University of Luxembourg

Faculty of Science, Technology and Medicine

Bachelor in Applied Information Technology – BINFO

https://binfo.uni.lu

Programme

Academic Year 2022-2023

Programme Director: A-Prof. Volker Müller

Version: 1.12.2022

Introduction

The "Bachelor in Applied Information Technology" at the University of Luxembourg (BINFO) offers an excellent, generalist education in information technology (IT), whose objectives are to give students operational skills that are relevant and desirable to potential employers and so allows a quick integration into the professional world, in either the private or the public sector. The BINFO provides students with the basic theoretical and applied knowledge in core IT areas like algorithms and data structures, databases, networks, software development in practice, mobile and web application programming, but also the practical thinking to apply these technologies in industry. This focus on an applied qualification combines theoretical components of a traditional study in computer science with a focused approach giving students real-world skills and applicable concepts geared towards their career path.

Beyond technical learning, BINFO prepares its graduates for the European labor market with various theoretical and applied trainings on soft skills like entrepreneurship, psychological aspects of team work, several practical team projects, and finally a one-semester practical IT-related project done either in an IT department of some Luxembourgish company (for more professionally interested students) or a research group in the CS department (for more research-oriented students). The programme is highly human-centered, with a strong international presence: a bilingual French / English curriculum, a mobility semester abroad, and teachers and fellows from different countries and cultures. BINFO graduates are optimally prepared to directly enter the labor market after their study, but also possess the basic foundations in computer science to continue their education with a Master study.

Admission to the BINFO programme is possible for every candidate with a university entrance degree, sufficient language background (level B1 in French and B2 in English), and motivation for learning about mathematics and computer science. Candidates with excellent mathematical background, but non-sufficient language skills, are nevertheless invited to apply, since specific language courses are offered in the Language Center of the University to help beginner students to improve their language skills.

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 basis for all academic programmes at the University of Luxembourg is defined in the following two documents (both in French language):

- Loi modifiée du 27 juin 2018 ayant pour objet l'organisation de l'Université du Luxembourg,
- Règlement des études du 5 mai 2020.

These documents also define the regulations and procedures that apply during the study. Both documents are available on the web page https://www.uni.lu/university/official_documents.

Mobility Semester

The aforementioned law on the university defines in Art 36 (6) that for Bachelor students a mobility semester in a foreign country is mandatory (with only few exceptions for students in special situations). The mobility semester for BINFO students can be chosen as either Semester 3 or Semester 4. For a student doing his/her mobility, the respective regular semester programme is replaced by a list of courses given at the host institution, defined together by the student and the study director before the start of the mobility semester. Technical details are available on a dedicated page on our Moodle platform.

Bachelor Project

In a final **Bachelor project** in the 6-th study semester students shall apply the technical and inter-personal 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. Additional- ly, 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 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. The Bachelor project must be done in one of two possible variants:
- A profession-oriented Bachelor project is done within a professional institution in Luxembourg. The student shall apply and extend his IT technical expertise and soft skills within in a professional environment. This variant of a Bachelor project is strongly recommended for students that plan to start a professional career directly after graduation.
- A research-oriented Bachelor project is done within a research group of the University of Luxembourg. The student shall apply and extend his IT scientific expertise for a research- focused question. This variant of a Bachelor project is only recommended for students that plan to continue their academic training within a Master programme.

The **Bachelor project defense** evaluates the achieved results of a Bachelor project. The evaluation is done by a jury that consists of both supervisors of the Bachelor project together with at least one more academic lecturer from the university. In the defense, both the scientific and technical quality of the Bachelor project and the professional appearance of the students are assessed. The defense consists of two parts:

- The student presents the work done during the bachelor project in a presentation of about 20 minutes.
- The student answers to questions by the jury members for about 30 minutes. These questions cover both the given presentation and the

provided Bachelor project report.

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

First Semester Programme

	CM	TD	ECTS
Module 1.1 (No module compensation)			**7**
Introduction à l'informatique	30	15	4
Technical English 1	30		3
Module 1.2 (Allows 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**
Technical English 2	15	15	2
Module 2.2 (Allows 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	15	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	**210**	**135**	**30**

Third Semester Programme

	CM	TD	ECTS
Module 3.1 (No module compensation)			**9**
Web Development 1: 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**
Database Management 1	30	15	4
Networks 1	30	15	4
TOTAL	**210**	**120**	**31**

Fourth Semester Programme

	CM	TD	ECTS
Module 4.1 (No module compensation)			**6**
Psychologie du travail en groupe	15	15	3
Droit pour informaticiens	30		3
Module 4.2 (No module compensation)			**8**
Networks 2	15	15	3
Software Engineering Project	10	45	5
Module 4.3 (No module compensation)			**8**
Algorithms 3	25	20	4
Database Management 2	30	15	4
Module 4.4 (No module compensation)			**8**
Interaction Design	25	20	4
Software Testing	37	20	4
TOTAL	**187**	**150**	**30**

Fifth Semester Programme

	CM	TD	ECTS
Module 5.1 (No module compensation)			**10**
Design Patterns	25	20	4
Introduction à la vie professionnelle	30		2
Web Development 2: Back-End	25	20	4
Module 5.2 (No module compensation)			
Students must choose at least five courses out of all optional courses.			**20**
Banking Information Technologies	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
Introduction to IT Security	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)			**30**
Bachelor Project			27
Bachelor Project Defense			3
TOTAL	**0**	**0**	**30**

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

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

TD: Total number of 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.

Remarks: One "hour" shown in the table corresponds to one teaching unit (*unité d'enseignement*) of 45 minutes. The given number of hours is the maximal number of hours for a course, which can change due to the academic calendar for 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 compensation or not. The study regulations with detailed explanations are available at https://www.uni.lu/university/official_documents.

