques that are involved in the Big Data paradigm. The main goal is to spark the discussion about the Trade Offs between the classical data processing techniques and the upcoming ones for big data. In addition, students should get basic knowledge on how to automatically process and analyze huge amount of data.

Learning Outcomes: On successful completion of this course, students are capable to:

- Demonstrate the ability to understand how to model a database system and the Trade Offs when the database goes big data, such as: consistency versus scalability and performance.

- Explain the database technologies for big data and analyze their pros and cons for proper usage.
- Design and develop big data solutions by adapting existing tools, designing new ones or a combination of both.
- Explain the concepts and the limits of the automated processing of data.
- Differentiate supervised and unsupervised learning and when one technique should be applied.
- Describe the concept of features and the importance of choosing discriminating ones.
- Select among basic algorithms for extracting information from a large data set and apply them.

Description:

The course is about (classical and new) techniques that are involved in the Big Data paradigm. The course combines two of the key dimensions of Big Data, namely:

Part I - Design and development for big data

The first part of the course will discuss databases and distributed computing algorithms for hosting and processing very large amounts of data:

1. Conceptual modeling (ER Model), Relational Model (Algebra and SQL), Schema design (ER to Relational).
2. Files and Access methods (except DHT).
3. Distributed Databases, Data Warehouse and C-Store.
4. NewSQL, Distributed Hash Tables, MapReduce, NoSQL.

The main goal of the first part is to spark the discussion about the Trade Offs between the classical data processing techniques and the upcoming ones for big data.

Part II - Data mining, classification and aggregation

The objectives of Part II are to guarantee that the students have a basic knowledge to automatically process and analyze huge amount of data. In particular, the two main objectives are to 1) extract information from a data set and 2) transform and store the data in a convenient way for further use (data mining, classification and aggregation):

1. Classification (random forest, . . .).
2. Clustering (k-means, . . .).
3. Outlier and anomaly detection (local outlier factor, . . .).
4. Regression (least squares, . . .).

Lecturer(s): Prof. Tegawendé Bissyande, Dr. Iyiola Emmanuel Olatunji

Prerequisites: —

Language: English

Evaluation:
First time students: Practical exercises, homework, continuous evaluation.

Repeating students: written exam (100%).

Literature: Relevant literature will be provided during the course.

36. Business Software Systems [4 ECTS]

Objectives: The students learn an overview and advantages of business software systems, especially ERP systems, and where these systems are used in business.

Learning Outcomes: On successful completion of this course, students are capable to:

- Explain the role of business software systems in professional environments.
- Understand the purpose of business software systems, especially enterprise resource planning (ERP) systems.

- Clarify the role of business processes and apply basic techniques about modeling of business processes.
- Know the core business processes supported by business software systems.
- Understand the term Business Intelligence (BI) and know the components of a business intelligence system.

Description:

Overview of business software systems:

- Introduction: business software systems, especially ERP-Systems, as basis of enterprise information processing.
- Classification of business software systems.
- History, market review of ERP systems.
- Architectures of ERP systems (client-server-architectures, layer models, interfaces).

Business Process Management (BPM) and its relation to business software systems:

- BPM cycle.
- Modeling Techniques: BPMN, ARIS.

Main business processes and their support by business software systems:

- Finance and accounting.
- Manufacturing.
- Supply Chain Management.
- Customer Relationship Management.

Management information systems / Business Intelligence:

Case studies are analyzed in the class and assignments with practical exercises with BPM tools are given to the students.

Lecturer(s): Prof. Andreas Lux

Prerequisites: —

Language: English

Teaching modality: Students must deliver all exercices and participate to all sessions.

Evaluation:

- **First time students**: written examination (70%) and control of homework/assignments (30%).
- **Repeating students**: written examination (100%).

Literature:

- Kenneth Laudon, Jane P. Laudon: Management Information Systems: Global Edition, 13th ed., Pearson Education, 2014.
- K. Ganesh et. al.: Enterprise Resource Planning - Fundamentals of Design and Implementation, Springer International Publishing, 2014.
- M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers: Fundamentals of Business Process Management, 2nd ed., Springer. 2018.
- B. Silver: BPMN Quick and Easy Using Method and Style: Process Mapping Guidelines and Examples Using the Business Process Modeling Standard, Cody-Cassidy Press. 2017

37. Circuits numériques [4 ECTS]

Objectif: Conception des circuits intégrés numériques à l'aide du langage VHDL et réalisation d'un petit microprocesseur simple sur un circuit programmable de type FPGA.

Learning Outcomes: Après avoir suivi ce cours, les étudiants seront capables :

- d'appliquer langage VHDL pour la réalisation d'un petit microprocesseur.

