BiCS
Bachelor in Computer Science
University of Luxembourg, LU

BiCS

Academic Year 2021/2022

Study Programme Annex Reference Document

- Preliminary Version -

bicshub.uni.lu

Friday 19th March, 2021 - 23:40

Contents

1 Introduction

This document provides the necessary organisation information concerning the BiCS - Bachelor in Computer Science.

The targeted audience of this document are:

- UL students: to know the content of the program and the rules that apply,
- UL instructors: to have a global view on the program,
- prospect students who want to know details of the programme as its syllabus and organisation, and
- the general public who may simply be courious about what the BiCS is, and how it functions.

The information contained in this document is aimed at specifying the organisation and rules of the programme. This information complements the information provided in the following official documents:

1. The document [1]: which contains the "Loi ayant pour objet l'organisation de l'Université du Luxembourg"
2. The document [2]: which contains the "Règlement Général des études de l'Université du Luxembourg"
3. The document [3]: which contains the "Règlement d'Ordre Intérieur de l'Université du Luxembourg".

Majority of legal documents are provided in French and not translated. It is up to the reader to ensure he/she is able to understand the content of these official documents. In case new versions of these documents are published, they automatically replace the one listed above.

In case of ambiguity or dispute between the information provided in this document and the information provided in any of the official documents indicated above, the latter prevails.

The BiCS is a bachelor programme that waS started in September 2017. In September 2019 it started offering the 6 study semesters of which it is composed of. In July 2020, the some of the first promotion of students succed to graduated.

During that period some major revisions of this document took place to adapt the programme to both foressen, and mainly, unforeseen situatios. Major changes are duly documented in Section **??**.

It is expected that, with the passing of time, the programme would become more and more stable. Nevertheless, some adaptations may be required to be applied from time to time. Thus, the target audience, and in particular students of the programme are advised to consult this document at the beginning of each semester.

2 Basic information

2.1 Official designation

The programme official designation is **"Bachelor in Computer Science"** (the acronyme used is "BiCS").

2.2 Type of diploma

The programme type of diploma is **"Bachelor degree"**.

2.3 Institutional constellation

The Bachelor in Computer Science is a **single degree** awarded by the University of Luxembourg only.

2.4 Institutional entity

The Bachelor in Computer Science is a programme offered by the **FSTM** faculty of sciences, technology and medicine.

2.5 Official languages

The Bachelor in Computer Science has two official languages:
The primary official language is: **English**.
The secondary official language is chosen between **French** and **German**.

To be accepted to the BiCS an applicant needs to satisfy the detailed language requirements which are indicated in the admission rules. For the BiCS a minimum CEFR equivalent level at application time of:

- B2 for English is "MANDATORY" to be proved
- B1 for French or German is "ADVISED" to be proved (the applicant decides which secondary language he/she chooses and which he/she will keep during all his/her studies).

2.6 Registrations fees

- 400€for semester 1 and 2
- 200€from semester 3 to 6

3 Objectives and learning outcomes

3.1 Objectives

The BiCS Bachelor in Computer science aims at bringing the theoretical and practical skills needed to successfully pursue studies in a Master programme related to Computer Science at the University of Luxembourg or any other world-class university or school. Three main dimensions are targeted by our education programme: **Creativity** as the capacity to generate new ideas (mainly trained using R&D projects embedded in academic or industrial projects), **Science** targeting precise knowledge determined using observation, experimentation, reasoning and formal expression; and current and future **Digital Technologies** which cover the concrete means which rely on electronic devices and used to process information.

3.2 Learning Outcomes

Upon successful completion of the BiCS programme, the student:

- General
  - Has knowledge and understanding of theoretical and methodological foundations of computer science;
  - Has knowledge and understanding of the most important aspects of computer science, including algorithms and data structures, theoretical computer science, programming, information management, modelling and analysis, and is able to apply this knowledge;
  - Has knowledge of a number of methodologies in the area of mathematics and statistics and is able to apply these to problems in computer science;
  - Possesses general academic skills, in particular in relation to computer science;
  - Can formulate a practical problem in computer science as a clear and concise research question and apply his/her knowledge and skills to formulate and analyse possible solutions;
  - Is aware of social, ethical, legal, psychological and other contextual factors related to the computer science discipline;
  - Has academic communication skills that will allow him/her to convey information, concepts and solutions to an audience of specialists, as well as non-specialists, and that will allow the him/her to work in a multidisciplinary team;
  - Possesses the learning skills required to continue with a master programme; and
  - Shows an independent and critical working mode and attitude, while having the ability to acquire further knowledge in the field of computer science.
- Specific
  - Is able to understand a range of algorithms that address an important set of well- defined problems, recognize their strengths and weaknesses, and determine their suitability in particular contexts;
  - Is able to capture, digitize, represent, organize, transform, and present information, also taking into account the problem domain;
  - Understands and describes a computer system's functional components, their characteristics, performance, and interactions;
  - Defines and produces both technical and policy controls and processes intended to protect and defend information and information systems by ensuring their confidentiality, integrity, and availability;
  - Designs interactions between human activities and the computational systems that support them, and constructs interfaces to facilitate those interactions;
  - Understands how networks behave and the key principles behind the organization and operation of the networks;
  - Understands the programming models underlying different languages and makes informed design choices in languages supporting multiple complementary approaches;

- Has the basic ability to select and apply appropriate theories, techniques, tools and practices to a given development effort in order to effectively and efficiently build reliable software systems that satisfy the requirements of customers and users;
- Is able to adapt to interdisciplinary conceptual or application domains; and
- Has the basic knowledge of the relevant social, ethical, legal and professional issues related to computer science.

4 Access and admission

4.1 Specific prerequisites

There is no additional programme-specific eligibility criteria and thus only the general statutory criteria as defined by the law and the common study regulations are used to determine eligibility to the BiCS programme.

4.2 Selection of applicants for admission

The programme restricts admissions to the first year to a maximum of **40 students** (*"numerus clausus"*).

The study programme director is attributed the following responsibilities and competences:

- review and assessment of application files, and
- decision over admission offers.

The selection model is based on the following principles:

1. Each application file is graded using a list of weighted criteria
2. Variable thresholds are defined for each criteria and for the final grade

The programme makes use of a fast track admission allowing to allocate a position to each application file as soon as it is eligible and has a final grade satisfying the constraints of the selection model.

Each application file is graded using the following criteria ranked by decreasing weight:

1. Motivation letter,
2. Level in mathematics since high school,
3. Level in scientific courses since high school,
4. General level since high school,
5. High school degree type (classical, technical, ...),

6. High school degree domain (sciences, economy, ...),
7. Level in English (reference scale CEFR levels A1 to C2),
8. Level in French or German (reference scale CEFR levels A1 to C2), and
9. Quality of the curriculum vitae.

The assessment of the application is made using exclusively the application file documents. No in-person, nor remote (phone, video-conference) interview takes place during the assessment process.

5 Organisation

5.1 Generalities

The BiCS programme has the following general organisation rules:

- it is a **full time** programme which features a regular course load equivalent to full time studies (30 ECTS per semester),
- the programme is organised in **6 semesters**, referred to as $S1, S2, S3, S4, S5$, and $S6$,
- the programme is defined such that a student can be **graduated in 6 semesters**,
- the maximum allowed time to fulfil the conditions that allow the student to be granted the degree is of **10 semesters** ([1], article 36, paragraphe 7). This maximal duration is reduced if the student has beneficed of granted ECTS or course waivers as result of a validation of knowlege acquisition([2], article 11.
- courses of the programme (and any of their associated activities - e.g. seminars, practical sessions, tutorials, etc) are organised from **Monday to Friday, and from 8am to 7pm**,
- by default, any course (or associated activity) requires **mandatory presence** of the student,
- there is **no specialisation track**, and
- all **modules** contain only one course for all semesters, except in semester 6.
- **the degree is granted** to a student if and only if:
  - he/she succeeds to each of the **mandatory courses**, AND
  - he/she gets a total of at least **180 ECTS** when adding all the ECTS granted for the mandatory courses and all optional courses the students obtained, AND
  - he/she satisfied all the other rules included in the legal official documents listed in Section 1.

5.2 Course Types

The BiCS contains different types of courses which are the following:

- **Mandatory (M):** this course must be validated by any student registered to the semester whose programme includes this course,
- **Optional (O):** the student is free to choose or not this course. Nevertheless, specific constraints can be added to the programme, which may restrict the freedom of choosing or not such a course by the student.

The opening of an optional course depends on various factors including the quantity of students that might register to the course. Thus, the official list of available optional courses is available only at semester start.

A student registered to the BiCS can register to an optional course if and only if:

- the course is included in the program definition of the study program annex reference document (i.e. this document) which corresponds to either a prior or current semester in which the student is registered, AND
- the course is open for registration at the beginning of the semester [1].

As illustrations, with the BiCS program (available from the academic year 2021/2022 on) is it possible for a student to collect the minimal number of required ECTS to get the degree with the following ECTS distribution:

Sem.	Student 1	Student 2	Student 3	Student 4	Student 5	Student 6
1	30	30	30	30	30	30
2	30	30	30	30	30	30
3	30	30	30	30	30	30
4	30	26	26	26	30	30
5	30	34	26	30	22	34
6	30	30	38	34	38	38
Total	180	180	180	180	180	>180

Table 1: Possibles ways to get at least 180 ECTS

5.2.1 Course Sub-Types

The following subtypes of courses are defined:

- **Computer Science (CS):** those courses teach knowledge which are directly related to the computer science field according to the CS2013 standard [4].
- **Outside Field (OF):** those courses are not directly related to "Computer Science" and their name contains "Outside Field Course" (or OFC).

By default, if no subtype is indicated, then a course is a CS subtype.

[1] Inclusion of a course in the program definition of the study program annex is by no means a guarantee that the course will be open for registration to the students

5.3 Course formats

The main characteristics of each course offered by the BiCS are given in the programme syllabus, which is presented in Section 12.
Nevertheless, it is worth informing the reader the different overall formats of the courses included into the programme.

5.3.1 Regular courses

A regular course refers to a course where students (often of a same promotion) share a common space with one (or more) instructors. This common space is usually known as the classroom. Depending on the activity to be performed, the settings of the classroom may be different: it could be a room where students are passively listening to the instructor (e.g. lecture or semminar), or a room equiped with special hardware (e.g. lab session, hands on).
Regular courses may be shared with students registered in different programmes. The instructor responsible of the course ensures that the different student's profiles and backgrounds do not impose any obstacles in allowing them to reach the course's objectives.

5.3.2 Bachelor Semester Project courses

A Bachelor Semester Project (referred to as BSP) is a course that follows the project-base learning methodology. The BiCS offers 6 BSP courses, one on each semester.
A BSP course is aimed at trainning a student to develop his/her scientific and technical capacities in a research-oriented settings: one student works in tandem with one tutor to solve a particular problem using scientific methods. These settings apply to all BSP courses, with the exception of the BSP offered in the first semester. This particular BSP is aimed at introducing freshman about the fundamentals of how to perform the following BSP courses along the programme.
Beside the information provided in the course card associated to each BSP course, there exists an entire document [5] that describes the overall functioning of the BSP courses, and each of them in particular.

5.3.3 Online courses

The BiCS has a portfolio of online courses. These are optional (O) courses and are referred to as "OL course" (or simply "OLC" for short). The details about each OLC are provided in the respective course card available in Section 12.

6 Languages

The BiCS implements applied multilingualism. Each student will have his/her studies in which he/she will practice English as primary language and a secondary language freely chosen between French and German.
The student practices both the primary and secondary language when following the **Bachelor Semester Projects** that take place on each semester of the programme.
Since there is a global rule concerning languages at university of Luxembourg which states that the quantity of ECTS obtained using courses labelled according to his/her secondary language must be 45, then each student has to select courses in a way around 25% of the collected ECTS come from courses labelled according to his/her secondary language.
The possibilities for a student to satisfy this rule are the following:

- Do a mobility semester in a university providing a programme in the student's secondary language,
- Choose BSPs (Bachelor Semester Projects) labelled with the student's secondary language
- Choose outside field courses labelled with the student's secondary language

By default, each student receives the ECTS allocated to the validated BSPs for his/her primary and secondary languages.

7 Mobility

The BiCS includes a mobility semester in accordance with the University of Luxembourg regulation documents([2], articles 25-29).
All the rules concerning the mobility semester are communicated to the students by the Mobility service of the University of Luxembourg ([2], articles 25, point 1).

7.1 Mobility rules for Outgoing Students

The following rules are specific to the BiCS:

- The mobility semester is by default the 4^{th} **semester** of the program,
- The BiCS's board of examiners ('jury d'examen') can decide to exempt the mobility semester of a particular student depending on his/her specific situation ([2], articles 27),
- **If a student fails to go in mobility during the 4th semester** it implies that the student cannot get his/her bachelor degree in three years. The student needs to validate his/her mobility semester to be able to get the bachelor degree.

Thus, the 4^{th} semester programme presented in this document may only concern the following categories of students:

- incoming students coming from another university in the context of an exchange program and that selected courses from semester 4,
- BiCS students that were released of mobility semester for eligible reasons (cf. internal regulation documents),
- BiCS Students that failed to get 30 ECTS during their mobility semester,
- guest students.

As a consequence, students going abroad in mobility do not have to select an OFC course in the 4^{th} semester.

7.2 Mobility rules for Incoming Students

Incoming students doing their mobility semester at the University of Luxembourg can select courses from the BiCS.
The following rules are specific to the BiCS to determine which courses can be included in the learning agreement of an incoming student:

- **Optional courses** Optional courses may not always be open each year and each semester, thus you need to contact the faculty single point of contact for mobility to be informed about which optional courses are open (**Mobility Contact**).
- **Bachelor Semester Project (BSP) courses**
  - You must be present [2] at the first session of the course to be allowed to be evaluated for the course,
  - You cannot be registered to another course which is planned on Mondays. Monday is a full day dedicated to the BSP during which you are required to be available from 8am to 7pm at the disposal of your project tutor,
  - Once the semester is started, you are not allowed to remove a BSP from your learning agreement,
  - a BSP requires a huge amount of work (10 ECTS = 300 hours of work), thus you have to be prepared to afford such load of work.