Details for Semester 1 Courses

Module 1.1

Computer Science is one of the disciplines of modern science in which we study the various aspects of computer technologies, their development, and their applications in the present world. Information technology (IT) is a branch of engineering that deals with the use of computers and technology to retrieve, transmit, and store information. The objectives of this module are to provide a solid basis for the core fundamental concepts in computer science and IT, combined with a strong practical knowledge of the English language, which is the most dominant language used in the IT domain.

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 the 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, comme par ex. la représentation des nombres, des textes et des images, le fonctionnement de l'Internet. Être familier avec les différentes composantes d'un système informatique (couche matérielle, système d'exploitation, logiciel et langages, couche réseau et internet). Le cours s'appuie dans un premier temps sur une vision historique de l'informatique afin d'appréhender les contraintes issues de l'existant sur le développement de l'informatique d'aujourd'hui, il se poursuit en présentant les différentes notions permettant de comprendre un système informatique moderne.

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 faire des raisonnements simples sur des concepts informatiques;
- de décrire des procédés et techniques informatiques.

Contenu:

- Historique des techniques de traitement de l'information
- Système binaire
- Structure générale et fonctionnement des ordinateurs
- Représentation des nombres
- Représentation des caractères, des textes et des images

- Fonctionnement de l'Internet
- L'ingénierie moderne du logiciel

Enseignant(s): Dr. Mathieu Jimenez, M. Fabien Bernier

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 1 [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 a language training programme.

Language: English

Evaluation: Continuous evaluations during the course (50%) and a final written exam (50%).

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

Module 1.2

The theoretical foundations of computer science are build upon mathematics. A basic mathematical background is therefore a strong requirement for the study of many theoretical and also applied areas in computer science, and as such, it will be essential for successfully following courses on more applied techniques in the coming semesters. The objective of this module is therefore to provide sufficient knowledge of basic essential mathematics and about the application of mathematical support tools, with 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 on 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 intends that all students have sufficient knowledge of basic calculus, especially on the analysis of elementary real functions in one variable and on the usage 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: One single final exam (100 %).

Literature: Franck Leprévost: What counts? A Hands-On Tutorial on Calculus, Ed. Amazon (2022).

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): Dr. Baptiste Lambin, Dr. Guillaume Maillard, Dr. Bruno Teheux

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: Un contrôle des connaissances pratiques (50%) et un examen écrit (50%).

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

Strong programming skills are an essential foundation for every applied computer scientist. Closely related is a solid understanding of fundamental concepts used in operating systems. In this module, students will acquire their first competences in programming with the introduction of the Java programming language, in combination with important concepts on how operating systems are managing resources and operations in a computer. The topics of both courses in this module will be further extended in coming 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 programming language Java, 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,
- use shell commands and basic scripting in Linux.

Description:

- Introduction to OS (Windows, Linux, MacOS)
- Processes: management and scheduling
- User and Rights management
- File Systems
- Basic of system administration, lab exercises, scripting, virtualisation
- Mobile OS (Android, iOS)

Lecturer(s): Dr. Christian Grévisse, Mr. Panissara Thanapol

Prerequisites: —

Language: English

Teaching modality: Lectures and lab exercises.

Evaluation:

- Midterm exam (40%)
- Final exam (60%)

Literature: Books and other inputs given in the lecture.

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, Dr. Ali Osman Topal, Mr. Aryobarzan Atashpendar, Mr. Stéven Picard

Prerequisites: —

Language: English

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

Evaluation: **Winter semester:**

- First session students: two practical exams (25%+25%) + final written exam (50%)
- Resitting students: final written exam (100%)

Summer semester: Final written 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

Firm IT Project management competences are a core expertise for a professional in the applied IT domain. The objectives of this module are the improvement of such competencies important for a future IT expert, with focus on technical discussions and presentations in English and proficiency in IT project management.

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;
- apply the fundamental concepts, the proceedings, and appropriate tools for the management of IT projects;
- analyse the different dimensions of the problematics of project management for IT projects.

Compensation

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

8. Technical English 2 [2 ECTS]

Objectives: Review the English grammar and develop composition and presentation skills in English. In addition, we introduce more technical vocabulary from the IT domain.

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

- understand courses and presentations given in English language.
- read and understand technical documents from the IT domain.
- show skills in giving presentations in English.

Description:

- Oxford Grammar and Listening Tests and diagnostic assessment (first session)
- Writing three short papers based on a class project
- Writing one oral presentation using Powerpoint covering the class project

Lecturer(s): Mr. Matthew Kahn

Prerequisites: Technical English 1

Language: English

Teaching modality:

1. Weekly class meetings
2. Grammar review (based on Oxford diagnostics)
3. Pair works on drafting papers
4. Pair works on drafting papers
5. Team work on class project
6. Peer marking of oral presentation
7. Participation: Maximum three unexcused absences

Evaluation: The final mark is based on the three papers (30 pts), one oral presentation (10 pts), participation to the course (10 pts), and the final examination (20 pts).

Literature:

1. Swan, Michael. Oxford English Grammar Course: Basic, Oxford Univ. Press, 2011, ISBN-10: 0194420779
2. Readings in Extreme Programming
3. Reading SDK for mobile phone applications
4. Reading tutorials for mobile phone applications

Notes: Students must participate in all weekly class meetings and complete three papers and one oral presentation.

Module 2.2

The objective of this module is the further expansion of the mathematical background of students to provide them with sufficient knowledge in the mathematical domain required for experts in applied information technology. In particular, the mathematical basics of probability theory which play an important role in many practical IT applications are covered.