- de programmer un circuit programmable de type FPGA.

Contenu:

- Langage VHDL: définition du langage, concepts de base et méthodologie.
- Logiques combinatoires et séquentielles.
- Automates à états finis et modes de synchronisation.
- Définition et spécification d'un microprocesseur basique.
- Architecture de Von Neuman, constituants d'un CPU et jeu d'instructions ALU et arithmétique des ordinateurs.
- Pipeline, architecture de Harvard et RISC.
- Améliorations successives apportées à l'architecture de base.

Enseignant(s): Prof. Bernard Steenis

Prérequis: —

Langue: Français

Modalité enseignement: Cours théoriques et travaux pratiques (comprenant un projet de microprocesseur simple).

Modalités d'évaluation:
Un examen écrit ou oral (50%), rapport et présentation orale du projet réalisé en TP (50%).

Notes: Les étudiants devront obligatoirement participer aux TP et remettre les rapports qui leur sont demandés.

38. Cloud-Based Applications [4 ECTS]

Objectives: The course provides an introduction to both the theoretical foundations and practical applications concerning the broad area of "Cloud Computing".

The course covers both the practical development and deployment of cloud-based computing applications as well as current tools and respective application-programming interfaces (APIs) in the context of the Apache Hadoop and Spark ecosystems (cloud-based "big data" processing).

Learning Outcomes: On successful completion of this course, students are capable to:

- Explain both the theoretical foundations and the practical usage of current cloud-computing architectures.
- Explain concepts on virtualization and application deployment into the cloud.
- Describe how different concepts concerning the modeling and management of large data collections are implemented on top of these architectures.
- Prove first-hand experience about usage of the introduced tools in AWS / Google Cloud.

Description:
The course will cover two important aspects of cloud-based applications, compute-centric and data-centric applications, in particular the following topics:

- Infrastructures for Cloud Computing.
- Virtualization vs Containers - Docker.
- Container orchestration with Kubernetes.
- Practical Cloud examples: AWS, Google Cloud, RedHat OpenShift.
- Distributed file systems (GFS & HDFS).
- Distributed computing principles (MapReduce), replication, fault tolerance, backup tasks, custom combiners and partitioners, local aggregation, linear scalability.
- Apache Pig: first dataflow language (Pig Latin), translation into MapReduce and optimization.
- Apache HBase: distributed key-value store for very large tabular data, columns and column families, indexing and lookups.

- Apache Hive: SQL-like query language on top of Hadoop, translation into MapReduce.
- MongoDB: API overview, JSON processing, user-defined functions.
- Apache Spark: distributed resilient data objects (RDDs), overview of streaming and machine-learning extensions.
- An introduction to High Performance Computing (HPC).

Lecturer(s): Prof. Volker Müller

Prerequisites: —

Language: English

Evaluation:

- **First time students:** 2 graded exercises (15% each) and final written exam (70%).
- **Repeating students:** final written exam (100%).

Literature: The course will rely on genuine documentation available on the web, which will be made available on the course Moodle website.

39. Design Patterns [4 ECTS]

Objectives: The course introduces students to the concepts and the best practices of software engineering centred on Design Patterns. Several individual Design Patterns and some of their possible combinations will be discussed in detail from both theoretical and practical points of view, further familiarizing students with the underlying ideas. The course focuses on the Java programming language in examples and for implementation work.

Learning Outcomes: On successful completion of the course, students are capable to:

- Explain best practices of software engineering centred on Design Patterns.

- Apply several individual design patterns and their combination in practice.
- Explain the underlying ideas of specific design patterns.

Description:

In the lectures, the following Design Patterns are explained, grouped with respect to usage themes:

- Theme #1: Creation of objects:
  - Singleton.
  - Flyweight.
  - Prototype.
  - Builder.
  - Factory method.
- Theme #2: Hierarchical structures:
  - Composite.
  - Visitor.
- Theme #3: Events and notifications:
  - Observer.
  - Chain of responsibility.
- Theme #4: Coordinating objects at run time:
  - Command.
  - Mediator.
  - Strategy.
- Theme #5: Connecting objects:
  - Facade.
  - Adapter.
  - Proxy..

The supervised exercises sessions are devoted to:

- the implementation of individual Design Patterns.

- the composition of multiple Design Patterns into the same code.
- the re-engineering of existing code with the help of Design Patterns.

Lecturer(s): Prof. Denis Zampunieris, Dr. Laurent Debrauwer

Prerequisites: Programming 1 and Programming 2

Language: English

Teaching modality: Interleaved sequence of lectures and supervised practical sessions. Students must participate to all practical sessions.