8 Evaluations

8.1 Generalities

- A course is evaluated using one of the following formats:
  - **final exam (or exam, for short:** the evaluation activities take place during a fix date during the exam period,
  - **continuous evaluation:** the evaluation activities take place during the delivery of the course. At least two evaluation activities need to be organised under this format.
  - **combined:** it relies on a both above mentioned methods.
- the couse responsible is in charge of the execution of the evaluation activities, as well as their supervision and assessment,
- A student enrolled in a course is automatically registered in the evaluation of such a course. For courses on which the evaluation relies on an exam, the student has a period to un-register of the examination of the course,
- A student must be registered to the examination of each course he/she wants to be evaluated for,
- A student who is not registered to a course examination does not have the rights to sit to the exam of such a course,

[2] Depending on the modality of the course this may be physical if situation returns to normal as before the pandemic, or virtual if the course is delivered online.

- The board of examiners [3] can only validate grades for those students who were duly registered to the examination of the course.
- A student has up to 4 attempts to pass a mandatory course, regardless its evaluation format ([1], 36, paragraphs 1, 4-attempts rule). After that, the student is exclude from the programme, preventing him/her from obtaining the diploma.

8.2 Course Examination Registration Pre-requisites (CERP)

- A student, to be allowed to register to the examination of a course, must satisfy the pre-requisites stated in such a course.
- Course examination registration pre-requisites (CERP) are available in this document and are updated and published after each official board of examiners (winter and summer).
- It is the responsibility of the student to consult the CERP fo each course he/she is concerned **before the start of each semester** to organise in the best possible way the development of his/her studies along the programme.

CERP are provided in two ways:

- **Global CERP** that can apply to several courses and which are provided either in this this section or in the official documents of the university of Luxembourg cited in section 1.
- **Local CERP** that apply to only one course and that are provided inside each course card given in section 12. If the term **MANDATORY** is used then the condition must be satisfied by the student to allow him/her to be registered to the examination of the concerned course. IF the term **RECOMMENDED** is used then the student is free to request his/her registration to the examination of this course, but there are high risks of failure if the recommended conditions are not satisfied. This is to help the student to plan his/her studies in the best possible way especially if he/she has to retake some course examination.

8.3 Global CERP

8.3.1 Bachelor Semester Projects (BSP)

There are 6 possible bachelor semester projects named **Bachelor Semester Projects 1,2,3,4,5 and 6 or** BSP_1 **(resp.** $BSP\ S_1$**),** BSP_2 **(resp.** $BSP\ S_2$**),... for short**.
The following default rules hold for any student registered to a BSP (see exceptions section in [5]):

- The following board of examiners are available for having the BSP grade validated:
  - Winter Jury (right after the winter semester exam period)
  - Summer Jury (right after the summer semester exam period)
  - Late Summer Jury (just before the next winter semester start)
- by default a board of examiners cannot validate the grades of two different BSPs,
- by default a board of examiners session cannot validate the grade for BSP_i (i $>$ 1) if the ECTS for the BSP_{i-1} were not previously acquired,

[3] Known in French as "jury d'examen".

- the Late Summer board of examiners can validate the grade for a BSP_j (j < 6) if and only if this BSP_j never had a grade validated at it has been formerly accepted, by a previous jury session, to do the BSP_j as a catch-up BSP (a BSP made during July and August and validated at the Late Summer jury session).

8.4 Absence

- By default, the only valuable reason to be absent to an exam is due to health issues. These issues should be such that the student cannot attend the exam. A medical certificate proving such health issues during the time of the exam has to be submitted to the **BiCS** secretary, who should acknowledge (in written) the reception of the certificate.
- Any other kind of request aimed to justify the absence to the exam has to be duly documented and accompanied with valid evidence of the facts. This document (along with any other files) have to be submitted to the **BiCS** secretary, who should acknowledge (in written) the reception of the files. Requests of this type should be submitted at **least 2 weeks before the board of examiners** aimed at validating the grades of the concerned exam would take place.
- The board of examiner determines whether any absence to an exam is considered as justified or not.
- A justified absence to an exam is recorded as "non évalué" on the student's record, and it does count as an attempt.

9 General regulations

It is worth highlighting some important general regulation rules that concern the students enrolled into the programme:

1. Plagiarism is a breach of the law ([1], section VI, article 42),
2. Plagiarism is taking someone else's work or ideas and passing them off as one's own,
3. The possible sanctions in case of plagiarism can be up to the exclusion for a maximum of 5 years of all examinations required for a university degree ([1], section VI, article 43),
4. If plagiarism is detected it is possible that all students having common parts will support the sanction,
5. By default, any artefact created by a student due to a request made by an instructor, it is the result of an individual production,
6. When a student decides to work together with one or more colleages during the preparation of an artefact requested by an instructor, each student must ensure that the artefact to be delivered remains exclusively his/her individual production.

10 Programme changes

This section lists the major changes that were made to the programme since its creation. These changes consisted of courses that were either stopped, or required some major actions like moving it from one semester to another one.
It is worth mentioning that a stopped course is maintained for its examination until there are no more students requiered to pass it.
Next, the list of stopped courses up to now is provided.

10.1 Language Project Course - Primary Language

This course will stop being included in the BiCS study plan starting as of september 2018.
Any student having to retake this course will have his/her primary language deliverables provided for his/her BSP (first session or retake) used to compute the Language Project Course - Primary Language (LPC-PL) retake grade using the following computation:
$grade_{LPC-PL} = 20 * (Grade(PL - Summary) * 0.5 + Grade(PL - Video) * 0.5)/6$

10.2 Language Project Course - Secondary Language

This course will stop being included in the BiCS study plan starting as of september 2018.
Any student having to retake this course will have his/her primary language deliverables provided for his BSP (first session or retake) used to compute the Language Project Course - Secondary Language (LPC-SL) retake grade using the following computation:
$grade_{LPC-SL} = 20 * (Grade(SL - Summary) * 0.5 + Grade(SL - Video) * 0.5)/6$

10.3 Computer Science Project 1

This Course is replaced by the Bachelor Semester Project 1. Any student having to retake this course will have to do a Bachelor Semester Project 1 course.

10.4 Computer Science Project 2

This Course is replaced by the Bachelor Semester Project 2. Any student having to retake this course will have to do a Bachelor Semester Project 2 course.

10.5 Information Management 3

This course will stop being included in the BiCS study plan starting as of september 2021. This course used to be offered in the semester 5.

10.6 Theoretical Computer Science 3

This course will stop being included in the BiCS study plan starting as of september 2021. This course used to be offered in the semester 5.

10.7 Intelligent Systems 2

From september 2021, this course is included in the semester 6 of the programme. Before it used to be in the semester 5.

10.8 Computational Science 2

From september 2021, this course is included in the semester 5 of the programme. Before it used to be in the semester 6.

11 Programme overview

Below are given the summary tables providing essential information about each semester [4].

11.1 Semester 1

S.	MR	CR	Type	L.	course	ECTS	W.
1	1	1	M	EN	Linear Algebra 1	5	150
1	2	1	M	EN	Analysis for Applications 1	5	150
1	3	1	M	EN	Discrete Mathematics 1	5	150
1	4	1	M	EN	Programming Fundamentals 1	5	150
1	5	1	M	EN	Web Development 1	5	150
1	6	1	M	EN/FR/DE	Bachelor Semester Project 1	5	150

11.2 Semester 2

S.	MR	CR	Type	L.	course	ECTS	W.
2	1	1	M	EN	Linear Algebra 2	4	120
2	2	1	M	EN	Theoretical Computer Science 1	4	120
2	3	1	M	EN	Computing Infrastructures 1	4	120
2	4	1	M	EN	Network and Communication	4	120
2	5	1	M	EN	Programming Fundamentals 2	4	120
2	6	1	M	EN/FR/DE	Bachelor Semester Project 2	10	300

11.3 Semester 3

S.	MR	CR	Type	L.	course	ECTS	W.
3	1	1	M	EN	Discrete Mathematics 2	4	120
3	2	1	M	EN	Programming Fundamentals 3	4	120
3	3	1	M	EN	Algorithms and Complexity	4	120
3	4	1	M	EN	Information Management 1	4	120
3	5	1	M	EN	Security 1	4	120
3	6	1	M	EN/FR/DE	Bachelor Semester Project 3	10	300

[4] *See appendix here for legends*

11.4 Semester 4

S.	MR	CR	Type	L.	course	ECTS	W.
4	1	1	M	EN/FR/DE	Bachelor Semester Project 4	10	300
4	2	1	M	EN	Information Management 2	4	120
4	3	1	M	EN	Theoretical Computer Science 2	4	120
4	4	1	M	EN	Programming Languages	4	120
4	5	1	M	EN	Intelligent Systems 1	4	120
4	6	1	O	EN	Online Course (OL)	4	120

11.5 Semester 5

S.	MR	CR	Type	L.	course	ECTS	W.
5	1	1	M	EN/FR/DE	Bachelor Semester Project 5	10	300
5	2	1	M	EN	Software Engineering 1	4	120
5	3	1	M	EN	Human-Computer Interaction (HCI)	4	120
5	4	1	O	EN	Computational Science 1	4	120
5	5	1	O	EN	Computational Science 2	4	120
5	6	1	O	EN	Web Development 2	4	120
5	7	1	O	EN	Natural Language Processing	4	120
5	8	1	O	EN	Online Course (OL)	4	120

11.6 Semester 6

S.	MR	CR	Type	L.	course	ECTS	W.
6	1	1	M	EN/FR/DE	Bachelor Semester Project 6	10	300
6	2	1	M	EN	Software Engineering 2	4	120
6	2	2	O	EN	Security 2	4	120
6	2	3	O	EN	User Centered Design	4	120
6	2	4	O	EN	Data Science for Humanities	4	120
6	2	5	O	EN	Computational Science 3	4	120
6	2	6	O	EN	Intelligent Systems 2	4	120
6	2	7	O	EN	Online Course (OL)	4	120

12 Programme Syllabus

12.1 Semester 1 - Linear Algebra 1

Responsible	**Dr. Perucca**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
1	1	1	M	5	70 h.	150 h.

Prerequisites	**none**
Description	First of two linear algebra courses. Introduces basic techniques for solving linear equations as well as basic notions of linear algebra (linear maps, kernel, image, determinant, etc) and basic related notions of geometry in 2 and 3 dimensions.
Evaluation	Continuous Assignments (homework and written tests) 100%. Redoing session: 100% homework, no kept grade.

Bibliography	The main reference will be book by Anton and Rorres 'Elementary linear algebra - Applications version' 11-th edition (notice that this edition can also be found in electronic form as pdf). Additional references will be given at the beginning of the course. New material and the homework assignments will be uploaded on Moodle.

Content

1. Mathematics
1.1. Matrices
1.1.1. Matrices and linear systems
1.1.2. Matrix multiplication
1.2. Vector spaces
1.2.1. Definitions
1.2.2. Examples of vector spaces
1.2.3. Basis
1.3. Linear maps
1.3.1. Definitions
1.3.2. Kernel, image. Relation between dimensions.
1.3.3. Matrix of a linear map
1.4. Determinants
1.4.1. Determinant of 2x2 and 3x3 matrices
1.4.2. Determinants and injectivity
1.4.3. Determinant and solutions of linear systems
1.5. Geometry in the Euclidean plane and space
1.5.1. Geometry in the plane
1.5.2. Geometry in space

12.2 Semester 1 - Analysis for Applications 1

Responsible	**Dr. McLeay**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
1	2	1	M	5	56 h.	150 h.

Prerequisites	**none**
Description	This course belongs to the set of mathematics courses of the program that covers basic techniques to identify a set of rules for reasoning in the context of the system under study. In this course the focus is made on basic functions and sequences.
Evaluation	First session: Three assignments worth 30%, 30%, and 40% Redoing session: 100% assignment, no kept grade.

-

Bibliography	Lecture notes.

-

Content

1. Analysis for Applications 1
1.1. Functions
1.1.1. Integer, rational, and real numbers. Polynomial and trigonometric functions.
1.2. Sequences of real numbers
1.2.1. Sequences of real numbers. The limit.
1.2.2. Supremum/infimum, monotone convergence. Exponential and hyperbolic trig.
1.3. Continuous real functions
1.3.1. Limit of a function. Sum and product rule.
1.3.2. Limits at infinity. Limits of ratios. Sandwich theorem.
1.3.3. Continuity. Sum and product rule. Intermediate value theorem.
1.4. Derivatives
1.4.1. Definition of derivatives. Tangent lines.
1.4.2. Sum, product, and chain rule. Examples.
1.4.3. Derivatives of inverse functions.
1.4.4. L'Hospital's rule and examples.
1.4.5. Local maxima / minima. Sign of derivative. Stationary points. Mean value theorem.
1.5. Taylor polynomials
1.5.1. Taylor polynomials and higher order approximations.
1.5.2. Big O notation. Computing limits with Taylor polynomials.

12.3 Semester 1 - Discrete Mathematics 1

Responsible	**Dr. Van Der Torre**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
1	3	1	M	5	70 h.	150 h.

Prerequisites	**Mandatory**
Description	Discrete structures are foundational material for computer science. Relatively few computer scientists will be working primarily on discrete structures, but many other areas of computer science require the ability to work with concepts from discrete structures. The discrete structures covered in this introduction include important material from such areas as set theory, logic, graph theory, and number theory.
Evaluation	Oral exmainations over material and exercises (every 2-3 weeks): 100% When redoing the session, the final grade is composed 100% of redone exam

Bibliography	The lecturer will distribute, in class and on Moodle, handouts and exercises covering the topics of each lecture. As extra material, we suggest to use: Rosen K., 'Discrete Mathematics and Its Applications', Mc Graw-Hill. A detailed list of further material will be published on Moodle.