After validation of this module, students are capable to:

- explain mathematical definitions and properties of vector spaces, matrices, formal languages, finite state machines, graph theory, and probabilities;
- 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;
- 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: —

Language: English

Teaching modality: Weekly class meetings including exercise sessions and weekly homework.

Evaluation: Final written exam (100%).

Literature: Franck Leprévost: "Order Matters! A Hands-On Tutorial on Linear Algebra", available on Amazon, 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/pˆdZ 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 φ-function
- (Z/nZ)ˆ∗ and φ(n)
- Exercises : Computing in Z/pˆd 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: Mathématiques discrètes 1

Language: English, French

Evaluation: Final exam (100%).

Literature: Franck Léprevost: "How Big is Big? How Fast is Fast? A Hands-On Tutorial on Mathematics of Computation", ISBN 9798642630556 - edited by Amazon

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: La note finale est composée :

- à 25% des notes obtenues pour les travaux personnels réalisés à domicile, et
- à 75% de la note obtenue à l'examen final.

Module 2.3

Programming skills are closely related with sufficient knowledge on algorithms and data structures to solve standard algorithmic problems. The objectives of this module are the further improvement of competences in programming with special focus on object-oriented programming techniques together with a systematic applied study of algorithms and data structures for several standard problems in computer science.

After validation of this module, students are capable to:

- explain the fundamental concepts of algorithms and object-oriented programming;
- analyse a problem of medium complexity to transform it into an algorithm and implement the problem in the object-oriented programming

language Java, 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: Familiarize students with algorithms and fundamental data structures and with the analysis of their complexity.

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

- solve problems computationally by developing a suitable algorithm and corresponding data structure;
- analyze the complexity of an existing algorithm;
- extend their algorithmic background independently using existing literature.

Description:

- Concept of algorithm;
- Complexity of algorithms;
- Sorting algorithms ;
- Mathematical background;

- Data structures;
- Hashing;
- Graph algorithms;
- Classification of algorithms;
- Complexity of problems.

Lecturer(s): Prof. Pierre Kelsen, Mrs. Monica Patricia Arenas Correa

Prerequisites: Programming 1, Mathématiques discrètes 1

Language: English

Evaluation: Evaluation of homeworks and practicals (50%), final written exam (50%).

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

13. Programming 2 [6 ECTS]

Objectives: This course introduces advanced notions of object-orientated 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. Aryobarzan Atashpendar, Mr. Sahar Niknam

Prerequisites: Programming 1

Language: English

Evaluation: **Summer semester:**

- First session students: software project (40%) + final written exam (60%)
- Resitting students: final written exam (100%)

Winter semester: final written exam (100%)

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

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

Module 2.4

Numerous different research areas exist in computer science. This module starts the introduction of dedicated IT-related topics with significance for practical applications. Graphics and data acquisition are core building blocks used in many IT tools and technologies. The objectives of this module is to provide an introduction to these two topics.

After validation of this module, students are capable to:

- explain the concepts and fundamental processes relative to data acquisition and processing;
- apply the basic principles of data transmission (encoding, modulation, multiplexing, routing) in low-level networked applications;
- explain the basic concepts in 2D and 3D graphics;
- use practical tools for graphics to perform basic graphical operations on 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, or the Java 2D API.

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 or Inkscape for simple image processing tasks.
- develop simple graphics-related programs with the help of the Java 2D API or JavaFX.

Description:

- Raster versus vector graphic
- Two-dimensional graphics:
  - Pixel, coordinate systems, colors
  - Basic graphical Shapes: line segments, circles, ellipsis, polygons
  - Affine transforms like translation, rotation, scaling, shearing
  - Image manipulation with convolution filters
  - Image manipulation based on statistical information
- Practical introduction to the Java 2D API 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

Lecturer(s): Prof. Volker Müller

Prerequisites: —

Language: English

Teaching modality: 2 hrs lecture per week and practical exercises (homework).

Evaluation: Final exam (50%), 4 graded exercises (40%), regular participation (10%).

Literature: Main 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 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
- NumPy basics: Arrays and Vectorized Computation
- Data Acquisition, Preparation and Management
- Data Visualization

- Time Series
- Introduction to Modeling Libraries in Python
- Data Analysis Examples

Lecturer(s): Dr. Ali Osman Topal

Prerequisites: —

Language: English

Teaching modality: Lectures and practical exercises

Evaluation:

- Quiz – 20%
- Homework and practical – 30%
- Final Exam (Practical and Written) – 50%

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 course web site.

Details for Semester 3 Courses

Module 3.1

An important aspect of modern computer applications, especially mobile applications, is a user-friendly interface for input and output of information. This module extends the previous modules on programming with information about the development of graphical user interfaces and closely related programming paradigms like event-driven programming and multi-threading. Another non-technical, nevertheless important, aspect of a solid programming background is basic knowledge about the legal situation of code development and code reuse, including the new legislation on data privacy.

After validation of this module, students are capable to:

- explain the concepts and principles in law applicable to the development of computer code or more generally computer science;
- discuss a concrete situation to determine the legal constraints that have to be respected by a computer scientist;
- use frameworks to develop graphical user interfaces for desktop computer or mobile devices;
- describe programming paradigms important for the development of mobile applications.

Compensation

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

16. Web Development 1: Front-End [5 ECTS]

Objectives: 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.

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.