Evaluation:

- **First time students**: Continuous assessment during the practical sessions (30%) and final written examination (70%).
- **Repeating students**: final written examination (100%).

Literature:

- "Design patterns: elements of reusable object-oriented software", E. Gamma, R. Helm, R. Johnson & J. Vissides, Addison-Wesley.
- "Design Patterns for dummies", Steve Holzner, Wiley Publishing.
- "Design Patterns pour Java", Laurent Debrauwer, ENI.
- "Design Patterns in Java", Steven J. Metsker & William C. Wake, Addison-Wesley.
- "Head First Design Patterns", Eric Freeman & Elisabeth Freeman, O'Reilly, 2004.
- "Object-Oriented Software Construction", Bertrand Meyer, Prentice Hall. [version française : "Conception et programmation orientées objet", Eyrolles].

40. Digital Transformation [4 ECTS]

Objectives:

1. **Historical Context:** Introduce the evolution of the digital revolution and its economic and societal impacts.
2. **Technological Understanding:** Provide in-depth insights into key technologies driving digital transformation (AI, Blockchain, etc.).
3. **Strategic Application:** Develop the ability to design and evaluate digital transformation initiatives for businesses.
4. **Futuristic Outlook:** Explore the next wave of digital technologies (Quantum Computing, AGI) and analyze potential risks for humanity.
5. **Practical Experience:** Enable students to apply concepts through a final project that addresses real organizational challenges.

Learning Outcomes: By the end of this course, students will be able to:

- **Analyze** the historical progression of digital technologies and articulate their influence on business models.
- **Evaluate** how data and emerging technologies can optimize and transform organizational processes.
- **Design** a digital transformation strategy aligned with business objectives, leveraging relevant tools.
- **Critically Assess** ethical, social, and human risks associated with advanced technologies.
- **Collaborate** effectively in teams to produce a comprehensive written report and deliver an engaging presentation on a proposed digital transformation plan.

Description:

This course provides a comprehensive exploration of digital transformation, centering on the technologies and strategic approaches that organizations adopt to thrive in today's evolving digital landscape. Students begin by examining the origins of the digital revolution to understand the economic and societal shifts that paved the way for current innovations. From there, the curriculum delves into how data-driven methodologies revolutionize business processes, enabling greater efficiency and competitive advantage.

Key technologies such as Artificial Intelligence (AI) and Blockchain will be discussed in depth, illustrating how they are applied across various industries. The course also highlights emerging fields like Quantum Computing and Artificial General Intelligence (AGI), considering both their transformative potential and associated risks. Real-world case studies, featuring both successful and failed digital transformation strategies—will demonstrate the importance of sound planning, change management, and stakeholder engagement.

A guest speaker from industry will offer practical insights, sharing lessons learned from an ongoing digital transformation journey. Building on all these insights, students will collaborate on a final group project where they propose a comprehensive digital transformation strategy for a real or fictional company, integrating the technological and managerial perspectives covered in the class.

Lecturer(s): Prof. Raphael Franck

Prerequisites: —

Language: English

Evaluation:

Final Project (50%): Group Work (2 - 3 students)

- **Written Report**: A thorough digital transformation strategy for a chosen real or fictional company
- **Presentation**: 10 - 15 minutes in class, followed by Q&A

Multiple Choice Exam (50%): Covering the topics seen in class (closed book, paper & pencil)

Literature:

- The Digital Transformation Roadmap: Rebuild Your Organization for Continuous Change, David L. Rogers, 2023, ISBN 978-0231196581
- Digital Transformation: Survive and Thrive in an Era of Mass Extinction, Thomas M. Siebel, 2019, ISBN 978-0231196581
- The Economics of Data, Analytics, and Digital Transformation: The theorems, laws, and empowerments to guide your organization's digital transformation, Bill Schmarzo, 2020, ISBN 978-1800561410

41. Generative AI [4 ECTS]

Objectives: This course introduces students to the field of Generative AI and LLM-based (Large Language Model) agents. The course will explain key concepts such as neural networks, transformers, RAG (Retrieval-Augmented Generation) systems, diffusion models, agentic AI and how to build and evaluate generative AI applications responsibly. Students will be guided through hands-on practical sessions using state-of-the-art tools and frameworks (e.g., PyTorch, Langchain, vector databases) and will learn best practices for designing, training, tuning, and deploying large-scale AI systems.