Content

1. Discrete Structures (DS)
1.1. Proof Techniques
1.1.1. Good and bad proofs
1.1.2. Proof by contradiction
1.1.3. Well ordering principle
1.2. Basic Logic
1.2.1. Propositional logic
1.3. Sets, Relations, and Functions
1.3.1. Sets and relations
1.3.2. Size of sets, mapping lemma
1.3.3. Predicates and quantifiers
1.3.4. Set theory, Russell paradox
1.3.5. Induction and strong induction
1.3.6. Partial orders
1.4. Graphs and Trees
1.4.1. Simple graphs, degrees, isomorphism
1.4.2. Graph connectedness, trees
1.4.3. Graph coloring, bipartite matching
1.4.4. Planar graphs
1.4.5. Directed Graphs
1.4.6. Scheduling
1.4.7. State machines, preserved invariants
1.4.8. Derived variables, termination
1.4.9. Stable matching
1.5. Numbers and counting
1.5.1. GCD and integer linear combinations
1.5.2. Modular arithmetic
1.5.3. Division rules

12.4 Semester 1 - Programming Fundamentals 1

Responsible	**Dr. Kelsen**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
1	4	1	M	5	70 h.	150 h.

Prerequisites	
Description	This course introduces the fundamentals of programming using the Python programming language. This is not primarily a Python programming course but rather a discussion of the fundamental concepts underlying computation using Python code examples as illustration. At the same time enough of the Python language is covered for the students to be able to tackle non-trivial problems (eg, in the context of projects). This introductory course forms the basis for more advanced courses on algorithms and object-oriented programming.

Evaluation	Final Exam - written: 50%
	Practicals - continous: 50%
	Redoing evaluation:
	If (last practical grade >= 10)
	then
	- written exam: 50%
	- last practical grade: 50%
	else
	- final exam: 100%

-

Bibliography	

-

Content

12.5 Semester 1 - Web Development 1

Responsible	**Dr. Leiva**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
1	5	1	M	5	70 h.	150 h.

Prerequisites	**None**
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.
Evaluation	Coding exercises: 25% Final exam (multiple-choice quiz): 25% Final project: 50% Redoing students: The final exam can be re-taken in the next exam session. Grades from the coding exercises and final project will be retained for the full academic year. Students who do not pass will be considered as first attendance students in the next academic year.

-

Bibliography	1. R. Anquetil. 2019. Fundamental Concepts for Web Development, 1st ed. 2. C. Lindley. 2019. Front-end Developer Handbook 3. S. Krug. 2000. Don't Make Me Think, 3rd ed.

-

Content

12.6 Semester 1 - Bachelor Semester Project 1

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
1	6	1	M	5	70 h.	150 h.

Prerequisites	**- For all BSP, meetings participation and deliverables submission are mandatory** **- For each non-first BSP, a mandatory participation pre-requisite is to have submitted BEFORE THE FIRST DAY OF THE SEMESTER a valid (description and tutor) validated (i.e. started using the tool) by a tutor using the online tool.** **For complete participation pre-requisites, see full official bachelor semester project reference document available here:** **https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0**
Description	This course is a preparation to the Bachelor Semester Projects (BSP) that a BiCS student will have to do idnividually during semester 2 to semester 6 of his studies. The students are introduced to research and development projects. The topics covered are: - project management - scientific methods for research and development projects - definition of scientific and technical deliverables - conducting a state of the art - scientific and technical report production - design and realization of oral and video presentations - soft skills A micro-BSP is used to learn by practice the necessary skills for a good execution of standard BiCS BSPs. During a normal BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with UL employees in a project context, learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS.
Evaluation	see full official bachelor semester project (BSP) reference document available here: https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0

-

Bibliography	N.A.

-

Content

1. Software Engineering Management
1.1. Initiation and Scope Definition
1.1.1. Determination and Negotiation of Requirements
1.1.2. Feasibility Analysis
1.1.3. Process for the Review and Revision of Requirements
1.2. Review and Evaluation
1.2.1. Determining Satisfaction of Requirements
1.3. Software Project Planning
1.3.1. Determine Deliverables
1.3.2. Process Planning
2. Social Issues and Professional Practice
2.1. Professional Communication
2.1.1. Communicating professionally with stakeholders
2.1.2. Dynamics of oral, written, and electronic team and group communication (cross-reference HCI/Collaboration and Communication/group communication, SE/Project Management/team participation)
2.1.3. Reading, understanding and summarizing technical material, including source code and documentation
2.1.4. Writing effective technical documentation and materials
2.2. Professional Ethics
2.2.1. Community values and the laws by which we live
2.2.2. Keeping up-to-date as a computing professional in terms of familiarity, tools, skills, legal and professional framework as well as the ability to self-assess and progress in the computing field
2.2.3. The nature of professionalism including care, attention and discipline, fiduciary responsibility, and mentoring
2.3. Group Dynamics and Psychology
2.3.1. Dealing with Multicultural Environments
2.3.2. Dealing with Problem Complexity
2.3.3. Dealing with Uncertainty and Ambiguity
2.3.4. Individual Cognition
3. Computing Foundations
3.1. Problem Solving Techniques
3.1.1. Analyze the Problem
3.1.2. Definition of Problem Solving
4. Digital Technologies
4.1. various
4.1.1. Digital technologies will be learned or applied depending on the project subject
5. Computer Sciences
5.1. various
5.1.1. Sciences will be learned or applied depending on the project subject

12.7 Semester 2 - Linear Algebra 2

Responsible	**Dr. Perucca**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
2	1	1	M	4	70 h.	120 h.

Prerequisites	**RECOMMENDED Linear algebra 1**
Description	This course belongs to the set of mathematics courses of the program that covers basic techniques to identify a set of rules for reasoning in the context of the system under study. In this course the focus is advanced notions of linear algebra.
Evaluation	Continuous assignments 100%. Redoing session: homework 100%

-

Bibliography	The main reference will be book by Anton and Rorres 'Elementary linear algebra - Applications version' 11-th edition (notice that this edition can also be found in electronic form as pdf). Additional references will be given at the begining of the course. New material and the homework assignments will be uploaded on Moodle.

-

Content

12.8 Semester 2 - Theoretical Computer Science 1

Responsible	**Dr. Mauw**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
2	2	1	M	4	56 h.	120 h.

Prerequisites	**Successful completion of the course S1/Discrete Mathematics 1 is RECOMMENDED** **Students should have the necessary knowledge in the following fields:** **Basic concepts of set theory** **Basic data structures such as graphs, trees, sequences** **Basic proof techniques: by induction, contradiction** **Basic paradigms in imperative programming languages** **Concepts of algorithms and their applications** **Predicate logic**
Description	This course covers basic principles and techniques in theoretical computer science. More specifically, automata theory, decidability and Turing machines will be the core part of the course. The course is structured into two main parts. In the first part, we introduce the basic concepts of automata theory such as deterministic and non-deterministic automata, regular and context-free languages. We then move to more advanced concepts of automata theory and computation (Turing machines and undecidable problems to name a few). The languages and methods covered in the course are the building blocks of the scientific foundation of computer science. The ultimate goal of the course is to provide a general understanding of the theoretical structures behind the implementation of reasoning techniques and programs in general.
Evaluation	First exam session: Midterm (written): 40% Final exam (written): 40% + optional redo of midterm questions (best grades per subject will be taken) Practical exercises: 20% Resit: Grade for midterm from first exam session: 40% Final exam (written): 40% + optional redo of midterm questions (for every subject, the best grade between the original one and the one for the current redo will be taken) Grade for practical exercises from first exam session: 20%

Bibliography	Introduction to the theory of computation, Michael Sipser, third edition, Cengage Learning, 2013. Plus informal lecture notes which will be made available through moodle.

Content

1. Foundations of Computing (FC)
1.1. Basic Automata Theory and Computability
1.1.1. Automata Theory: introduction, context, motivation, history, notation, basic concepts
1.1.2. Motivation and history, Deterministic finite automata
1.1.3. Non-deterministic finite automata, equivalence of deterministic and non-deterministic automata
1.1.4. Regular expressions, closure properties of regular languages
1.1.5. equivalence of regular languages and finite automata, Non-regular languages
1.1.6. Pumping lemma for regular languages, Context-free languages, context-free grammars, parsing, ambiguity
1.1.7. Pushdown automata

1.2. Advanced Automata Theory and Computability
1.2.1. Inclusion of regular languages in context-free languages, equivalence of pushdown automata and context-free grammars
1.2.2. Chomsky normal forms, pumping lemma for context-free languages
1.2.3. Turing machines
1.2.4. Turing-decidability, Turing machine variants
1.2.5. Church-Turing thesis, decidable problems
1.2.6. Undecidability, halting problem
1.3. Application of Automata Theory
1.3.1. Application: compiler design
1.3.2. Wrap up and wider perspective

12.9 Semester 2 - Computing Infrastructures 1

Responsible	**Dr. Navet**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
2	3	1	M	4	52 h.	120 h.

Prerequisites	**none**
Description	Computing professionals should not regard the computer as a black box that executes programs by magic. This course first aims to develop a deeper understanding of the hardware and software environment upon which all computing is based, and the interface it provides to higher software layers. Students should acquire an understanding and appreciation of a computer system's functional components, their characteristics, performance, and interactions. In particular, it will be taught how data are represented in memory and how they are manipulated during computations. An operating system defines an abstraction of hardware and manages resource sharing among users. This course provides a basic knowledge of operating systems such as the concept of processes and the difference between the kernel and user modes.
Evaluation	- Assignments including programming exercises: 40%. - Final exam: 60%. - Students having failed the course will have to re-sit the final exam at a next exam session, which will then count for 100% of the grade.

-

Bibliography	- 'Structured Computer Organization: International Edition', by Andrew S. Tanenbaum, Todd Austin, Person, 6th edition, 2012. - 'Modern operating systems', by Andrew S Tanenbaum, Herbert Bos, 4th edition, 2015. - 'Operating System Concepts', by Abraham Silberschatz et al, 8th edition, 2008. - 'Computer Systems: a Programmer's Perspective', by Randal E. Bryan, David R. O'Hallaron, 2nd edition, 2011.

-

Content

1. Systems Fundamentals
1.1. Computational paradigms
1.1.1. Basic building blocks of a computer
1.1.2. Sequential vs parallel processing, threads, processes, multicore architectures
1.2. Cross-layer communications
1.2.1. Programming abstractions, interfaces, use of libraries
1.2.2. Layered architecture: Hardware, VM, OS and applications
1.3. Ressource allocation and scheduling
1.3.1. Kinds of ressources: processor, memory, disk, etc
1.3.2. Kinds of scheduling: FIFO, priority
2. Architecture and Organization
2.1. Machine Level Representation of Data
2.1.1. Bits, bytes, and words
2.1.2. Numeric data representation and number bases
2.1.3. Fixed- and floating-point systems
2.1.4. Signed and two-complement representations
2.2. Assembly Level Machine Organization
2.2.1. Control unit, instruction fetch, decode, and execution
2.2.2. Instruction formats
2.3. Memory System Organization and Architecture
2.3.1. Main memory organization and operations
2.3.2. Cache memories and storage systems
2.3.3. Virtual memory
2.3.4. Fault handling and reliability
2.4. Interfacing and Communication
2.4.1. direct-memory access (DMA)
2.5. Multiprocessing and Alternative Architectures
2.5.1. Example SIMD and MIMD instruction sets and architectures
3. Operating Systems
3.1. Overview of Operating Systems
3.1.1. Role and purpose of the operating system
3.2. Operating System Principles
3.2.1. Abstractions, processes, and resources
3.2.2. Concepts of application program interfaces (APIs)
3.2.3. The evolution of hardware/software techniques and application needs
3.2.4. Device organization
3.2.5. Interrupts: methods and implementations
3.2.6. Concept of user/system state and protection, transition to kernel mode
3.2.7. Process scheduling
3.3. Concurrency
3.3.1. Structures: ready list, process control block
3.3.2. The role of interrupts
3.3.3. Dispatching and context switching
3.3.4. Implementing synchronization primitives
3.4. Real-Time and Embedded Systems
3.4.1. Specificities of real-time and embedded systems

12.10 Semester 2 - Network and Communication

Responsible	**Dr. Engel**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
2	4	1	M	4	56 h.	120 h.

Prerequisites	**none**
Description	The Internet and computer networks are now ubiquitous and a growing number of computing activities strongly depend on the correct operation of the underlying networks. Many computing applications that are used today would not be possible without networks. The objective of the course is to give an introduction to networking at large, and TCP/IP networks specifically. The course will provide an introduction to current network architectures, application level protocols (e.g., http), transport protocols (TCP/UDP), routing and IP. The course will introduce Medium Access Control techniques as well as quantitative performance measures for networks. The course will also give a first introduction to mathematical concepts of networks. The course will consist of approx. 28 hours of teaching and 28 hours of practical exercises.
Evaluation	- Practicals: 40%. - Final exam / written: 60%. - Students having failed the course will have to re-sit the final exam at a next exam session, which will then count for 100% of the grade.

Bibliography	'Computer Networking, A Top-Down Approach Featuring the Internet', by James F. Kurose and Keith W. Ross, 7th edition, Pearson, 2017.

Content

1. Programming Languages (PL)
1.1. Imperative programming
1.1.1. The C Programming language
2. Computer Networks and the Internet
2.1. Introduction
2.1.1. What is the Internet? History of the Internet
2.1.2. What is a protocol?
2.1.3. Network edge and core
2.1.4. Switching techniques
2.1.5. Performance metrics: delay, loss, throughput
2.1.6. Protocol layers and OSI model
3. Distributed applications
3.1. Application level protocols
3.1.1. Principles of network applications
3.1.2. Naming and adressing schemes: IP, DNS, URL
3.1.3. Transport-level service models: TCP vs UDP
3.1.4. Application-level service models: client-server, peer-to-peer, producer-consumers
3.1.5. HTTP,FTP, SMTP, POP3, IMAP, DNS
3.1.6. Sockets API to program distributed applications
4. Core networking protocols
4.1. Transport layer
4.1.1. Multiplexing and demultiplexing
4.1.2. Reliable data-transfer
4.1.3. flow and congestion control
4.1.4. Congestion control in TCP
4.2. Network layer
4.2.1. Virtual circuit and datagram networks
4.2.2. IP: the Internet Protocol
4.2.3. Routing protocols
4.2.4. Routing in the Internet
4.2.5. Broadcast and multicast routing
4.2.6. IPV6 vs IPV4
4.2.7. ICMP: internet control message protocol
5. Local Area Networks (LAN)
5.1. Link Layer
5.1.1. Link Layer services
5.1.2. Local Area Networks
5.1.3. MAC level protocols
6. Wireless and multimedia
6.1. Multimedia networking
6.1.1. Multimedia audio and video data
6.1.2. Quality of service

12.11 Semester 2 - Programming Fundamentals 2

Responsible	**Dr. Talbot**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
2	5	1	M	4	56 h.	120 h.