1. HTML: The structural layer of the Web
2. History - web browser wars, architectures, ACID test
    - Document types
        - XML, DTDs, quirks mode
    - Markup notation
        - tags, microformats, accessibility
3. CSS: The presentation layer of the web
    - CSS Object Model

  - box model, selectors, specificity levels

- Standards & Preprocessors
  - Less, Sass, BEM

- Frameworks
  - Bootstrap, Material UI

4. JavaScript: The behavioral layer of the Web
   - Document Object Model
     - parsers, implementations
     - ES5, ES6, typescript
     - jQuery, mootools

5. UI design fundamentals
   - Layout
     - navigation, forms

   - Responsive web design
     - media queries

   - Callbacks
     - event-driven programming

6. Database integration
   - Transport protocols
     - JSON, XML, text

   - Serverless
     - RESTful APIs, SPAs

7. Tooling
   - Command line interface
     - CLI fundamentals

   - Dependency management
     - bower, browserify, webpack

   - Building systems

  - grunt, gulp

- Version control
  - git, subversion

8. Web app frameworks
   - Fundamentals
     - N-way binding, MVC, virtual DOM
   - Examples
     - Angular, Vue, React
9. Performance
   - Minifiers
     - uglifyjs
   - Monitoring
     - inspectors, debuggers
   - Auditing
     - PageSpeed, Lighthouse
10. Testing
    - Fundamentals
      - unit testing, integration testing, end-to-end testing

Lecturer(s): Prof. Luis Leiva, Mr. Kayhan Latifzadeh

Prerequisites: —

Language: English

Evaluation:

- Coding exercises: 50%
- Final exam (multiple-choice quiz): 50%

Redoing session:

The final exam can be re-taken in the next exam session. Grades from the coding exercises and final project will be retained until the student passes the course.

Literature: **Reference texbook:**

1. J. Robbins. 2012. Learning Web Design. O'Reilly Media, 4th ed.

Recommended books:

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

17. Programming 3 [4 ECTS]

Objectives: The aim of this course is to familiarize the students with the basics of graphical user interface (GUI) programming with two different popular frameworks and related programming paradigms like event-driven programming and multi-threading.

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

- develop event-driven programs for Swing-based desktop applications or Android mobile phones;
- apply multi-threading in programming tasks;
- explain design patterns commonly used for mobile application development;
- extend their knowledge on relevant techniques in mobile programming independently with literature research.

Description: This course addresses the theory and practice of graphical user interface (GUI) programming. Topics include:

- event-driven programming,
- multi-threading, and
- related design patterns (Model-View-Controller, ...).

We examine various practical examples in

- **Java Swing** for desktop applications, and
- **Android** for mobile phones.

Lecturer(s): Dr. Christian Franck

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

Language: English

Teaching modality: The course is held according to the flipped classroom model. Students have to read documents or view short video lectures at home, and the time in class is mainly devoted to discussions, exercises and projects.

Evaluation: Final exam (100%).

Literature: Online documentation of the different toolkits.

Module 3.2

Detailed knowledge about programming languages is not sufficient for a skilled programmer, but solid competencies on closely related topics like operating systems and algorithms and data structures are also required. After a first introduction into these topics in previous semesters, this module comprises a more advanced extension of prior courses on these two topics.

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. The course focuses on the C programming language in examples and for most implementation work.

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): Prof. Denis Zampunieris, Dr. Jean Botev

Prerequisites: Algorithms 1

Language: English

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

Evaluation: Mini-project (25%, part 1), homework assignments (25%, part 2), final exam (50%).

Literature:

- Introduction to Algorithms, Thomas H. Cormen et al., MIT Press, 978-0262033848
- The Art of Computer Programming – Volume 1: Fundamental Algorithms, Donald E. Knuth, Addison Wesley, 0-201-89683-4
- Compilers: Principles, Techniques and Tools, Alfred V. Aho et al., Addison Wesley, 978-0321547989
- Programming Languages - An Interpreter-Based Approach, Samuel N. Kamin, Addison Wesley, 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 session students: practical midterm exam (30%) + final written exam (70%)
- Resitting students: final written exam (100%)

Summer semester: final written 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.

Module 3.3

Development of "good" software is a core activity of every programmer. This development work does not only consist of pure coding, but a development process combining techniques from engineering science should be used to improve the speed and the quality of the development process. This module constitutes the first part of two modules on methodologies and tools which play important roles in the overall software development process. The module introduces techniques for modeling software, software development methodologies and tools important for software engineering in practice.

After validation of this module, students are capable to:

- use the different types of UML diagrams in the correct way for describing all steps in an object-oriented software development process;
- explain the core ideas of common methodologies used in software development;
- explain the principles of the different steps and tasks 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]

Objectives: 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;
- create a domain model and model associations and attributes;
- 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;
- design state machine diagrams.

Description: Introduction to UML

- A notation to support iterative and incremental Software Development.

Use case diagrams

- Actor, Use cases, Assocations, 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.

Lecturer(s): Dr. Laurent Debrauwer

Prerequisites: Students should have basic knowledge of common principles of Object Oriented Programming.

Language: English

Evaluation: Continuous assessment during the practical sessions (30%) and written final examination (70%).

Literature:

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: The course aims to prepare a future software engineer to run a software engineering project in practice with a team following a development methodology. Common methodologies for software engineering and its constituent parts are combined with student presentations on (usually open-source) tools important for software engineering in 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 principles and techniques of software engineering in the architectural design, detail design, and implementation of software applications.
- use basic features of the practical tools explained in the group presentations in a practical software engineering project.