Learning Outcomes: At the end of the course, students will be able to:

- Understand the history, current impact, and real-world applications of Generative AI and LLM-based agents.
- Understand core concepts of NLP, including embeddings, vector representations, semantic similarities, and key ML paradigms (supervised, unsupervised, reinforcement learning).
- Describe how neural networks are trained, including essential knowledge of PyTorch (tensor ops, autograd).
- Demonstrate proficiency with transformers (attention, tokenization, pruning, distillation) for language tasks.
- Engineer effective prompts for generative models, mitigating limitations and hallucinations.
- Build, fine-tune, and evaluate RAG systems using vector databases.
- Fine-tune diffusion models and GANs for generative tasks.
- Analyze ethical, socio-economic, and security implications of deploying and maintaining GenAI solutions.
- Apply best practices when creating LLM-based agents with reasoning and tool usage.
- Evaluate GenAI/ML models using standardized metrics, procedures, and datasets.

Description:

- Introduction to Generative AI, LLMs, and course structure.
- Python basics and AI environment setup.
- Key machine learning concepts and data importance.
- Intro to PyTorch: tensors, autograd, and GPU use.
- Neural networks and training fundamentals.
- Building and training a basic neural network in PyTorch.
- NLP basics and word embeddings (word2vec, GloVe).
- Prompt engineering with OLLAMA and handling limitations.
- Transformer architecture and model optimization.
- Building a RAG (Retrieval-Augmented Generation) system.
- Fine tuning vs. instruction tuning of LLMs.
- Running a basic fine-tuning job on a pre-trained LLM.
- Generative models: GANs and diffusion models.
- Tuning parameters in a Stable Diffusion demo.
- Lifecycle of a generative AI project and cost trade-offs.
- Building a chatbot with LangChain (hands-on).
- Understanding LLM-based agents and tool use.
- Advanced prompt engineering and reasoning workflows.
- Evaluation metrics for generative AI models.
- Hands-on model evaluation and monitoring in production.
- Ethics and socio-economic impact of generative AI.
- Security in GenAI: prompt injection and jailbreaking (Part I).
- GenAI threats and deployment hardening strategies (Part II).
- Practical security lab: securing an LLM-based system.
- Real-world GenAI case studies across industries.

- Mini-project: advanced RAG and agent integration.
- Future of AI, AGI, and open research problems.
- Final project presentations and peer feedback.

Lecturer(s): Dr. Jérôme Francois, Dr. Antonio Ken Iannillo, Dr. Jorge Augusto Meira, Dr. Lama Sleem

Prerequisites: —

Language: English

Evaluation:

- **First time students:**
  - Take-home assignment: Implement a mini GenAI application (e.g. chatbot, RAG system, ...) and provide a concise report documentation design, approach, and evaluation (40%).
  - Final written exam (60%).
  - Active participation (up to 2 bonus points).
- **Repeating students:** final written exam (100%).

Literature: Will be announced during the course via Moodle.

42. Introduction to IOT [4 ECTS]

Objectives: Students will be equipped with competences for the development of applications for the Internet of Things (IOT):

- Know the challenges of IoT networks considering the various specificities and application constraints (e.g. QoS, scalability, real-time application, operational safety, etc.).
- Master the basics of IoT communication protocols (Application layer).
- Master the selection and implementation of different IoT communication protocols.

- Design, analyze and criticize different data collection, storage and processing data architectures, both their possibilities and their limits.
- Design one or more IoT applications using machine learning modules based on the collected data.

Learning Outcomes: On successful completion of this course, students are capable to:

- Define and develop the functional and technical specifications of a network and telecom equipment (hardware, software, implementation, etc.).
- Determine architecture components, technologies, equipment, tools supports and integrate them according to the specifications.

Description:

- Introduction to issues relating to the interoperability of IOT networks / protocols (application layer - OSI).
- Real-time networks: Profibus, Modbus, Modbus-TCP.
- IOT protocols on application layer: HTTP (REST API), MQTT, CoAP, OneM2M, O-MI/O-DF.
- NOSQL databases: MongoDB, ElasticSearch.
- Getting started with Node-Red (visual programming tool - open source - developed by IBM) for IoT application development:
  - data collection: Arduino, & sensors, Cloud API endpoints...
  - data storage: databases (SQL, NoSQL).
  - data treatment: Node-Red (JavaScript).
  - publication of data via dashboard (H2M) and machine interfaces (M2M): Implementation of an HTTP server (REST API specification, server deployment).

Lecturer(s): Dr. Sylvain Kubler

Prerequisites: Basic knowledge on networks and telecommunication (OSI model, network protocols), and some background in programming (Python, Java).

Language: English

Teaching modality: Lectures and practical exercises.

Evaluation:

- **First time students:**
  - 40%: practical group project.
  - 60%: final written exam.
- **Repeating students:** 100% final written exam.

Literature: Relevant literature will be provided during the lecture.