Prerequisites	**RECOMMENDED: Programming Fundamentals 1**
Description	Programming is a craft rather than an exact science. Similarly to the craftsman in his workshop, we need a set of programming tools and guiding principles to develop reliable applications. The main purpose of Programming Fundamentals 2 is to learn how to design and implement programs above 500 lines of code. We give emphasis on clean and beautiful code, well-thought architectures, and a disciplined programming workflow. To achieve our goals, we will rely on the object-oriented paradigm and the Java programming language, as well as various programming tools (e.g., an editor, the shell, git and Github, Maven). Further, this class provides a number of innovative features such as: 1. Automated and personalized feedback on your projects (quick grading, code analysis session). 2. Standard and competitive tracks. 3. An A.I. competition.
Evaluation	* 6 labs (70%) * Coding exam during week 8 (30%) * Bonus points can be collected through various events announced during classes. Redoing evaluation: a coding exam plus oral defence (100%).

-

Bibliography	General programming: * Clean Code: A Handbook of Agile Software Craftsmanship, Robert C. Martin * Agile Software Development, Principles, Patterns, and Practices, Robert C. Martin * Design Patterns: Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides * The Mythical Man-Month: Essays on Software Engineering, Frederick Brooks Java: * Effective Java 3rd Edition, Joshua Bloch * Core Java Volume I - Fundamentals, Eleventh Edition, Cay S. Horstmann

-

Content

12.12 Semester 2 - Bachelor Semester Project 2

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
2	6	1	M	10	70 h.	300 h.

Prerequisites	**- For all BSP, meetings participation and deliverables submission are mandatory** **- For each non-first BSP, a mandatory participation pre-requisite is to have submitted BEFORE THE FIRST DAY OF THE SEMESTER a valid (description and tutor) validated (i.e. started using the tool) by a tutor using the online tool.** **For complete participation pre-requisites, see full official bachelor semester project reference document available here:** **https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0**
Description	This course is a preparation to the Bachelor Semester Projects (BSP) that a BiCS student will have to do idnividually during semester 2 to semester 6 of his studies. The students are introduced to research and development projects. The topics covered are: - project management - scientific methods for research and development projects - definition of scientific and technical deliverables - conducting a state of the art - scientific and technical report production - design and realization of oral and video presentations - soft skills A micro-BSP is used to learn by practice the necessary skills for a good execution of standard BiCS BSPs. During a normal BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with UL employees in a project context, learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS.
Evaluation	see full official bachelor semester project (BSP) reference document available here: https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0

-

Bibliography	N.A.

-

Content

1. Software Engineering Management
1.1. Initiation and Scope Definition
1.1.1. Determination and Negotiation of Requirements
1.1.2. Feasibility Analysis
1.1.3. Process for the Review and Revision of Requirements
1.2. Review and Evaluation
1.2.1. Determining Satisfaction of Requirements
1.3. Software Project Planning
1.3.1. Determine Deliverables
1.3.2. Process Planning
2. Social Issues and Professional Practice
2.1. Professional Communication
2.1.1. Communicating professionally with stakeholders
2.1.2. Dynamics of oral, written, and electronic team and group communication (cross-reference HCI/Collaboration and Communication/group communication, SE/Project Management/team participation)
2.1.3. Reading, understanding and summarizing technical material, including source code and documentation
2.1.4. Writing effective technical documentation and materials
2.2. Professional Ethics
2.2.1. Community values and the laws by which we live
2.2.2. Keeping up-to-date as a computing professional in terms of familiarity, tools, skills, legal and professional framework as well as the ability to self-assess and progress in the computing field
2.2.3. The nature of professionalism including care, attention and discipline, fiduciary responsibility, and mentoring
2.3. Group Dynamics and Psychology
2.3.1. Dealing with Multicultural Environments
2.3.2. Dealing with Problem Complexity
2.3.3. Dealing with Uncertainty and Ambiguity
2.3.4. Individual Cognition
3. Computing Foundations
3.1. Problem Solving Techniques
3.1.1. Analyze the Problem
3.1.2. Definition of Problem Solving
4. Digital Technologies
4.1. various
4.1.1. Digital technologies will be learned or applied depending on the project subject
5. Computer Sciences
5.1. various
5.1.1. Sciences will be learned or applied depending on the project subject

12.13 Semester 3 - Discrete Mathematics 2

Responsible	**Dr. Sorger**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	1	1	M	4	56 h.	120 h.

Prerequisites	
Description	Discrete structures are foundational material for computer science. Relatively few computer scientists will be working primarily on discrete structures, but many other areas of computer science require the ability to work with concepts from discrete structures. This part focuses on the analysis of discrete data. Methods used come from combinatorics, probability theory and statistics.
Evaluation	Intermediate exam / written: 25% Final exam / written: 50% Part from Navet: project or written: 25% Repeating students: Exam oral or written: 100%

-

Bibliography	Online notes from MIT by Lehman, Leighton and Meyer, see https://courses.csail.mit.edu/6.042/spring18/mcs.pdf

-

Content

1. Discrete Structures (DS)
1.1. Basics of Counting
1.1.1. Counting arguments: Set cardinality and counting, Sum and product rule, Inclusion-exclusion principle
1.1.2. The pigeonhole principle
1.1.3. Permutations and combinations, Pascal's identity, The binomial theorem
1.1.4. Solving recurrence relations (cross-reference: AL/Basic Analysis): An example of a simple recurrence relation, such as Fibonacci numbers.
1.2. Discrete Probability
1.2.1. Finite probability space, events
1.2.2. Axioms of probability and probability measures
1.2.3. Conditional probability, Bayes' theorem
1.2.4. (Conditional) Independence
1.2.5. Expectation and its Properties, Mean and Variance
1.2.6. Integer Random Variables (Bernoulli, Binomial, etc..)
1.2.7. Continuous Random Variables (Gauss, Poisson, etc..)
1.3. Analysis of Discrete Data
1.3.1. Maximum Likelihood and A Posteriori Estimation
1.3.2. Law of Large Numbers, Central Limit Theorem
1.3.3. Sampling
1.3.4. Baysian Estimation
1.3.5. Basic concepts of estimation (Statistical model, Point estimation, Confidence intervals, Hypothesis testing)
1.3.6. Regression
1.3.7. Introduction to Python/Pandas

12.14 Semester 3 - Programming Fundamentals 3

Responsible	**Dr. TBD**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	2	1	M	4	42 h.	120 h.

Prerequisites	**RECOMMENDED Programming Fundamentals 1** **RECOMMENDED Programming Fundamentals 2** **RECOMMENDED Computing Infrastructures 1** **RECOMMENDED Networking and Communication**
Description	This course covers several advanced notions of software design and development. The fundamentals of concurrency and parallelism are introduced, together with a thorough discussion of synchronisation issues. Basic concepts of distributed software design and implementation are examined and elaborated on. Principles of event-driven programming are demonstrated and studied.
Evaluation	winter first session: 100% continuous control (submission of code and short explainer videos for lab exercises) winter redoing session: 40% programming task + 60% final written or oral exam

-

Bibliography	To be defined

-

Content

1. Parallel and Distributed Computing
1.1. Parallelsim Fundamentals
1.1.1. Multiple simultaneous computations Goals of parallelism (e.g., throughput) versus concurrency (e.g., controlling access to shared resources) Parallelism, communication, and coordination: Programming constructs for coordinating multiple simultaneous computations, Need for synchronization Programming errors not found in sequential programming: Data races (simultaneous read/write or write/write of shared state), Higher-level races (interleavings violating program intention, undesired non-determinism) o Lack of liveness/progress (deadlock, starvation)
1.2. Parallel Decomposition
1.2.1. Need for communication and coordination/synchronization Independence and partitioning
1.3. Communication and Coordination
1.3.1. Shared Memory Consistency, and its role in programming language guarantees for data-race-free programs
Message passing: Point-to-point versus multicast (or event-based) messages, Blocking versus non-blocking styles for sending and receiving messages, Message buffering (cross-reference PF/Fundamental Data Structures/Queues)
1.3.2. Atomicity: Specifying and testing atomicity and safety requirements, Granularity of atomic accesses and updates, and the use of constructs such as critical sections or transactions to describe them, Mutual Exclusion using locks, semaphores, monitors, or related constructs, Potential for liveness failures and deadlock (causes, conditions, prevention), Composition: Composing larger granularity atomic actions using synchronization, Transactions, including optimistic and conservative approaches
1.3.3. Conditional actions: Conditional waiting (e.g., using condition variables)
1.4. Parallel Algorithms, Analysis, and Programming
1.4.1. Producer-consumer and pipelined algorithms
1.4.2. Examples of non-scalable parallel algorithms
1.5. Distributed Systems
1.5.1. Faults (cross-reference OS/Fault Tolerance): Network-based (including partitions) and node-based failures, Impact on system-wide guarantees (e.g., availability)
1.5.2. Distributed message sending: Data conversion and transmission, Sockets, Message sequencing, Buffering, retrying, and dropping messages
1.5.3. Distributed service design: Stateful versus stateless protocols and services, Session (connection-based) designs, Reactive (IO-triggered) and multithreaded designs
1.5.4. Core distributed algorithms: Election, discovery
2. Programming Languages
2.1. Event-Driven and Reactive Programming
2.1.1. Events and event handlers
2.1.2. Canonical uses such as GUIs, mobile devices, robots, servers
2.1.3. Using a reactive framework: Defining event handlers/listeners, Main event loop not under event-handler-writer's control
2.1.4. Externally-generated events and program-generated events
2.1.5. Separation of model, view, and controller

12.15 Semester 3 - Algorithms and Complexity

Responsible	Dr. Kelsen

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	3	1	M	4	42 h.	120 h.

Prerequisites	
Description	This course presents fundamental algorithms and data structures that are required to solve common problems. The notion of computational complexity of algorithms will be introduced, and mathematical techniques will be presented to analyse the complexity of the algorithms presented in the course. Finally an introduction to problem complexity will be given.

Evaluation	Final Exam - written: 50%
	Practicals - continous: 50%
	Redoing evaluation:
	If (last practical grade >= 10)
	then
	- final exam: 50%
	- last practical grade: 50%
	else
	- final exam: 100%

Bibliography	

Content

1. Algorithms and Complexity (AL)
1.1. Basic Analysis
1.1.1. Differences among best, expected, and worst case behaviors of an algorithm
1.1.2. Asymptotic analysis of upper and expected complexity bounds
1.1.3. Big O notation: formal definition
1.1.4. Little o, big omega and big theta notation
1.1.5. Complexity classes, such as constant, logarithmic, linear, quadratic, and exponential
1.1.6. Big O notation: use
1.1.7. Empirical measurements of performance
1.1.8. Time and space trade-offs in algorithms
1.1.9. Recurrence relations
1.1.10. Analysis of iterative and recursive algorithms
1.1.11. Some version of a Master Theorem
1.2. Algorithmic Strategies
1.2.1. Greedy algorithms
1.2.2. Divide-and-conquer (cross-reference SDF/Algorithms and Design/Problem-solving strategies)
1.2.3. Dynamic Programming [Core-Tier2]
1.2.4. Brute-force algorithms
1.2.5. Recursive backtracking
1.2.6. Branch-and-bound
1.2.7. Heuristics
1.2.8. Reduction: transform-and-conquer
2. Software Development Fundamentals (SDF)
2.1. Algorithms and Design
2.1.1. The concept and properties of algorithms: Informal comparison of algorithm efficiency (e.g., operation counts)
2.1.2. The role of algorithms in the problem-solving process
2.1.3. Problem-solving strategies: Iterative and recursive mathematical functions: Iterative and recursive traversal of data structures: Divide-and-conquer strategies
2.1.4. Fundamental design concepts and principles: Abstraction: Program decomposition: Encapsulation and information hiding: Separation of behavior and implementation

12.16 Semester 3 - Information Management 1

Responsible	**Dr. Theobald**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	4	1	M	4	70 h.	120 h.

Prerequisites	**none**
Description	Information Management is primarily concerned with the capture, digitization, representation, organization, transformation, and presentation of information, algorithms for efficient and effective access and updating of stored information, data modeling and abstraction, and physical file storage techniques. The student needs to be able to develop conceptual and physical data models, determine which IM methods and techniques are appropriate for a given problem, and be able to select and implement an appropriate IM solution that addresses relevant design concerns including scalability, accessibility and usability.
Evaluation	Regular examinations: - intermediate exam / written or oral: 50% - final exam / written or oral: 50% - up to 2 bonus grade points from exercise presentations Redoing students: - exam / written or oral: 100% - up to 2 bonus grade points from exercise presentations if previously gained

Bibliography	To be defined

Content

1. Information Management (IM)
1.1. Information Management Concepts
1.1.1. Information systems as socio-technical systems
1.1.2. Basic information storage and retrieval (IS&R) concepts
1.1.3. Information capture and representation
1.1.4. Supporting human needs: searching, retrieving, linking, browsing, navigating
1.1.5. Information management applications
1.1.6. Declarative and navigational queries, use of links
1.1.7. Content analysis and indexing
1.1.8. Quality issues: reliability, scalability, efficiency, and effectiveness
1.2. Relational Databases
1.2.1. Mapping conceptual schema to a relational schema
1.2.2. Keys and foreign-keys, referential integrity
1.2.3. Relational algebra and relational calculus
1.2.4. Relational database design
1.2.5. Functional dependencies
1.2.6. Decomposition of a schema, lossless-join and dependency-preservation properties of a decomposition
1.2.7. Candidate keys, superkeys, and closure of a set of attributes
1.2.8. Normal forms (2NF, 3NF BCNF)
1.2.9. Multi-valued dependencies (4NF)
1.2.10. Join dependencies (PJNF, 5NF)
1.2.11. Representation theory
1.3. Query Languages
1.3.1. Overview of database languages
1.3.2. SQL (data definition, query formulation, update sublanguage, constraints, integrity)
1.3.3. Select-project-join queries
1.3.4. Aggregations and group-by
1.3.5. Over-operator and sliding windows
1.3.6. Subqueries in SQL
1.3.7. Constraints and triggers
1.3.8. Stored procedures and PL/SQL
1.3.9. QBE and 4th-generation environments
1.3.10. Different ways to invoke non-procedural queries in conventional languages
1.3.11. Overview of other major query languages (e.g., XPATH, SPARQL)

12.17 Semester 3 - Security 1

Responsible	**Dr. Cardoso dos Santos**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	5	1	M	4	72 h.	120 h.