Description: Common software development process methodologies (waterfall model, incremental approaches, and agile approaches) and its constituent parts are covered in the theoretical part of the course. The principles behind requirement analysis, design, implementation, testing and maintenance will be explained. In addition, the course is centered around the realization of a group project in which the students will give a detailed presentation on

how to use (usually open-source) tools important for software engineering in practice:

1. Introduction to **Docker**
2. Source code version control with **git**
3. Build automation systems: **maven**, **gradle**
4. Using an IDE like **Eclipse** or **IntelliJ** - refactoring, syntax checking with **checkstyle,** using a **debugger**
5. Source API Documentation tools: **javadoc, doxygen**
6. Static Code analysis: **findbugs**
7. Profiling Java code: **VisualVM,** load testing with **JMeter**
8. Testing frameworks: **junit**, **selenium, mockito**
9. Code review with **Gerrit**
10. Continuous delivery with **Jenkins**
11. Project Management with **OpenProject**

Lecturer(s): Prof. Volker Müller

Prerequisites: Students should have some practical experience on programming in the Java language with an IDE (Eclipse or IntelliJ).

Language: English

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

- The lecturer teaches about the more theoretical aspects of software engineering like software development methodologies.
- Student groups give practical presentations on tools commonly used in practical software engineering processes.

Evaluation: The final grade of the course is composed of four parts:

- Presentation of the group project (25%);

- Deliverables of the group project (video 25%, paper 25%);
- Final written exam (25%).

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

Module 3.4

Data management and networking are core domains of information technology. In this module, an introduction on these two topics is provided as basis for a possible future specialization. The courses provide a description of standard networking protocols used world-wide in the Internet as well as a practical introduction to data modeling techniques, data storage and data retrieval using relational database systems and the SQL language.

After validation of this module, students are capable to:

- explain the principles of commonly used technologies and protocols in the Internet;
- apply standard data acquisition and data processing techniques in practice;
- describe and use normalization properties used in databases;
- practically use the SQL language to manipulate relational databases;
- use the acquired knowledge on networks and databases to autonomously read technical documentation or publications on these topics.

Compensation

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

22. Database Management 1 [4 ECTS]

Objectives: The course aims to give an introduction to databases, especially concentrating on data modeling techniques and data retrieval from data sources.

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

- apply semantic data modeling for practical problems;
- create queries in SQL for relational database systems;
- interpret an XML schema for a XML database design correctly.

Description: Early existing problems regarding the processing of data (in the 1960s) have been the motivation to define a novel (hierarchical) concept to manage data. Because of many disadvantages regarding the multi-use of the system, security, or data organization, this original concept has been improved step-by-step and finally found its triumphal procession by the works of E. F. Codd (1970s), who proposed the use of the Relational Data Model and Relational Algebra as fundamental concepts. Since this time, Relational Database Systems have been established and are still the most popular data management systems in almost any application domain. The course covers the following topics:

- Semantic Data Modeling with the Object Definition Language and Entity Relationship Model.
- Relational Algebra and Schema Design (1NF, 2NF, 3NF, BCNF, 4NF)
- SQL: Queries, Constraints, Programming & Triggers
- XML Database Design: DTDs & XML-Schema

Lecturer(s): Prof. Christoph Schommer, Mrs. Nina Hosseini Kivanani

Prerequisites: Requirements for the course are a general mathematical understanding and abstract thinking.

Language: English

Teaching modality: All students are obliged to participate to each session.

Evaluation:

- Applied project (related with COVID-19 data): 50%
- Final exam on theoretical aspects given in the lectures: 50%

Literature:

- R. Elmasri, S. Navathe: Fundamentals of Database Systems. 5th Edition. Pearson Addison Wesley. 2006.
- H. Garcia-Molina, J. D. Ullman, J. Widom: Database Systems – The Complete Book. Prentice Hall International. 2008.
- R. Ramakrishnan, J. Gehrke: Database Management Systems. Mcgraw-Hill Professional. 2002.
- J. Ullman: Principles of Database Systems. W. H. Freeman & Co Ltd. 1982.

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 programming or C 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. They will also have to complete a project over a three weeks practical lab.

Evaluation: Practical exam (20%), practical project (20%), written exam (60%).

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

Details for Semester 4 Courses

Module 4.1

In the fourth semester, student preparation for the professional world starts in a concrete way by improving their knowledge on practical aspects important for enterprises. Objectives of this module are an increase of understanding on important topics concerning work in an enterprise, both from an organisational and legal point of view, and to more sensibilize the students on the problematic and importance of teamwork. The module includes active participation of practitioners coming from the business world in Luxembourg.

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.

Compensation

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

24. Psychologie du travail en groupe [3 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).

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

- de décrir les différents aspects du travail en groupe;
- de réfléchir sur l'expérience pratique par une étude de cas en groupe.

Contenu: **Chapitre I. Psychologie du travail en groupe**

1. **Dynamique des groupes en psychologie sociale**: Caractéristiques d'un groupe; notion de groupe de travail/groupe au travail; identité sociale
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; contre-pouvoir
4. **L'efficacité d'un groupe au travail:** Notion d'efficacité : approche quanti/quali; effet de groupe; importance de la tâche; phénomène de paresse sociale Modèle Input-Process-Output et critiques; approche cognitive (modèles mentaux, mémoire transactive)
5. **La prise de décision en groupe:** Phénomène de polarisation dans un groupe; Risky shift; pensée de groupe (effet Janis : description, analyse, comment l'éviter ?); Partage d'informations; éviter les biais; vers la coopération
6. **La recherche du consensus dans le groupe:** Notion de « bonne » décision (rationalité) et phénomènes d'influence sociale, pressions à l'uniformité; Notion de consensus; influence des minorités; quelle rationalité dans la prise de décision ? Traitement de l'information; validité de l'information; biais de confirmation et conflit socio-cognitif; Conclusion : le rôle du consensus au sein du groupe

Exercices pratiques : apprentissage de l'écoute active/empathie; techniques de négociation des conflits

Chapitre II. Introduction à l'approche sociologique des usages du Web

1. **Aux origines d'internet:** Histoire sociale, imaginaire et valeurs des pionniers du net; Etude de cas : les hackerspaces
2. **Web social et Web participatif:** Réseaux sociaux et blogosphère; e-réputation ; traces numériques, big data et algorithmes; approche éthique; Les GAFA(M) : les cas Google et Amazon; Wikipédia comme objet de connaissance (vigilance participative et évaluation/notation des articles) : apprendre à utiliser intelligemment les sources; Etude de cas de travail en groupe: analyse de conflits éditoriaux dans l'écriture collaborative au sein de Wikipédia.