Notes: This course is shared with the *Bachelor in Computer Science (BiCS).*

43. Introduction to IT Security [4 ECTS]

Objectives: The general goal of the course is to increase awareness for IT security. The course explains a broad range of common algorithms and techniques used in IT security including IT security management.

By the end of the course, the students will have:

- A good understanding of the role of security and the importance of security in IT systems.
- A good understanding of security properties and the mechanisms used to enforce such properties.
- Familiarity with techniques for modelling and evaluating secure systems against given security requirements.
- A vision of the management of the IT securities and the related regulatory that we need to manage as IT responsible of a firm.

Learning Outcomes: On successful completion of this course, students are capable to:

- Explain the role of security and the importance of security in IT systems.
- Describe security properties and the mechanisms used to enforce such properties.
- Apply techniques for modelling and evaluating secure systems against given security requirements.
- Describe common techniques used for the management of IT security.

Description:

Introduction – IT security concepts:

1. Overview.
2. Key concepts and definitions (confidentiality, integrity, authentication, privacy, security dilemmas).

First part:

1. Introduction to basic cryptographic tools (symmetric/asymmetric cryptography, hash functions, signatures).
2. Identification and authentication.
3. Access control and security models: Authorization, policies and enforcement mechanisms.
4. Database and datacenter securities and kind of attacks:
   (a) Malicious software.
   (b) Trojans.
   (c) DOS.
5. Intrusion detection and firewalls.
6. Software and Network securities and Trusted systems:
   (a) Web security.
   (b) Email security.

(c) Security protocols.

Second part: Administering security:

1. Cybersecurity policies and risk management approaches:
    (a) Risk management and policies according with ISO 27K.
    (b) IT security management.
    (c) Data privacy management.

Selection of topics from: Digital forensics, Digital Rights Management, Trust management, RFID security.

Lecturer(s): Dr. Daniel Mathieu

Prerequisites: Basic knowledge of computer languages, operating systems and computer networks.

Language: English

Evaluation:

- **First time students**: written exam (60%) and practical exercises (40%).
- **Repeating students**: written exam (100%).

Literature:

- W. Stallings, L. Brown: Computer Security Principles and Practice, Pearson, 4th edition, 2018, ISBN: 978-0-13479435-8.
- B. Schneier: Applied Cryptography: Protocols, Algorithms, and Source Code in C, Wiley, 2nd edition, 1996, ISBN: 978-1-119-09672-6 .

44. Introduction to Machine Learning [4 ECTS]

Objectives: This course introduces Machine Learning (ML) principles and its three main learning paradigms (supervised, unsupervised, and reinforcement learning). For each learning paradigm, it presents some of its most typical foundational models and discuss them from the perspective of representation, evaluation, and optimization. A special attention is given to a basic introduction into deep learning techniques and generalization. The course mixes theoretical concepts with vanilla implementations of various ML models.

Learning Outcomes: At the end of the course the student will be able to:

- Define Machine Learning and differentiate the terms AI, Machine Learning, and Deep Learning.
- Describe the differences among the three Machine Learning paradigms: supervised, unsupervised, and reinforcement learning.
- Determine which of the three Machine Learning paradigms is appropriate to a particular type of problems.
- Derive, implement, and evaluate some of the most typical Machine Learning models.
- Derive, implement, and evaluate some basic Deep Learning models and their learning algorithms.
- Explain proper ML evaluation procedures, including the differences between training and testing performance.
- Apply Machine Learning models to real-world problems.
- Identify overfitting in the context of a problem and describe solutions to overfitting.
- Evaluate the performance of a Machine Learning algorithm on a real-world dataset.

Description:

The following topics are covered in the course:

- Basics - ML Introduction.
- Basics - Data preparation for ML.
- Supervised Learning - Regression.
- Supervised Learning - Classification.
- Unsupervised Learning - Dimensionality reduction.

- Unsupervised Learning - Clustering.
- Reinforcement Learning - Preliminaries.
- Reinforcement Learning - Basic methods.
- Deep Learning - Learning Deep Representations.
- Deep Learning - Models.
- Deep Learning - Deep Reinforcement Learning.
- Generalization.
- Research - Glimpse on state-of-the-art research.
- Engineering - ML and the real-world.

Lecturer(s): Prof. Decebal Mocanu, Mr. Christopher Gadzinski

Prerequisites: —

Language: English

Teaching modality: A lecture and directed work session every other week.

Evaluation:

- **First time students**: two take-home assignments for 10% and one take-home assignment for 20% of the final grade, written exam: 60% of the final grade, open-book exam (i.e., course material allowed).
- **Repeating students**: final written exam (100%).