Prerequisites	
Description	Information assurance and security as a domain is the set of controls and processes both technical and policy intended to protect and defend information and information systems by ensuring their confidentiality, integrity, and availability, and by providing for authentication and non-repudiation. The concept of assurance also carries an attestation that current and past processes and data are valid. Both assurance and security concepts are needed to ensure a complete perspective. Information assurance and security education, then, includes all efforts to prepare a workforce with the needed knowledge, skills, and abilities to protect our information systems and attest to the assurance of the past and current state of processes and data. The importance of security concepts and topics has emerged as a core requirement in the Computer Science discipline, much like the importance of performance concepts has been for many years.
Evaluation	The course is divided into two main parts, and evaluated via homework as following: - First session: Part 1 Grade - Homework submission Topics covered: Security and Symmetric crypto Part 2 Grade - Homework submission Topics covered: Public key crypto Final grade = (Part 1 grade + Part 2 Grade) / 2 - Redoing session: + During the Winter Semester (preferred option): The student should join the class during the following year, and improve his Homework grades. The student will them be evaluated in the same manner as a first session students. + During the Summer Semester (for special cases): For Part 1: Individual oral examination, on the topics discussed during the course. The duration of this exam will be of around 45 minutes, and held in English. The exam will be based on the course material in Moodle, and each question will have the same weight. This Examination can be done via Teleconferencing. For part 2: Individual oral examination, on the topics discussed during the course. The duration of this exam will be of around 30 minutes, and held in English. The exam will be based on the course material in Moodle, and each question will have the same weight. This Examination can be done via Teleconferencing. Final grade = (Part 1 grade + Part 2 Grade) / 2

-

Bibliography	Recomended Reading Book:Introduction to Computer Security, by Matt Bishop ISBN: 0-321-24744-2 Chapters of interest: 1, 8, 11, 12, 14. Blog: Scheneier on Security https://www.schneier.com/

-

Content

1. Information Assurance and Security (IAS)
1.1. Foundational Concepts in Security
1.1.1. CIA (Confidentiality, Integrity, Availability)
1.1.2. Concepts of risk, threats, vulnerabilities, and attack vectors (cros- reference SE/Software Project Management/Risk)
1.1.3. Concept of trust and trustworthiness
1.1.4. Secure software
1.1.5. Attacks against computer systems
1.1.6. Malwares, Software and Malware vulnerabilities
1.1.7. Authentication and authorization, access control (mandatory vs. discretionary)
1.1.8. Ethics (responsible disclosure). (cross-reference SP/Professional Ethics/Accountability, responsibility and liability)
1.2. Foundations of security
1.2.1. Basic Cryptography Terminology covering notions pertaining to the different (communication) partners, secure/unsecure channel, attackers and their capabilities, encryption, decryption, keys and their characteristics (responsible disclosure). (cross-reference SP/Professional Ethics/Accountability, responsibility and liability)
1.2.2. Security definitions and attacks on cryptographic primitives: Goals: indistinguishability, unforgeability, collision-resistance ,Attacker capabilities: chosen-message attack (for signatures), birthday attacks, side channel attacks, fault injection attacks.
1.3. Secret-key cryptography
1.3.1. Cipher types (e.g., Caesar cipher, affine cipher) together with typical attack methods such as frequency analysis
1.3.2. Cryptographic primitives: pseudo-random generators and stream ciphers ,block ciphers (pseudo-random permutations), e.g., AES ,pseudo-random functions ,hash functions, e.g., SHA2, collision resistance ,message authentication codes ,key derivations functions
1.3.3. Symmetric key cryptography: Perfect secrecy and the one time pad ,Modes of operation for semantic security and authenticated encryption (e.g., encrypt-then-MAC, OCB, GCM) ,Message integrity (e.g., CMAC, HMAC)
1.4. Security in the real world
1.4.1. Work session with symmetric cryptography tools. Veracrypt e GPG
1.5. Public-key cryptography
1.5.1. Mathematical Preliminaries essential for cryptography: Number theory, modular aritmetic.
1.5.2. Introduction to Public key Cryptography, RSA and Diffie-Hellman
1.5.3. Introduction to Public Key Infrastructure, digital signature and encryption and its challenges.
1.5.4. Authenticated key exchange protocols, e.g., TLS e SSH
1.5.5. Motivate concepts using real-world applications, e.g., electronic cash, secure channels between clients and servers, secure electronic mail, entity authentication, device pairing, voting systems.
1.5.6. Invited Talk: Evren Bulut, on the security applied to Luxembourguish Banks

12.18 Semester 3 - Bachelor Semester Project 3

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	6	1	M	10	70 h.	300 h.

Prerequisites	**- For all BSP, meetings participation and deliverables submission are mandatory** **- For each non-first BSP, a mandatory participation pre-requisite is to have submitted BEFORE THE FIRST DAY OF THE SEMESTER a valid (description and tutor) validated (i.e. started using the tool) by a tutor using the online tool.** **For complete participation pre-requisites, see full official bachelor semester project reference document available here:** **https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0**
Description	This course is a preparation to the Bachelor Semester Projects (BSP) that a BiCS student will have to do idnividually during semester 2 to semester 6 of his studies. The students are introduced to research and development projects. The topics covered are: - project management - scientific methods for research and development projects - definition of scientific and technical deliverables - conducting a state of the art - scientific and technical report production - design and realization of oral and video presentations - soft skills A micro-BSP is used to learn by practice the necessary skills for a good execution of standard BiCS BSPs. During a normal BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with UL employees in a project context, learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS.
Evaluation	see full official bachelor semester project (BSP) reference document available here: https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0

-

Bibliography	N.A.

-

Content

1. Software Engineering Management
1.1. Initiation and Scope Definition
1.1.1. Determination and Negotiation of Requirements
1.1.2. Feasibility Analysis
1.1.3. Process for the Review and Revision of Requirements
1.2. Review and Evaluation
1.2.1. Determining Satisfaction of Requirements
1.3. Software Project Planning
1.3.1. Determine Deliverables
1.3.2. Process Planning
2. Social Issues and Professional Practice
2.1. Professional Communication
2.1.1. Communicating professionally with stakeholders
2.1.2. Dynamics of oral, written, and electronic team and group communication (cross-reference HCI/Collaboration and Communication/group communication, SE/Project Management/team participation)
2.1.3. Reading, understanding and summarizing technical material, including source code and documentation
2.1.4. Writing effective technical documentation and materials
2.2. Professional Ethics
2.2.1. Community values and the laws by which we live
2.2.2. Keeping up-to-date as a computing professional in terms of familiarity, tools, skills, legal and professional framework as well as the ability to self-assess and progress in the computing field
2.2.3. The nature of professionalism including care, attention and discipline, fiduciary responsibility, and mentoring
2.3. Group Dynamics and Psychology
2.3.1. Dealing with Multicultural Environments
2.3.2. Dealing with Problem Complexity
2.3.3. Dealing with Uncertainty and Ambiguity
2.3.4. Individual Cognition
3. Computing Foundations
3.1. Problem Solving Techniques
3.1.1. Analyze the Problem
3.1.2. Definition of Problem Solving
4. Digital Technologies
4.1. various
4.1.1. Digital technologies will be learned or applied depending on the project subject
5. Computer Sciences
5.1. various
5.1.1. Sciences will be learned or applied depending on the project subject

12.19 Semester 4 - Bachelor Semester Project 4

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
4	1	1	M	10	70 h.	300 h.

Prerequisites	**- For all BSP, meetings participation and deliverables submission are mandatory** **- For each non-first BSP, a mandatory participation pre-requisite is to have submitted BEFORE THE FIRST DAY OF THE SEMESTER a valid (description and tutor) validated (i.e. started using the tool) by a tutor using the online tool.** **For complete participation pre-requisites, see full official bachelor semester project reference document available here:** **https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0**
Description	This course is a preparation to the Bachelor Semester Projects (BSP) that a BiCS student will have to do idnividually during semester 2 to semester 6 of his studies. The students are introduced to research and development projects. The topics covered are: - project management - scientific methods for research and development projects - definition of scientific and technical deliverables - conducting a state of the art - scientific and technical report production - design and realization of oral and video presentations - soft skills A micro-BSP is used to learn by practice the necessary skills for a good execution of standard BiCS BSPs. During a normal BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with UL employees in a project context, learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS.
Evaluation	see full official bachelor semester project (BSP) reference document available here: https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0

-

Bibliography	N.A.

-

Content

1. Software Engineering Management
1.1. Initiation and Scope Definition
1.1.1. Determination and Negotiation of Requirements
1.1.2. Feasibility Analysis
1.1.3. Process for the Review and Revision of Requirements
1.2. Review and Evaluation
1.2.1. Determining Satisfaction of Requirements
1.3. Software Project Planning
1.3.1. Determine Deliverables
1.3.2. Process Planning
2. Social Issues and Professional Practice
2.1. Professional Communication
2.1.1. Communicating professionally with stakeholders
2.1.2. Dynamics of oral, written, and electronic team and group communication (cross-reference HCI/Collaboration and Communication/group communication, SE/Project Management/team participation)
2.1.3. Reading, understanding and summarizing technical material, including source code and documentation
2.1.4. Writing effective technical documentation and materials
2.2. Professional Ethics
2.2.1. Community values and the laws by which we live
2.2.2. Keeping up-to-date as a computing professional in terms of familiarity, tools, skills, legal and professional framework as well as the ability to self-assess and progress in the computing field
2.2.3. The nature of professionalism including care, attention and discipline, fiduciary responsibility, and mentoring
2.3. Group Dynamics and Psychology
2.3.1. Dealing with Multicultural Environments
2.3.2. Dealing with Problem Complexity
2.3.3. Dealing with Uncertainty and Ambiguity
2.3.4. Individual Cognition
3. Computing Foundations
3.1. Problem Solving Techniques
3.1.1. Analyze the Problem
3.1.2. Definition of Problem Solving
4. Digital Technologies
4.1. various
4.1.1. Digital technologies will be learned or applied depending on the project subject
5. Computer Sciences
5.1. various
5.1.1. Sciences will be learned or applied depending on the project subject

12.20 Semester 4 - Information Management 2

Responsible	**Dr. Theobald**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
4	2	1	M	4	70 h.	120 h.

Prerequisites	**None**
Description	Information Management is primarily concerned with the capture, digitization, representation, organization, transformation, and presentation of information, algorithms for efficient and effective access and updating of stored information, data modeling and abstraction, and physical file storage techniques. The student needs to be able to develop conceptual and physical data models, determine which IM methods and techniques are appropriate for a given problem, and be able to select and implement an appropriate IM solution that addresses relevant design concerns including scalability, accessibility and usability.
Evaluation	First session examinations: - intermediate exam / written or oral: 30% - final exam / written or oral: 60% - online assignments: 10% - up to 2 bonus grade points from exercise presentations Redoing sessions examination: - exam / written or oral: 100% - up to 2 bonus grade points from exercise presentations if previously gained

Bibliography	Database Systems – The Complete Book (2nd Ed) Hector Garcia-Molina, Jeffrey D. Ullman and Jennifer Widom. Pearson Prentice Hall 2009. ISBN: 978-0131873254

Content

1. Information Management (IM)
1.1. Data Modeling
1.1.1. Data modeling principles
1.1.2. Conceptual models (e.g., entity-relationship, UML diagrams)
1.1.3. Spreadsheet models
1.1.4. Relational data models
1.1.5. Object-oriented models (cross-reference PL/Object-Oriented Programming)
1.1.6. Semi-structured data model (DTDs and XML Schema)
1.1.7. XQuery and XSLT
1.2. Database Systems
1.2.1. Approaches to and evolution of database systems
1.2.2. Components of database systems
1.2.3. Design of core DBMS functions (e.g., query mechanisms, transaction management, buffer management, access methods, recovery)
1.2.4. Database architecture and data independence
1.2.5. Use of a declarative query language
1.2.6. Approaches for managing large volumes of data (e.g., noSQL database systems, use of MapReduce).
1.2.7. Systems supporting semi-structured and/or streaming content
1.3. Indexing
1.3.1. Basic index structures
1.3.2. Creating indexes with SQL
1.3.3. Multi-dimensional and text indices
1.3.4. Physical query operators and buffer management
1.3.5. Query- and join-order-optimization
1.3.6. Indexing the web (e.g., web crawling)

12.21 Semester 4 - Theoretical Computer Science 2

Responsible	Dr. Pang

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
4	3	1	M	4	70 h.	120 h.