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: Examen écrit pour 50% de la note finale (questions de cours et études de cas) et un dossier à réaliser en petit groupe (étude de cas avec cahier des charges) pour 50%.

Ouvrage de référence:

- Anzieu D., Martin J.-Y., 2007, *La dynamique des groupes restreints*, Paris, PUF Quadrige.
- 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., 2013, Dans l'esprit du PageRank. Une enquête sur l'algorithme de Google, Réseaux « Politique des algorithmes », 1, no 177, pp. 63-95.

- Cardon D., Levrel J., 2009, « La vigilance participative. Une interprétation de la gouvernance de Wikipédia », Réseaux, 2, no 154, pp. 51-89.
- Cardon D., Levrel J., 2009, *« La vigilance participative. Une interprétation de la gouvernance de Wikipédia »*, Réseaux, 2, no 154, pp. 51-89.
- Cardon D., 2010, *La démocratie internet. Promesses et limites*, Paris, Seuil.
- 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, 177, 63-95.
- Flichy P., 2001, *L'imaginaire d'internet,* Paris, La Découverte.
- Hellrieger D., Slocum J.W., 2006, *Management des organisations*, Bruxelles, De Boeck Université, 2e éd. (chapitre 16 : la gestion des conflits).
- 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.
- Mucchielli R., 2008, *La dynamique des groupes*, Paris, ESF.
- Origgi G., 2013, *Communications* (Dossier *La réputation*), nº 93, Seuil.
- Sekiou L., Blondin L., Fabi B. et al., 2001, *Gestion des ressources humaines*, Bruxelles, De Boeck Université, 2e éd. (chapitre 25 : La gestion des dysfonctionnements).

25. Droit pour informaticiens [3 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): Mme Claire Leonelli, M. Mickael Tome

Prérequis: —

Langue: Français, Anglais

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

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

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

Module 4.2

Software developers require practical experience with the development process. As a consequence, the objectives of this module are the extension of practical knowledge with practically running a software development project for 12 weeks. This practical course puts into practice the both technical and soft skills lessons from the previous semester.

After validation of this module, students are capable to:

- describe the complete cycle of software engineering of an application;
- apply and combine the theories learned in the courses "Modeling with UML" and "Software Engineering" ;

- reflect on their experiences made in a 12 week practical software development project;
- compare recently developed techniques used in local or wireless networks, and specific network services;
- extend their knowledge on networks and 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. 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: Practicals and homeworks (100%).

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

27. Software Engineering Project [5 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).
- Software development in a team of 4-5 students
- 2-3 sprints; after each sprint, a presentation on the current status of the project with discussion must be given before the other students.

Lecturer(s): Prof. Volker Müller

Prerequisites: Software Engineering

Language: English

Teaching modality: Participation to tutorials and practicals is mandatory. All deliverables must be provided. Any non-participation to tutorials and practicals might induce a failure to the course.

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 from the Internet will be announced in class.

Module 4.3

This module contains the a continuation of two topics covered in previous semesters. It deepens the knowledge on databases, including a practical project, and further extends the knowledge on dedicated advanced algorithms, especially relevant for practical applications.

After validation of this module, students are capable to:

- analyse and develop software application linked with a relational database;
- apply the techniques and fundamental concepts of data structures for large volumes of data and advanced algorithms (heuristics, optimisations);
- compare these techniques and concepts to determine a solution best matching requirements in a given development task;
- extend their knowledge on algorithms and databases by autonomously consulting technical journals and articles.

Compensation

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

28. 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 and C++.
- describe how generics work in the Java programming language;
- explain exact and heuristic algorithms for optimization problems.

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.

Lecturer(s): Prof. Steffen Rothkugel, Mr. Aryobarzan Atashpendar

Prerequisites: Algorithms 1, Algorithms 2.

Language: English

Evaluation: **Summer semester:**

- First session students: two practical projects (30%+20%) + final written exam (50%)
- Resitting 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

29. Database Management 2 [4 ECTS]

Objectives: This course is the second part of the Database Trilogy. While the first part discussed data modeling, the relational concept (including SQL and normal forms) and XML, the second part deals with selected advanced aspects of modern database technology, for example database authorization concepts, privacy issues, concepts for handling parallel user requests,

overview of Business Intelligence, and others. The course is mainly a lecture, but we will have exercises in order to reflect the content.

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

- apply the techniques related with DB security in a practical situation;
- explain the basics of DB concurrency control and transactions;
- explain the core ideas of data warehouses and data mining;
- describe the differences between relational DB systems and NoSQL databases.

Description:

- Database Security
- Transactions
- Concurrency Control
- Data Warehousing, ETL, OLAP, OLTP, and Data Mining
- MapReduce
- NoSQL Databases

Lecturer(s): Prof. Christoph Schommer

Prerequisites: Database Management 1

Language: English

Teaching modality: Lectures, Exercise sessions.

Evaluation: 2 intermediate tests (40%); final exam (60%).