Literature:

- Bishop, C.: Pattern Recognition and Machine Learning, 2006
- Goodfellow, I., Bengio, Y. & Courville, A.: Deep Learning, 2016
- Sutton, R., Barto, A.: Reinforcement Learning: An Introduction, second edition, 2018
- Hastie, T, Tibshirani, R. & Friedman, J.: The Elements of Statistical Learning, 2009

- Zhang, A., Smola, A.J., Lipton, Z., Li, M.: Dive into Deep Learning, 2023

45. IT in a Real Financial Organization [4 ECTS]

Objectives: The students will learn how IT is structured, driven and operated in a real financial organization - based on an example of a bank in Luxembourg. The course will cover different IT job families, IT project governance and lifecycle, practical hints for system design, insights into the hiring and job onboarding process.

Learning Outcomes: On successful completion of this course, students are capable to:

- Understand the place, the role, the priorities and the driving factors of IT in a real organization.
- Differentiate between different IT job families.
- Understand how IT projects are structured and executed.
- Implement practical system designs.
- Navigate through the hiring and job onboarding process.

Description:

- A structure of a typical bank in Luxembourg and a place of IT in it.
- IT job families – from a software engineer to quality assurance, system analysis, user experience and project management.
- IT project governance and project execution.
- System design – DOs and DON'Ts.
- Hiring process – job search, interview preparation, interview types (live coding, brain teaser, tech vs HR), interview follow-up, first days at job.

Lecturer(s): Mr. Rustev Nasyrov

Prerequisites: In-progress (3rd year and above) Computer Science university education

Language: English

Teaching modality: Lectures with a final evaluation lab team-work

Evaluation:

First time students:

- End-course evaluation lab teamwork (30%): Design and build a demo system
  - Structure project: scope, governance, timelines, planning.
  - Execution phases: launch, planning, analysis, design, implementation, deployment, landing.
  - Project roles and their responsibilities: lead, analyst, engineers etc.
- Individual student presentation (30%).
- Final written exam (30%).
- Participation in course (10%).

Repeating students:

- Individual student presentation (50%).
- Final written exam (50%).

Literature: Lecture slides and supporting materials will be provided on the Moodle course website.

46. Java for Enterprise Applications [4 ECTS]

Objectives: Students will learn how Jakarta EE (formerly Java EE) offers a component-based approach to designing, developing, assembling, and deploying enterprise applications. Both theoretical and practical aspects of Jakarta EE components will be explored, with particular emphasis on the role of web services in enabling service-oriented architectures. Hands-on experience will be gained using WildFly, a free Jakarta EE application server closely related to the popular commercial JBoss server. In the second part, students will receive a practical comprehensive introduction to the competitive Spring Boot framework.

Learning Outcomes: On successful completion of this course, students are capable to:

- Explain the main ideas behind the Jakarta EE framework.
- Develop views with the help of JSF and named beans.
- Explain the practical significance of the most important APIs in the framework.
- Develop Jakarta EE-based applications of medium-level difficulty with the application server Wildfly.
- Implement web services and web applications with Spring Boot.
- Extend their knowledge on Jakarta EE and Spring Boot by autonomously reading the specifications and other documentation available.

Description:

- Introduction to the Docker containerization framework and its usage in this course.
- General Jakarta EE concepts.
- Traditional Java-based web applications: Java Servlets / Java Server Pages (JSP) / Custom tags / JSTL.
- Java Server Faces and Named Beans as view technology in Jakarta EE.
- Session Beans: stateless, stateful, and singleton beans.
- Context Dependency Injection (CDI) and its benefits.
- The Java Persistence API.
- Dedicated APIs of the Jakarta EE framework (XML, JSON, batch jobs, ...).

- The Java Message Service (JMS) and Message-Driven Beans.
- Develop web services with Jakarta EE.
- Introduction to Jakarta EE Security.
- Introduction to Spring Boot, especially REST web services, Spring Boot persistence, and MVC web applications with Spring Boot.
- Spring Boot with relational databases (MariaDB) and NOSQL databases (MongoDB, Redis)
- Spring Boot and GraphQL.
- Concepts for Spring Boot Security (Spring Boot app combined with a KeyCloak authentication server)
- Native enterprise applications with Spring Native, Quarkus, and Micronaut.
- Comparison of advantages and disadvantages of Jakarta EE and Spring Boot.

Lecturer(s): Prof. Volker Müller

Prerequisites: Advanced knowledge of the Java programming language.

Language: English

Teaching modality: Lectures and practical (partly graded) homework. The practical homework will be done within a provided Docker container-based environment using the free Wildfly application server (1st part) and with Spring Boot (2nd part).