Prerequisites	**RECOMMENDED Discrete Mathematics 1 / Theoretical Computer Science 1** **Specific recommended knowledge for this course:** **- Automata theory** **- Mathematical logic (propositional and first-order logic)** **- Discrete mathematics (sets, graphs, trees, etc.)** **- Complexity theory** **- Algorithms and data structures**
Description	This course covers basic techniques that have been devised for formal modelling and verification of computer systems. It starts with introducing models for concurrent systems and the explanation of the state-space explosion problem. This is followed by a study of linear-time properties and regular properties. Later, both linear-time and branching time temporal logics (LTL and CTL), and their model checking algorithms will be introduced. The course also covers techniques that deal with the state-space explosion problem, if possible.
Evaluation	First exam session: Practical exercises/homeworks: 100% (the best 80% of all exercises) Resit: Practical exercises/homeworks: 100% (the best 80% of all exercises)

Bibliography	The course is based on 'Principles of Model Checking', by C. Baier and J.-P. Katoen, MIT Press, 2008. Additional literature: 'Model Checking' by E. M. Clarke, O. Grumberg, D. A. Peled, MIT Press, 1999 and 'Logic in Computer Science – Modelling and Reasoning about Systems' by M. Huth and M.D. Ryan, Cambridge University Press, 2004

Content

1. Theoretical Computer Science
1.1. Introduction
1.1.1. Introduction, motivation, overview
1.2. Syntax and Semantics
1.2.1. Concurrency and communication
1.2.2. Labeled transition systems
1.3. Transition systems
1.3.1. States, Sets and predicates in First Order Logic
1.3.2. Linear temporal logic (LTL) and computation tree logic (CTL)
1.3.3. Linear-time properties and regular properties
1.3.4. State generation (over-approximation)
1.4. Model Checkers
1.4.1. The state-space explosion problem
1.4.2. LTL and CTL model checking algorithms
1.4.3. Abstraction, equivalences, partial order reduction
2. Software Engineering (SE)
2.1. Formal Methods
2.1.1. Model checking
2.2. Software Reliability
2.2.1. Software reliability models

12.22 Semester 4 - Programming Languages

Responsible	**Dr. Herter**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
4	4	1	M	4	70 h.	120 h.

Prerequisites	**Programming Fundamentals 1 RECOMMENDED** **Programming Fundamentals 2 RECOMMENDED** **Programming Fundamentals 3 RECOMMENDED**
Description	A program is a written representation of a computation, in a formal language. An interpreter is a program whose input is a program P and some input data x, it executes P on x. It's usually too expensive and not runtime-efficient to use interpreters for high-level languages, so we generally translate them into something more easily interpretable. A translator (compiler) is a program which takes as input a program P in some language L1 and outputs a program P' in another language L2 such that P and P' have the same semantics, ie. compute the same results from the same input data. This course aims to make you understand programming language implementation in an as easy as possible way through concrete examples. It will guide you through all the main phases of the design and the implementation of an interpreter and of a compiler. To be able to design and implement interpreters and compilers will: - make you a better programmer in general, as you will better understand a language's intricacies. - make you a better computer scientist, because programming technologies span so many areas of the discipline, including formal language theory, grammars, computability, semantics, virtual machines and all the advanced concepts in modern programming languages, to cite a few. - allow you to practice software engineering principles and tools seen in previous semesters, for interpreters and compilers are generally large and complex software. Due to the approach chosen in this course - an interleaved mix of lectures and exercice sessions, practical work is an essential part of the course and will be assessed - you will get very quickly into the business of actually implementing a programming language and running programs written in it.
Evaluation	continuous control in mandatory homework assignments / submission: 100% Redoing session: individual remote video oral examination: 100%

-

Bibliography	R. WILHELM, H. SEIDL, S. HACK, Compiler Design - Syntactic and Semantic Analysis R. WILHELM, H. SEIDL: Compiler Design. Virtual Machines H. SEIDL, R. WILHELM, S. HACK: Compiler Design. Analysis and Transformation F. NIELSON, H. NIELSON, C. HANKIN: Principles of Program Analysis P. COUSOT, R. COUSOT: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints

-

Content

1. Programming Languages (PL)
1.1. Program Representation
1.1.1. Programs that take (other) programs as input such as interpreters, compilers, type-checkers, static code analyzers, documentation generators.
1.1.2. Abstract syntax, grammars, semantics tree.
1.1.3. Main data structures for execution or translation of programs.
1.2. Program Execution
1.2.1. Interpretation vs. compilation to native or intermediate code.
1.2.2. Program execution pipeline: lexing, parsing, type-checking, analysis/optimization, computation/translation.
1.2.3. Runtime representation of core language constructs.
1.3. Program Translation
1.3.1. Compilation schemes. Execution as native code or within a virtual machine.
1.3.2. Scope and binding resolution. Sub-program calls and passing of parameter values.
1.4. Program Analysis
1.4.1. Static analysis and formal validation/verification.

12.23 Semester 4 - Intelligent Systems 1

Responsible	Dr. Brust

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
4	5	1	M	4	70 h.	120 h.

Prerequisites	none
Description	Artificial intelligence (AI) is the study of solutions for problems that are difficult or impractical to solve with traditional methods. It is used pervasively in support of everyday applications such as email, word-processing and search, as well as in the design and analysis of autonomous agents that perceive their environment and interact rationally with the environment. The solutions rely on a broad set of general and specialized knowledge representation schemes, problem solving mechanisms and learning techniques. They deal with sensing (e.g., speech recognition, natural language understanding, computer vision), problem-solving (e.g., search, planning), and acting (e.g., robotics) and the architectures needed to support them (e.g., agents, multi-agents). The study of Artificial Intelligence prepares the student to determine when an AI approach is appropriate for a given problem, identify the appropriate representation and reasoning mechanism, and implement and evaluate it.
Evaluation	First session: 40% project work, 40% written exam, 20% participation Redoing rule: 100% written exam

Bibliography	Artificial Intelligence: A Modern Approach (Prentice Hall Series in Artificial Intelligence)by Stuart Russell and Peter Norvig http://aima.cs.berkeley.edu/

Content

1. Intelligent Systems (IS)
1.1. Fundamental Issues
1.1.1. Overview of AI problems, examples of successful recent AI applications
1.1.2. What is intelligent behavior? The Turing test, Rational versus non-rational reasoning
1.1.3. Problem characteristics: Fully versus partially observable, Single versus multi-agent, Deterministic versus stochastic, Static versus dynamic, Discrete versus continuous
1.1.4. Nature of agents: Autonomous versus semi-autonomous, Reflexive, goal-based, and utility-based, the importance of perception and environmental interactions
1.1.5. Philosophical and ethical issues. [elective]
1.2. Basic Search Strategies
1.2.1. Problem spaces (states, goals and operators), problem solving by search
1.2.2. Factored representation (factoring state into variables)
1.2.3. Uninformed search (breadth-first, depth-first, depth-first with iterative deepening)
1.2.4. Heuristics and informed search (hill-climbing, generic best-first, A*)
1.2.5. Space and time efficiency of search
1.2.6. Two-player games (introduction to minimax search)
1.2.7. Constraint satisfaction (backtracking and local search methods
1.3. Basic Machine Learning
1.3.1. Definition and examples of broad variety of machine learning tasks, including classification
1.3.2. Inductive learning
1.3.3. Simple statistical-based learning, such as Naive Bayesian Classifier, decision trees
1.3.4. The over-fitting problem
1.3.5. Measuring classifier accuracy

12.24 Semester 4 - Online Course (OL)

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
4	6	1	O	4	120 h.	120 h.

Prerequisites	**Consult the 'Online Course' section of the BiCS Study Programme Annex Reference Document**
Description	The BiCS program offers in the list of optional courses the possibility to follow selected 'Online Courses' (OL Courses). The rules that must be followed concerning the selection / execution and evaluation of online courses are indicated in the 'Online Course' section of the BiCS Study Programme Annex Reference Document. The list of possible online courses (provided in this course card) can change each semester and it is advised to consult the following the course folder to get details and access to the available courses descriptions and registrations. The course folder path in the BiCS students chared folder (https://dropit.uni.lu/invitations?share=99a33b312f6853c03ed0) is Courses/OnlineCourses
Evaluation	Consult the 'Online Course' section of the BiCS Study Programme Annex Reference Document

Bibliography	N.A.

Content

12.25 Semester 5 - Bachelor Semester Project 5

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	1	1	M	10	70 h.	300 h.

Prerequisites	**- For all BSP, meetings participation and deliverables submission are mandatory** **- For each non-first BSP, a mandatory participation pre-requisite is to have submitted BEFORE THE FIRST DAY OF THE SEMESTER a valid (description and tutor) validated (i.e. started using the tool) by a tutor using the online tool.** **For complete participation pre-requisites, see full official bachelor semester project reference document available here:** **https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0**
Description	This course is a preparation to the Bachelor Semester Projects (BSP) that a BiCS student will have to do idnividually during semester 2 to semester 6 of his studies. The students are introduced to research and development projects. The topics covered are: - project management - scientific methods for research and development projects - definition of scientific and technical deliverables - conducting a state of the art - scientific and technical report production - design and realization of oral and video presentations - soft skills A micro-BSP is used to learn by practice the necessary skills for a good execution of standard BiCS BSPs. During a normal BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with UL employees in a project context, learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS.
Evaluation	see full official bachelor semester project (BSP) reference document available here: https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0

-

Bibliography	N.A.

-

Content

1. Software Engineering Management
1.1. Initiation and Scope Definition
1.1.1. Determination and Negotiation of Requirements
1.1.2. Feasibility Analysis
1.1.3. Process for the Review and Revision of Requirements
1.2. Review and Evaluation
1.2.1. Determining Satisfaction of Requirements
1.3. Software Project Planning
1.3.1. Determine Deliverables
1.3.2. Process Planning
2. Social Issues and Professional Practice
2.1. Professional Communication
2.1.1. Communicating professionally with stakeholders
2.1.2. Dynamics of oral, written, and electronic team and group communication (cross-reference HCI/Collaboration and Communication/group communication, SE/Project Management/team participation)
2.1.3. Reading, understanding and summarizing technical material, including source code and documentation
2.1.4. Writing effective technical documentation and materials
2.2. Professional Ethics
2.2.1. Community values and the laws by which we live
2.2.2. Keeping up-to-date as a computing professional in terms of familiarity, tools, skills, legal and professional framework as well as the ability to self-assess and progress in the computing field
2.2.3. The nature of professionalism including care, attention and discipline, fiduciary responsibility, and mentoring
2.3. Group Dynamics and Psychology
2.3.1. Dealing with Multicultural Environments
2.3.2. Dealing with Problem Complexity
2.3.3. Dealing with Uncertainty and Ambiguity
2.3.4. Individual Cognition
3. Computing Foundations
3.1. Problem Solving Techniques
3.1.1. Analyze the Problem
3.1.2. Definition of Problem Solving
4. Digital Technologies
4.1. various
4.1.1. Digital technologies will be learned or applied depending on the project subject
5. Computer Sciences
5.1. various
5.1.1. Sciences will be learned or applied depending on the project subject

12.26 Semester 5 - Software Engineering 1

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	2	1	M	4	70 h.	120 h.

Prerequisites	**none**
Description	'Software engineering is the discipline concerned with the application of theory, knowledge, and practice to effectively and efficiently build reliable software systems that satisfy the requirements of customers and users. This discipline is applicable to small, medium, and large-scale systems. It encompasses all phases of the lifecycle of a software system, including requirements elicitation, analysis and specification, design, construction, verification and validation, deployment, and operation and maintenance. Whether small or large, following a traditional plan-driven development process, an agile approach, or some other method, software engineering is concerned with the best way to build good software systems. Software engineering uses engineering methods, processes, techniques, and measurements. It benefits from the use of tools for managing software development, analyzing and modeling software artifacts, assessing and controlling quality, and for ensuring a disciplined, controlled approach to software evolution and reuse. The software engineering toolbox has evolved over the years. For instance, the use of contracts, with requires and ensure clauses and class invariants, is one good practice that has become more common. Software development, which can involve an individual developer or a team or teams of developers, requires choosing the most appropriate tools, methods, and approaches for a given development environment.' [CS 2013]
Evaluation	- First evaluation: 100% continuous control AND ONLY DURING WINTER SEMESTER (no first evaluation is possible in summer) - Redoing evaluation: Final written and oral examination * 2/3 + previous continuous control * 1/3 rmq: final written and oral examination means that you'll have one written examination during the examination period + one oral examination the during examination period.

-

Bibliography	I. Sommerville. Software Engineering. Pearson, 2015. ISBN 9781292096131 S. Bennett, Ray Farmer, and S. McRobb. Object-oriented Systems Analysis and Design: Using UML. McGraw-Hill Education, 2010. ISBN 9780077125363 J. Rumbaugh, I. Jacobson, and G. Booch. The Unified Modeling Language Reference manual. Addison-Wesley object technology series. Addison-Wesley, 2010. ISBN 9780321718952. G. Booch, J. Rumbaugh, and I. Jacobson. The Unified Modeling Language User Guide. Object Technology Series. Addison-Wesley, 2005. ISBN 9780321267979.