Literature:

- R. Elmasri, S. Navathe: Fundamentals of Database Systems. 6th Edition. Pearson Addison Wesley. 2011. (Main reference)
- Silberschatz, Henry F. Korth, S. Sudarshan. Database System Concepts. Sixth Edition.

Module 4.4

Most software development methodologies comprise a large set of different sub-tasks. This module covers two specific topics important in software development: it introduces in a systematic way techniques for software testing and the development of user-friendly and ergonomic user interfaces for different type of applications.

After validation of this module, students are capable to:

- 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;
- use the methods and best practices for these topics during the software engineering process;
- extend their knowledge on software engineering and machine-human interface design by consultation of journals and scientific articles.

Compensation

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

30. 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: Two exams during the semester (30% each), a group final project (30%), and participation in the classroom (10%).

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

31. 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. Mathieu Jimenez, Mr. Dietmar Gehring

Prerequisites: —

Language: English, French

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

Evaluation: 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.

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.

Details for Semester 5 Courses

Module 5.1

Sufficient preparation of students for their first steps towards a professional career is very important. This preparation does not only include IT-related aspects, but also detailed information about the necessary administrative steps to start a professional career. This module prepares students for these tasks, especially also for the mandatory internship in a partner institution done in the last semester of the BINFO. Many of these internships are done in IT teams that develop web-based or advanced enterprise applications. The technical courses in the module concentrate on very relevant topics for such internships, namely advanced development patterns for Java and different languages and frameworks focusing on "Web 2.0" applications. It also includes many practical trainings based on latest techniques popular in the "Web 2.0".

After validation of this module, students are capable to:

- apply well known patterns in the design and development of object oriented applications;
- independently develop practical web applications using the latest technologies and popular frameworks;
- writing CV and other documents required for job applications;
- understand legal and other important aspects of work contracts.

Compensation

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

32. 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: Continuous assessment during the practical sessions (30%) and written final examination (70%).

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].

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
2. Caractéristiques du marché du travail luxembourgeois
3. Aperçu des principaux secteurs d'activités
4. Fonctionnement d'une entreprise
5. Missions des principaux départements
6. Impact de la transformation digitale des entreprises
7. Techniques de recherche d'emploi
8. Description du processus de recrutement d'une entreprise
9. Identification des attentes du recruteur

10. Rédaction de la lettre de motivation et du CV
11. Préparation de l'entretien d'embauche
12. Le droit du travail
13. Composantes du contrat de travail
14. Différents types de contrat de travail
15. Période d'essai et fin du contrat
16. L'entrepreneuriat
17. Aperçu des différents types d'entreprises
18. 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: 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)

34. Web Development 2: Back-End [4 ECTS]

Objectives: The course covers advanced topics in back-end web development, ranging from databases, version control, API development, testing, performance assessment, deployment, monitoring, and documentation. After the course, students will be able to build back-end services for any kind of websites and web applications.

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

- Design and develop back-end systems for websites and web applications,
- Recognize the importance of software engineering for web development,
- Understand classic and modern tools for prototyping,
- Judge and support frameworks, coding standards, and design patterns,
- Choose and employ different development methodologies.

Description: The course covers advanced topics in back-end web development, ranging from databases, version control, API development, testing, performance assessment, deployment, monitoring, and documentation. After the course, students will be able to build back-end services for any kind of websites and web applications.

1. JavaScript on the server side
   - nodejs
     - event loop, async programming, promises
2. Databases
   - File-based
     - SQLite, ndjson, CSV
   - SQL
     - mysql
   - NoSQL
     - mongodb

3. REST API development
    - Design
        - verbs, codes, routing, wrappers, formats
    - Implementation
        - flask, requests, Express, postman
    - Documentation
        - apidoc, swagger

4. Coding standards
    - Linting
        - eslint, pep8
    - Transpilers
        - coffescript, typescript, babel
        - jsdoc

5. Performance
    - Debugging
        - inspectors, debuggers
    - Profiling & Benchmarking
        - inspectors, ApacheBench
    - Logging
        - middlewares, tracer, winston

6. Tooling
    - Command line interface
        - CLI fundamentals
    - Dependency management
        - npm
    - Building systems
        - Makefile, npm scripts
    - Version control
        - git, subversion

7. Testing
    - Unit & Integration testing
        - Jasmine, mocha
    - End-to-end testing
        - Puppeteer
8. Deployment and monitoring
    - PM2
        - startup, watchers
    - Containers
        - docker
9. Continuous integration
    - Hooks
        - workflows, actions

Lecturer(s): Prof. Luis Leiva, Dr. Bereket Abera Yilma

Prerequisites: —

Language: English

Evaluation:

- Coding exercises: 50%
- Final exam (multiple-choice quiz): 50%

Redoing session:

The final exam can be re-taken in the next exam session. Grades from the coding exercises will be retained until the student passes the course.

Literature:

1. M. Haverbeke. 2018. Eloquent JavaScript, 3rd ed.
2. B. Griggs. 2020. Node Cookbook, 4th ed.

3. R. Martin. 2011. The Clean Coder, 1st ed.

Module 5.2

This module provides a portfolio of optional courses where each student can choose five courses based on his/her preferences for specialization. The courses cover different special topics, mostly from the software development domain, that currently attract a lot of attention in the professional world.

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).

35. Banking Information Technologies [4 ECTS]

Objectives: To provide the student with basic knowledge of the banking environment and organisation and give them insight into the possibilities of banking IT and banking networks, both in a theoretical and practical approach.

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

- explain the different components commonly used in banking environments;
- compare the theoretical descriptions about banking environments with practical experiences made during a site visit;

- reflect about the learned lessons in relation with a potential future professional career.