Evaluation:

- **First time students:** two graded assignments (30%) and final written exam (70%)
- **Repeating students:** final written exam (100%).

Literature: The course mainly uses the various Jakarta EE component specifications available online. A detailed list of references is made available on the Moodle course website.

47. Parallel and Distributed Systems [4 ECTS]

Objectives: Many applications and systems extend beyond the boundaries of a single process, often being deployed on multiple machines. The main objective is to provide a solid understanding of the theoretical background and general principles on how to compose a single coherent system composed of multiple distributed components. Modern system architectures are presented and discussed, including cloud computing, peer-to-peer, grid, ad-hoc and mobile systems. Diverse issues in terms of the design and implementation of such systems are explored, with a strong emphasis on practical aspects, including performance. Real-world distributed systems are examined as case studies.

Learning Outcomes: On successful completion of this course, students are capable to:

- Prove a solid understanding of the general principles used in distributed systems.
- Explain the usage of distributed systems in real-world situations.
- Practically realize a distributed system.

Description:

- Distributed system architectures.
- Time in distributed systems.
- Distributed processes.
- Interprocess communication.
- Distributed synchronisation.
- Fault tolerance (reliability, replication).
- Peer-to-peer systems.
- Grid computing.
- RPC, RDMA, MPI.
- Real-time streaming storage and processing.

- Partitioning, Scaling, Performance.
- Latency, Throughput.

Lecturer(s): Dr. Ovidiu-Cristian Marcu

Prerequisites:

- Programming knowledge (any of the following Java/C/C++).
- Basic knowledge of the Linux operating system.

Language: English

Teaching modality: This course employs a blended teaching model that combines traditional lectures with an interactive, inverted classroom approach to balance theoretical foundations with practical application.

Foundational Lectures: Approximately half of the sessions will be conducted as traditional lectures. These sessions are designed to introduce core theoretical concepts, fundamental principles of distributed systems, and architectural overviews. The instructor will present material, and students are encouraged to engage through questions and discussion.

Applied Inverted Sessions: The other half of the sessions will follow an inverted (flipped) classroom model.

Before Class: For these sessions, students will prepare by independently studying assigned materials, which may include academic papers, pre-recorded video lectures, or technical documentation.

In Class: The corresponding class time will be dedicated to hands-on, practical work. These sessions will function as guided labs and workshops where we collaboratively solve problems, develop/debug system components, and hold in-depth discussions on the prepared topics.

This hybrid structure ensures that foundational knowledge is delivered efficiently, freeing up valuable class time for the interactive, hands-on learning essential for mastering distributed systems.

Evaluation:
First time students:

- 40%: In-class practical labs: Four labs are designed to directly apply the concepts learned from both traditional lectures and pre-class materials. Evaluation is based on successful implementation, problem-solving approach, and demonstrated understanding during the session.

- 20%: Quizzes with written explanations: Four quizzes will assess understanding of key concepts from both lectures and preparatory materials. To evaluate depth of knowledge, students will be required to provide clear written explanations to justify their answers, demonstrating their reasoning process.
- 30%: Final written exam (mandatory).
- 10%: Active engagement in class.

Repeating students: 100% final written exam.

Literature:

- Distributed Systems, Tanenbaum as optional reading.
- Lecture notes and papers are based on MIT's 6.824

Details for Semester 6 Courses

Module 6.1: Bachelor Project

The final project is a full-time, individual IT undertaking lasting a minimum of 12 weeks. Students choose between two tracks: the Professional Internship, for those seeking hands-on experience at a partner organization, or the Research-Based Bachelor Project, designed for students pursuing a research-oriented path and planning to continue with a Master's degree. Participants join a team within the partner institution or research group, applying their IT knowledge to a practical subproject. Upon completion, students are required to write and defend a Bachelor thesis that documents the project scope, outcomes, and practical insights gained.

After validation of this module, students are capable to:

- Reflect critically on their practical experiences gained through a real-world project in a professional setting.
- Make informed decisions about their future, whether pursuing a Master's degree or entering the professional workforce.

Compensation

This module does not allow internal compensation between different courses (*Règlement des études*, Art. 35).

48. Research-based Bachelor Project [30 ECTS]

Objectives: The main objective of the research-based Bachelor project is the practical application of the learned technical expertise and a professional attitude that the students have acquired during their BINFO study by participating within a research project done in a research group of the Department of Computer Science at the University of Luxembourg. The technical aspects and results of the project and the experiences made during the project must be documented in a report, the **Bachelor thesis**.

Learning Outcomes: On successful completion of this course, students are capable to:

- Integrate into a research team working in the IT domain of an academic institution;
- Show maturity in working in the IT research domain;
- Prove the acquired expertise for a practical problem in the research-related information technology context;
- Reflect on the experience made during this practical work.