-

Content

1. Software Development Fundamentals (SDF)
1.1. Development Methods
1.1.1. Program comprehension
1.1.2. Program correctness : Types of errors (syntax, logic, run-time) ,The concept of a specification ,Defensive programming (e.g. secure coding, exception handling) ,Code reviews ,Testing fundamentals and test-case generation ,The role and the use of contracts, including pre- and post-conditions ,Unit testing
1.1.3. Simple refactoring
1.1.4. Modern programming environments: Code search, Programming using library components and their APIs
1.1.5. Debugging strategies
1.1.6. Documentation and program style
2. Software Engineering (SE)
2.1. Software Processes
2.1.1. Systems level considerations, i.e., the interaction of software with its intended environment (cross- reference IAS/Secure Software Engineering)
2.1.2. Introduction to software process models (e.g., waterfall, incremental, agile), Activities within software lifecycles
2.1.3. Programming in the large vs. individual programming
2.1.4. Evaluation of software process models
2.1.5. Software quality concepts
2.1.6. Process improvement
2.1.7. Software process capability maturity models
2.1.8. Software process measurements
2.2. Software Project Management
2.2.1. Team participation: Team processes including responsibilities for tasks, meeting structure, and work schedule ,Roles and responsibilities in a software team ,Team conflict resolution ,Risks associated with virtual teams (communication, perception, structure)
2.2.2. Effort Estimation (at the personal level)
2.2.3. Risk (cross reference IAS/Secure Software Engineering): The role of risk in the lifecycle, Risk categories including security, safety, market, financial, technology, people, quality, structure and process
2.2.4. Team management: Team organization and decision-making ,Role identification and assignment ,Individual and team performance assessment
2.2.5. Project management: Scheduling and tracking o
2.2.6. Software measurement and estimation techniques
2.2.7. Software quality assurance and the role of measurements
2.2.8. Risk: Risk identification and management ,Risk analysis and evaluation ,Risk tolerance (e.g., risk-adverse, risk-neutral, risk-seeking) ,Risk planning
2.2.9. System-wide approach to risk including hazards associated with tools
2.3. Tools and Environments
2.3.1. Software configuration management and version control
2.3.2. Release management
2.3.3. Requirements analysis and design modeling tools
2.3.4. Testing tools including static and dynamic analysis tools
2.3.5. Programming environments that automate parts of program construction processes (e.g., automated builds): Continuous integration ,
2.3.6. Tool integration concepts and mechanisms
2.4. Requirements Engineering
2.4.1. Describing functional requirements using, for example, use cases or users stories
2.4.2. Properties of requirements including consistency, validity, completeness, and feasibility
2.4.3. Software requirements elicitation
2.4.4. Describing system data using, for example, class diagrams or entity-relationship diagrams
2.4.5. Non-functional requirements and their relationship to software quality (cross-reference IAS/Secure Software Engineering)
2.4.6. Evaluation and use of requirements specifications
2.4.7. Requirements analysis modeling techniques
2.4.8. Acceptability of certainty / uncertainty considerations regarding software / system behavior
2.4.9. Prototyping
2.4.10. Basic concepts of formal requirements specification
2.4.11. Requirements specification
2.4.12. Requirements validation
2.4.13. Requirements tracing
2.5. Software Design
2.5.1. System design principles: levels of abstraction (architectural design and detailed design), separation of concerns, information hiding, coupling and cohesion, re-use of standard structures
2.5.2. Design Paradigms such as structured design (top-down functional decomposition), object-oriented analysis and design, event driven design, component-level design, data-structured centered, aspect oriented, function oriented, service oriented
2.5.3. Structural and behavioral models of software designs
2.5.4. Design patterns
2.5.5. Relationships between requirements and designs: transformation of models, design of contracts, invariants
2.5.6. Software architecture concepts and standard architectures (e.g. client-server, n-layer, transform centered, pipes-and-filters)
2.5.7. Refactoring designs using design patterns
2.5.8. The use of components in design: component selection, design, adaptation and assembly of components, components and patterns, components and objects (for example, building a GUI using a standard widget set)

12.27 Semester 5 - Human-Computer Interaction (HCI)

Responsible	**Dr. Bongard-Blanchy**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	3	1	M	4	49 h.	120 h.

Prerequisites	**none**
Description	Human-computer interaction (HCI) investigates interactions of humans and technology (e.g., computers). It emerged in the 1980s (with roots in several earlier disciplines) and focuses on the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them [1]. Accordingly, HCI is multidisciplinary at its core and draws on a huge variety of influences, including psychology, design, computer science, media studies, ergonomics, engineering and anthropology. The first part of this course will teach the fundamentals of HCI: its theories and history, important psychological and physiological aspects (e.g. cognition, emotion, needs, perception), HCI principles and key terms (e.g. user interface, usability, user experience), and selected sub-disciplines. The second part of the course teaches principles and methods of User Experience (UX) design. Every session is an interactive training session, combined with theoretical input from the lecturers and training activities where students discuss and apply the methods within their project groups. After a successful completion of the HCI course, the student 1. knows the theoretical foundations of HCI. 2. understands the basic psychological principles behind the application of HCI and UX methods 3. can apply interface guidelines from a Human-centered design perspective 4. can apply the various user research methods, along the design process including their evaluation strategies [1] Churchill, Bowser, Preece (2013): Teaching and Learning Human-Computer Interaction. https://interactions.acm.org/archive/view/march-april-2013/teachingand-learning-human-computer-interaction
Evaluation	At the beginning of the course students will form small groups. Throughout the course, students will work collaboratively on their projects. Each group is required to show their week-to-week homework during the course to demonstrate how they applied the methods. The final evaluation will be based on each group's project presentation. If the delivered homework assignment does not meet the expected learning objective, the students will be asked to rework the deliverable.

-

Bibliography	The necessary material will be introduced throughout the course.

-

Content

12.28 Semester 5 - Computational Science 1

Responsible	**Dr. Navet**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	4	1	O	4	42 h.	120 h.

Prerequisites	**none**
Description	This introduction to Computational Science will provide you with a general overview on the modelling of complex systems, the use of simulation to learn from the models and optimization techniques to make design choices. The learning relies on assignments that will required you to solve problems inspired from real-world applications. Each time, you will be able to evaluate the efficiency of your proposals by comparison with baseline results obtained with standard algorithms of the field. On successful completion of this course, students will have practical insights into: - The formalisms and techniques to model complex systems with illustrations in the domains of engineering and finance, - The process of building, validating and simulating a model, and how it can be used for decision making, - Modern approaches to problem solving with Deep Neural Networks, Reinforcement learning and Generative Design. The course is organized as a series of 5 lectures: - Computational Science (CS) and the use of Artificial Intelligence in CS - Introduction to Reinforcement Learning (RL) - Introduction to Neural Networks - Deep Reinforcement Learning: Reinforcement Learning driven by Neural Networks - Simulation models in the design of complex systems
Evaluation	- Two individual projects and a larger group project - Students having failed the course will be given an individual project at a next semester, which will then count for 100% of the grade.

-

Bibliography	- R.S. Sutton and A.G. Barto, Reinforcement Learning: An Introduction, 2018. Available on-line. - A. Zai, B. Brown, Deep Reinforcement Learning in Action, Mannings, 2019. - F. Chollet , Deep Learning with Python, Manning publications, 2017. - Tensorflow 2.0: Deep Learning and AI, course on Udemy, 2019.

-

Content

1. Computational Science (CN)
1.1. Processing
1.1.1. Introduction to Neural networks and Deep Neural Networks. Application to a case-study (assignment nb. 1).
1.1.2. Introduction to Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL). Application to a case-study (trying to outperform a standard algorithm from engineering with RL and DRL - assignment nb. 2)
1.1.3. Introduction to optimization algorithms: optimal techniques and heuristics. Multi-objective optimization.
1.1.4. Introduction to Generative Design (exploring design space to find solutions that meet constraints and objectives). Illustration in the CAD and Electrical/Electronic engineering domain. Application to a case-study.
1.2. Introduction to Modeling and Simulation
1.2.1. Models as abstractions, models used in Model-Driven Engineering, multi-physics modelling
1.2.2. Simulation techniques and tools, such as physical simulations, human-in-the-loop guided simulations, and virtual reality
1.2.3. Approaches to validating models (e.g., comparing a simulation's output to real data or the output of another model)
1.3. Modeling and Simulation
1.3.1. Purpose of modeling and simulation including optimization, supporting decision making, forecasting, safety considerations, for training and education
1.3.2. Tradeoffs including performance, accuracy, validity, and complexity
1.3.3. The simulation process, identification of key characteristics or behaviors, simplifying assumptions, validation of outcomes
1.3.4. Model building: use of mathematical formulas or equations, graphs, constraints, methodologies and techniques, use of time stepping for dynamic systems.
1.3.5. Assessing and evaluating models and simulations in a variety of contexts, verification and validation of models and simulations
1.3.6. Important application areas including economics and finance, city and urban planning, science and engineering
1.3.7. Software in support of simulation and modeling, packages, languages
1.4. Interactive Visualization
1.4.1. Principles of data visualization
2. Data Warehousing
2.1. Business Intelligence
2.1.1. An introduction to machine learning with a focus on clustering and classification

12.29 Semester 5 - Computational Science 2

Responsible	**Dr. Beex**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	5	1	O	4	56 h.	120 h.

Prerequisites	**RECOMMENDED, BUT NOT NECESSARILY REQUIRED: Computational Sciences I & II**
Description	Computational Science is a field of applied computer science, that is, the application of computer science to solve problems across a range of disciplines. It combines computer simulation, scientific visualization, mathematical modeling, computer programming and data structures, networking, database design, symbolic computation, and high performance computing with various disciplines. This area offers exposure to many valuable ideas and techniques, including precision of numerical representation, error analysis, numerical techniques, parallel architectures and algorithms, modeling and simulation, information visualization, software engineering, and optimization.
Evaluation	During the course (weeks 5 & 8), the student will receive two short exams, strongly based on the exercises in the lectures notes. The grades of these short exams will each be weighted with 20% for the final grade. At the end of the course (in the exam period), a final exam will take place, which will be weighted for 60%. The final grade is thus composed by 0.2 times the grade of the first short exam, by 0.2 times the grade of the second short exam, and by 0.6 times the grade of the final exam. Redoing students: - will receive two new short exams and a final exam in the next winter semester (short exams in weeks 5 and 8, final exam in the exam period of the winter semester). The weight factors of these exams are the same as described above.

-

Bibliography	Notes and exercises will be provided. These notes and exercises are amongst others based on 'Nocedal, J & Wright, S.J., Numerical optimisation. Springer Series in Operational Research and Financial Engineering, 2nd Edition, Springer Science+Business Media, LLC, New York, USA (2006). ISBN-10: 0-387-30303-0, ISBN-13: 978-0387-30303-1

-

Content

12.30 Semester 5 - Web Development 2

Responsible	**Dr. Leiva**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	6	1	O	4	70 h.	120 h.

Prerequisites	**RECOMMENDED: The "Web development 1" course is highly encouraged, or previous experience in front-end web development.**
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.
Evaluation	Coding exercises: 25% Final exam (multiple-choice quiz): 25% Final project: 50% Redoing students: The final exam can be re-taken in the next exam session. Grades from the coding exercises and final project will be retained for the full academic year. Students who do not pass will be considered as first attendance students in the next academic year.

-

Bibliography	1. M. Haverbeke. 2018. Eloquent JavaScript, 3rd ed. 2. K. Simpson. 2015. You Don't Know JS, 1st ed. 3. R. Martin. 2011. The Clean Coder, 1st ed.

-

Content

12.31 Semester 5 - Natural Language Processing

Responsible	**Dr. Schommer**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	7	1	O	4	56 h.	120 h.

Prerequisites	**The processing of natural language as well as the understanding of complex language-related problems as part of Artificial Intelligence is one of the most important aspects of computer science in general. This is not just due to the natural communication among human beings, but also due to daily applications: actual examples include the communication with/between robots, chatbots, and artificial companions. Also, the area of machine learning provides an essential basis for the solution of natural language problems such as machine translation, emotion detection or the comprehension of texts. The aim of this course is to contribute to a better understanding of the meaning of NLP and to provide solutions to current problems. RECOMMENDED: INTELLIGENT SYSTEMS I** **Part B: mandatory: programming skills in any language, desired: practival experiences with Python.**
Description	Part [A]: The processing of natural language as well as the understanding of complex language-related problems as part of Artificial Intelligence is one of the most important aspects of computer science in general. This is not just due to the natural communication among human beings, but also due to daily applications: actual examples include the communication with/between robots, chatbots, and artificial systems in general. Also, the area of machine learning provides an essential basis for the solution of natural language problems such as machine translation, emotion detection or the comprehension of texts - just to name a few. The aim of this part is to motivate the theoretical background and to contribute to a better understanding of the meaning of NLP. Part [B]: You learn in practice how to solve problems using Natural Language Processing tools and libraries. We mainly use Python as a programming language. We use Jupiter Notebooks, NLTK and spaCy for 'classical' NLP tasks such as tokenisation, part-of-speech tagging, parsing and nam,ed entity recognition. You will learn how to apply these techniques for solving problems such as Named Entitiy Recognition and Text summarisation. You become familiar with OpenMT to try out machine translation tool. You will also use natural language understanidng libraries such as RASA and DialogFlow to build chatbots.
Evaluation	First session: 50% Final examination on part [A] of the course 50% Practical deliverables provided for Part [B] Redoing session: Written examination covering knowledge acquired during part [A] and Part [B]

-

Bibliography	Part [A]: James Allen: Natural Language Understanding (Pearson), Christopher Manning, Henning Schütze: Foundations of Statistical Natural Language Processing (MIT Press), David Jurafsky, James Martin: Speech and Language Processing (Prentice Hall), Stuart Russel, Peter Norvig: Artificial Intelligence, A Modern Approach (Pearson), Steven Bird, Ewan Klein, Edward Loper: Natrual Language Processing with Python (O' Reilly). Part [B]: To get familiar or refresh Python: Charles Severance: Python for Everybody. https://www.py4e.com (2016, Python 3.0 edition).

-

Content

1. Part [A] Natural Language Processing
1.1. Introduction
1.1.1. Course Organisation , Aims and Goals , Contents
1.2. Foundations
1.2.1. Stemming, Part-of-Speech Tagging, n-grams, similarity measures and distances
1.2.2. Syntax, Parsing, Grammars
1.2.3. Dictionaries and the representation of words, Vector Space model
1.2.4. Ambiguity, Word sense disambiguation, semantic aspects
1.2.5. Text processing: statistical approaches
1.3. Selected Applications
1.3.1. Sentiment Detection, Topic Modeling, and others
1.4. Summary, Review
1.4.1. Course Summary, Preparation for the final examination
1.5. Student Project
1.5.1. Project with NLTK
2. Part [B] NLP in Practice
2.1. Intro
2.1.1. Intro, how-to, principles. Team building activities, background of the students. What is language: data session. Jupiter Notebooks.
2.2. Basic techniques
2.2.1. Tokenisation, part-of-speech tagging, stemming, soundex, parsers
2.3. Applications
2.3.1. Named Entity Recognition, Text representaion, Indexing
2.4. Semantic analysis
2.4.1. Thesauri, Ontologies, Sentiment tagging
2.4.2. Sentiment analysis, Text summarisation
2.5. Machine translation
2.5.1. Rule-based, phrase-based, sequence-to-sequence tools
2.6. Dialogue
2.6.1. Intents, RASA, DialogFlow, AIML, Pandorabots/ProgramD
2.7. Final session: do your favourite
2.7.1. Students choose what they do in this session, they can re-do a practical to improve their mark or experiment with one of the tools that we used during the semester.

12.32 Semester 5 - Online Course (OL)

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	8	1	O	4	120 h.	120 h.