Description:

- General introduction to the financial domain.
- IT architecture for financial services.
- Banking information systems.
- Payment systems.
- Management of banking IT.
- Acquisition of banking IT systems.
- IT-audit.
- Strategic financial information systems.
- Visit of a computer center of a bank.

Lecturer(s): Mr. Helmut Haag

Prerequisites: —

Language: English

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

Evaluation: Written exam (90%), participation in the classroom (10%).

Literature: Will be given during the lecture.

36. 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 tradeoffs 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 tradeoffs 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 tradeoffs 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):

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

Lecturer(s): Prof. Tegawendé Bissyande, Prof. Yves Le Traon, Mr. Abdoul Kader Kabore

Prerequisites: —

Language: English

Evaluation: Practical exercises, homework, continuous evaluation.

Literature: Relevant literature will be provided during the course.

37. 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:

- Written examination (70%)
- Control of homework/assignments (30%)

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

38. 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.

39. 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. Some of the practical examples discussed during the course will be interactively deployed by using the Amazon Web Services (AWS) platform as our infrastructure.

The course covers 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: The final grade will be determined with 2 graded exercises (25% each) and a final written exam (50%).

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

40. 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: Weighted average of written exam (60%) and practical exercises (40%).

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 .

41. Java for Enterprise Applications [4 ECTS]

Objectives: The students will learn how Jakarta EE (formerly Java EE - Java 2 Enterprise Edition) provides a component-based approach to the design, development, assembly, and deployment of enterprise applications. The

different components of Jakarta EE will be examined in theory and practice, with special focus on the value of Web services for realizing service-oriented architectures. The students will also get practical experience with Wildfly, a free Jakarta EE application server (closely related with the popular, but commercial JBoss application server). In addition, a practical introduction to the competitive framework SpringBoot will be given.

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 SpringBoot;
- extend their knowledge on Jakarta EE and SpringBoot 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 Managed/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 SpringBoot, especially web services, SpringBoot persistence, and MVC web applications with SpringBoot.
- Comparison of advantages and disadvantages if Jakarta EE and SpringBoot.

Lecturer(s): Prof. Volker Müller

Prerequisites: Advanced knowledge of the Java programming language.

Language: English

Teaching modality: Lectures and practical sessions. The practical sessions will be done within a Docker container-based environment using the free Wildfly application server (Jakarta EE part) and the Tomcat Server typically used in SpringBoot applications. The last practical session ends with a graded student project using Jakarta EE technology.

Evaluation:

- Four weekly or bi-weekly homeworks (40%)
- Practical Jakarta EE programming project (30%)
- Final written exam (30%)

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.

42. 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): Prof. Pascal Bouvry, 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: Lectures from real systems described in academic papers and labs where students develop/debug on top of a distributed system.

Evaluation:

- 50% project
- 20% quizzes
- 20% paper presentation (5 slides, 1 minute per slide)
- 10% class participation

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

The Bachelor project realizes an at least 12 weeks long, full time, individual project in the IT domain either done in a professional partner institution (for more professionally interested students) or within a research group of the CS department (for more research-oriented students). Students are integrated in a team in the partner institution / research group and shall apply the IT knowledge achieved during their study in a practically relevant (sub)-project in this institution. A Bachelor thesis with a project description, achieved results and practical work experiences made must be written at the end of the project and defended in front of a jury. The Bachelor project is a first active involvement of students in a professional environment to help them to decide about their future career (continue with a Master programme or start a professional career).

After validation of this module, students are capable to:

- reflect on the practical experiences made in a "real life project" made in a professional environment;
- decide on their future plans for either a continued study in a Master programme or a career in the professional world.

Compensation

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

43. Bachelor Project [27 ECTS]

Objectives: The main objective of the 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 in an IT-project realized in a 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 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 Bachelor project students shall apply the technical and inter-personal 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 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.

The Bachelor project must be done in one of two possible variants:

- A **profession-oriented Bachelor project** is done within a professional institution in Luxembourg. The student shall apply and extend his IT technical expertise and soft skills within in a professional environment. This variant of a Bachelor project is strongly recommended for students that plan to start a professional career directly after graduation.
- A **research-oriented Bachelor project** is done within a research group of the University of Luxembourg. The student shall apply and extend his IT scientific expertise for a research-focused question. This variant of a Bachelor project is only recommended for students that plan to continue their academic training within a Master program.

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. 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.

44. Bachelor Project Defense [3 ECTS]

Objectives: The main objective of the Bachelor project defense is the presentation of the done project work and achieved results in front of a jury, to get a fair evaluation of the work.

Learning Outcomes: On successful completion of the Bachelor project defense, students have gained some experience to:

- present the results of a 12-week-long project in a structured and clear way in front of a jury;
- design a pedagogically well-structured and clear presentation of a project;
- answer to questions of a jury and explain the technical and non-technical background of the project (decisions on design, implementation, algorithms, ...);
- show maturity in defending their own ideas and practical work in front of a jury.

Description: The Bachelor project defense evaluates the achieved results of a Bachelor project. The evaluation is done by a jury that consists of both supervisors of the Bachelor project together with at least one more academic lecturer from the university. In the defense, both the scientific and technical quality of the Bachelor project and the professional appearance of the students are assessed. The defense consists of two parts:

- The student presents the work done during the bachelor project in a presentation of about 20 minutes.
- The student answers to questions by the jury members for about 30 minutes. These questions cover both the given presentation and the provided Bachelor project report.

Lecturer(s): Prof. Volker Müller

Prerequisites: —

Language: English

Evaluation: The evaluation of the Bachelor project defense is done by a **Bachelor project jury** consisting of both the local and the academic supervisor of the Bachelor 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.