Description:
In the Bachelor project, students shall apply the technical and interpersonal expertise achieved during their previous study within a research-based project of at least 12 weeks, done in a research group at the Department of Computer Science. The project must contain a substantial research contribution from the student. The Bachelor project should prepare students for continuation of their study within a CS Master programme. The three key elements of a Bachelor project are:

- An academic supervisor and the study director must support the acceptance of the student, based on prior academic results.
- The Bachelor project and the expected research question must be clearly defined before the start of the project and feasible for a 12-weeks long work.
- The student must be mentored by an "academic supervisor" (and possibly a local supervisor) from within the department that supports the scientific learning process of the student in the context of the project.
- The work done during the Bachelor project must lead to a Bachelor thesis, a written report that documents the complete work done and describes the technical background, all IT related aspects (including problem analysis and design, implementation aspects, ...) and the achieved results. This report will be evaluated by a jury.

Note that a research-based Bachelor project is an unpaid academic activity of the student.

Lecturer(s): Prof. Volker Müller

Prerequisites: To be eligible for the Bachelor project, the student must:

1. be enrolled at the University of Luxembourg for the semester in question.
2. have successfully completed all modules of the first two semesters of the program (in case, successful completion of only a single course of the first two semesters is missing, the Study Director can grant an exemption in case the student participates in the exam for this course during the last exam session before the expected start of the Bachelor project).
3. have validated at least 40 ECTS of courses in the other semester modules, such that in total courses for at least 100 ECTS are validated.
4. being accepted by the academic supervisor and the study director as a student member of the resp. research group (based on previous academic performance).
5. have met the mobility requirement (or have been exempted from this obligation).

Language: English

Evaluation:
The evaluation of the Bachelor project and the respective project report (the Bachelor thesis) is done before a **Bachelor project jury** based on the results achieved during the project. This jury consists of both the local and the academic supervisor and at least one additional member of the academic body.

49. Professional Internship [30 ECTS]

Objectives: The main objective of a final professional internship is the practical application of the learned technical expertise and a professional attitude that the students have acquired during their BINFO study by participating in an IT-project realized in a professional host institution. The technical aspects and results of the project and the experiences made during the project must be documented in a report, the **Bachelor thesis**.

Learning Outcomes: On successful completion of this course, students are capable to:

- Integrate into a team working in the IT domain of a professional host institution.
- Show maturity in working in the IT domain, at least equivalent to regular junior level staff.
- Prove the acquired expertise for a practical problem in the information technology context.
- Reflect on the experience made during this practical work.

Description:
In the professional internship, students shall apply the technical and interpersonal expertise achieved during their previous study within a practical project of at least 12 weeks with a close relationship to Information Technology. The realized project can consist of developing a new or extending an existing software product (e.g. a prototype / proof-of-concept, (part of) an application, an API, ...), a hardware-related development (e.g. prototype used in a feasibility study, a controller, a data-acquisition device,...), or a contribution to the management of an IT project (e.g. specifications of technical requirements, study of state-of-the art technology, study of suitable hardware / software for a future system). The project must contain a substantial contribution from the student. The three key elements of a Bachelor project are:

- The Bachelor project and the expected result must be clearly defined before the start of the internship and feasible for a 12-weeks long work.
- The student must be mentored by a "local supervisor" within the host institution. Additionally, an "academic supervisor" from within the university supports the scientific learning process of the student in the context of the project.
- The work done during the internship project must lead to a Bachelor thesis, a written report that documents the complete work done and

describes the technical background, all IT related aspects (including problem analysis and design, implementation aspects, ...) and the achieved results. This report will be evaluated by a jury.

Note that by the Luxembourgish labor law an internship student is obliged to follow the guidance of his/her superior. This type of internship must be paid.

Lecturer(s): Prof. Volker Müller

Prerequisites: To be eligible for the professional internship, the student must:

1. be enrolled at the University of Luxembourg for the semester in question.
2. have successfully completed all modules of the first two semesters of the program (in case, successful completion of only a single course of the first two semesters is missing, the Study Director can grant an exemption in case the student participates in the exam for this course during the last exam session before the expected start of the Bachelor project).
3. have validated at least 40 ECTS of courses in the other semester modules, such that in total courses for at least 100 ECTS are validated.
4. have met the mobility requirement (or have been exempted from this obligation).

Language: English

Evaluation:

The evaluation of the professional internship is done by a **internship jury** consisting of both the local and the academic supervisor of the internship project together with at least one additional member from the academic body. The jury evaluates the clearness and preciseness of the presentation and the quality of the answers given to the jury's questions.