Prerequisites	**Consult the 'Online Course' section of the BiCS Study Programme Annex Reference Document**
Description	The BiCS program offers in the list of optional courses the possibility to follow selected 'Online Courses' (OL Courses). The rules that must be followed concerning the selection / execution and evaluation of online courses are indicated in the 'Online Course' section of the BiCS Study Programme Annex Reference Document. The list of possible online courses (provided in this course card) can change each semester and it is advised to consult the following the course folder to get details and access to the available courses descriptions and registrations. The course folder path in the BiCS students chared folder (https://dropit.uni.lu/invitations?share=99a33b312f6853c03ed0) is Courses/OnlineCourses
Evaluation	Consult the 'Online Course' section of the BiCS Study Programme Annex Reference Document

Bibliography	N.A.

Content

12.33 Semester 6 - Bachelor Semester Project 6

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
6	1	1	M	10	70 h.	300 h.

Prerequisites	**- For all BSP, meetings participation and deliverables submission are mandatory** **- For each non-first BSP, a mandatory participation pre-requisite is to have submitted BEFORE THE FIRST DAY OF THE SEMESTER a valid (description and tutor) validated (i.e. started using the tool) by a tutor using the online tool.** **For complete participation pre-requisites, see full official bachelor semester project reference document available here:** **https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0**
Description	This course is a preparation to the Bachelor Semester Projects (BSP) that a BiCS student will have to do idnividually during semester 2 to semester 6 of his studies. The students are introduced to research and development projects. The topics covered are: - project management - scientific methods for research and development projects - definition of scientific and technical deliverables - conducting a state of the art - scientific and technical report production - design and realization of oral and video presentations - soft skills A micro-BSP is used to learn by practice the necessary skills for a good execution of standard BiCS BSPs. During a normal BSP, students discover research and development domains, produce concrete artefacts related to computer science knowledge areas covered in the BICS, collaborate with UL employees in a project context, learn new technologies related to computer science, learn new knowledge related to computer science, apply the scientific and technical knowledge learned during the BICS, apply the primary and secondary languages knowledge learned during the BICS.
Evaluation	see full official bachelor semester project (BSP) reference document available here: https://dropit.uni.lu/invitations/?share=99a33b312f6853c03ed0

-

Bibliography	N.A.

-

Content

1. Software Engineering Management
1.1. Initiation and Scope Definition
1.1.1. Determination and Negotiation of Requirements
1.1.2. Feasibility Analysis
1.1.3. Process for the Review and Revision of Requirements
1.2. Review and Evaluation
1.2.1. Determining Satisfaction of Requirements
1.3. Software Project Planning
1.3.1. Determine Deliverables
1.3.2. Process Planning
2. Social Issues and Professional Practice
2.1. Professional Communication
2.1.1. Communicating professionally with stakeholders
2.1.2. Dynamics of oral, written, and electronic team and group communication (cross-reference HCI/Collaboration and Communication/group communication, SE/Project Management/team participation)
2.1.3. Reading, understanding and summarizing technical material, including source code and documentation
2.1.4. Writing effective technical documentation and materials
2.2. Professional Ethics
2.2.1. Community values and the laws by which we live
2.2.2. Keeping up-to-date as a computing professional in terms of familiarity, tools, skills, legal and professional framework as well as the ability to self-assess and progress in the computing field
2.2.3. The nature of professionalism including care, attention and discipline, fiduciary responsibility, and mentoring
2.3. Group Dynamics and Psychology
2.3.1. Dealing with Multicultural Environments
2.3.2. Dealing with Problem Complexity
2.3.3. Dealing with Uncertainty and Ambiguity
2.3.4. Individual Cognition
3. Computing Foundations
3.1. Problem Solving Techniques
3.1.1. Analyze the Problem
3.1.2. Definition of Problem Solving
4. Digital Technologies
4.1. various
4.1.1. Digital technologies will be learned or applied depending on the project subject
5. Computer Sciences
5.1. various
5.1.1. Sciences will be learned or applied depending on the project subject

12.34 Semester 6 - Software Engineering 2

Responsible	**Dr. Le Traon**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
6	2	1	M	4	56 h.	120 h.

Prerequisites	**software engineering 1** **Programming Languages (in particular OO programming)** **Database basics**
Description	Software engineering is the discipline concerned with the application of theory, knowledge, and practice to effectively and efficiently build reliable software systems that satisfy the requirements of customers and users. The purpose of this course is two address the fundamentals and practicals of two key areas of software engineering: - Part 1. Testing and Validation (1/2) - Part 2. Big Data and AI based software development (1/2) — Part 1. Software Testing and Validation Testing is the predominant technique used by the software industry to make software reliable, making software testing a challenging and exiting research field. The goal of testing and validation is to assess the consistency/conformity of a product with respect to its specification.These activities are thus crucial and costly activities for software companies, and eventually aim at providing a controlled level of trust in the final product, before it is delivered to the client (and then during maintenance). Testing is related to all the design stages of the development process and must deal with many application contexts (embedded systems, mobile applications, information systems ...) and various dimensions of complexity (programming-in-the-small, in-the-large and in-the-duration). Besides the fundamentals of software testing, the focus will be on practical techniques that can be applied in real-world modern software development cycles (agile, continous integration). — Part 2. Big Data and AI based software development (1/2) The race for developing efficient decision-support services and prescriptive recommendation systems is challenging, but inescapable due to the emergence of new societal and technical paradigms, such as IoT, CPS, Smart systems, Industry 4.0, Fintech. The main goal is to spark the discussion about the tradeoffs between the classical data processing techniques and the upcoming ones for big data. The course will remind the basics of databases, statistics and machine learning, and will focus on querying/processing/storing very large amounts of data as well as on devising a machine-learning based system. The course will study how to develop software that manipulate a potentially huge amount of complex, heterogeneous, temporal data streams for analytics purpose, leveraging machine learning algorithms. The use cases that we will study will be taken from, or inspired by, real-world industrial software.
Evaluation	The evaluation will be done through a final exam (70%) and evaluation through practical exercises (30%).

-

Bibliography	To be defined

-

Content

1. Software Engineering (SE)
1.1. Software Testing and Validation
1.1.1. Software Testing and Validation: fundamentals
1.1.2. Unit Testing and Diagnosis
1.1.3. Integration Testing
1.1.4. Requirements and System Validation
1.1.5. Transversal aspects to functional testing

1.2. Big Data software development
1.2.1. Basics on stats, databases, machine learning
1.2.2. Temporal data and graph databases
1.2.3. Model-driven analytics
1.2.4. Prescriptive analytics
1.2.5. Modelling expert knowledge
1.2.6. Vizualization and dashboarding

12.35 Semester 6 - Security 2

Responsible	**Dr. Mauw**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
6	2	2	O	4	56 h.	120 h.

Prerequisites	**Successful completion of the course Security 1 is RECOMMENDED.**
Description	This course allows students to obtain in-depth knowledge from a selection of areas in the field of Computer Security. This year, the focus will be on the following topics: - Attack trees: applied to e-passport, applied to space systems. - Attack trees - student led (a slection of papers in attacks / defence trees) - Information Flow overview: Chinese Wall + Messeguer & Goguen models - Hardware, Verifying Information Flow Goals in Security Enabled Linux, Xen or ... something adaptable - Proverif-based verification of security protocols - ePayment protocol in Proverif, Mafia fraud proof in Proverif - Multiparty computation - Blockchain basic - Blockchain privacy and scalability These topics will be treated while taking the following three learning dimensions into account: - Topical knowledge (i.e. theoretical and technical knowledge of the topic). - Academic skills (e.g. reading, presentation, summarizing, writing, pitching, discussing, critical thinking, writing of an abstract, formulating research questions, poster development). - Understanding the context (e.g. ethical considerations, legal aspects, societal relevance).
Evaluation	

Bibliography	The lecturers will provide links to academic papers and other study material.

Content

1. Introduction
1.1. Introduction
1.1.1. Background knowledge
2. Security protocols
2.1. Security protocols
2.1.1. General concepts
2.1.2. Secrecy
2.1.3. Authentication
3. Risk assessment
3.1. tobefilledin...
3.1.1. tobefilledin...
4. Hardware separation
4.1. tobefilledin...
4.1.1. tobefilledin...
5. Multiparty computation
5.1. tobefilledin...
5.1.1. tobefilledin...
6. Blockchain operation and security
6.1. tobefilledin...
6.1.1. tobefilledin...
7. Fair exchange on the blockchain
7.1. tobefilledin...
7.1.1. tobefilledin...

12.36 Semester 6 - User Centered Design

Responsible	Dr. Bongard-Blanchy

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
6	2	3	O	4	42 h.	120 h.

Prerequisites	none
Description	This course puts a strong emphasis on the practical usage of UX design methods and provides the appropriate theoretical basics required to apply the methods. You will gain advanced knowledge about several user-centered design techniques and methods that you can implement to improve and evaluate user experiences. The course will include user research methods to gain a better understanding of your target audience (including experimental design and data analysis) and design strategies for user-centered problem-solving. The practical part consists of a hands-on project carried out in small groups. You are free to bring your own project.
Evaluation	At the beginning of the course students will form small groups. Throughout the course, students will work collaboratively on their projects. Each group is required to show their week-to-week homework during the course to demonstrate how they applied the methods. The final evaluation will be based on each group's project presentation. If the delivered homework assignment does not meet the expected learning objective, the students will be asked to rework the deliverable.

-

Bibliography	The necessary material will be introduced throughout the course.

-

Content

1. User-Centered Design
1.1. Introduction, course organisation and structure
1.1.1. Introduction of lecturer, overview of course structure, learning goals, deliverables, warm-up exercise, team assignment.
1.2. Discover phase
1.2.1. interview
1.2.2. survey
1.3. Define phase
1.3.1. personas, experience mapping
1.3.2. how might we, story boarding
1.4. Develop phase
1.4.1. brainstorming, user story mapping
1.4.2. user story cards, prototyping
1.5. Deliver phase
1.5.1. user testing
1.5.2. libraries such as fontawesome, google material, design guides, inspirational resources
1.5.3. interview, questionnaire, observation
1.5.4. reporting, goals, roles and stakeholders
1.6. Project Presentations
1.6.1. each team presents their learning outcomes
1.7. Wrap-up
1.7.1. wrap-up: lessons learned and other UX methods such as Design sprint, UX resources (newsletter, dribbble...)

12.37 Semester 6 - Data Science for Humanities

Responsible	Dr. Schommer

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
6	2	4	O	4	56 h.	120 h.

Prerequisites	**RECOMMENDED: Information Management I, II and Intelligent Systems I. MANDATORY: strong interest in the field of Digital History and Humanities.**
Description	The course is multi-disciplinary and concerns selected aspects of Data Science as a preparatory step towards the application of intelligent methods for questions related to Digital History and Humanities. For this reason, the course will contain guest lectures as well as interdisciplinary exercises. As a learning outcome, each participant should understand the interdisciplinarity of data science and should understand how intelligent methods should be applied. Selected aspects of the data life-cycle will be concerned, for example Data Quality, Data Visualisation, Data mining, Data management, and Data retrieval. The course is organised as a lecture and is accompanied by a training sessions.
Evaluation	Summer semester: 50% (oral or written) exam + 50% Practical Studies Redoing session: - Winter semester: 100% oral or written exam - Summer semester: the course has to be redone entirely (50% Practical studies + 50% Final (oral or written) examination)

Bibliography	The literature will be published in the course.

Content

1. Data Science for Humanities
1.1. Introduction
1.1.1. Course Overview, Contents, Aims and Goals
1.2. Management of data
1.2.1. Structured, semi-structured, unstructured data, Relational system, XML-based systems
1.3. Retrieval of data
1.3.1. XPATH, XQuery, SQL, Alternative ways of Retrieval
1.4. Quality Aspects
1.4.1. Data Preprocessing Techniques: Data Cleaning, Data Transformation, Data Discretistation, Data Sampling, and others
1.4.2. Selected Statistics: Probability, Precision, Recall, F-score, BoxPLots, and others
1.4.3. Privacy aspects for Data Publishing: k-anonymity, l-diversity, t-closeness
1.5. Data + Text Mining
1.5.1. Link Analysis, Collocations, n-grams, Text Classification, Clustering, Training
1.6. Visualisation Aspects
1.6.1. Cognitive Aspects of Visualisation, Effectiveness and Expressiveness, Lie factor, selected techniques
1.7. Research stories
1.7.1. Researchers from C2DH, Computer Science Dept, or others talk about their Data Science projects.
1.8. Project or Exercises
1.8.1. Programming and Project Report

12.38 Semester 6 - Computational Science 3

Responsible	**Dr. Aleksandrova**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
6	2	5	O	4	56 h.	120 h.

Prerequisites	**RECOMMENDED Linear Algebra I and II** **RECOMMENDED Analysis for Applications I.** **RECOMMENDED Computational Science I.**
Description	Computational Science is a field of applied computer science, that is, the application of computer science to solve problems across a range of disciplines. It combines computer simulation, scientific visualization, mathematical modeling, computer programming and data structures, networking, database design, symbolic computation, and high performance computing with various disciplines. This area offers exposure to many valuable ideas and techniques, including precision of numerical representation, error analysis, numerical techniques, parallel architectures and algorithms, modeling and simulation, information visualization, software engineering, and optimization.
Evaluation	Final grade = 0.5 * continuous control + 0.5 * project Redoing evaluation: IF continuous control < 10, then final grade = redoing exam grade ELSE: final grade = max(redoing exam grade, 0.5 * continuous control + 0.5 * redoing exam grade)

Bibliography	Communicated during lecture.

Content

1. Computational Science (CN)
1.1. Eigensystems
1.1.1. Jacobi Transformations of a symmetric matrix
1.1.2. Householder Reductions to Tridiagonal Form
1.1.3. Eigenvalues and Eigenvectors of a Tridiagonal Matrix
1.1.4. Hermitian Matrices
1.1.5. QR Algorithm for real (Hessenberg) matrices
1.2. Fast Fourier Transform
1.2.1. Introduction and Recap: Fourier Transform of discrete data
1.2.2. The Fast Fourier Transform (FFT) concept
1.2.3. FFT of real functions
1.2.4. FFT in more dimensions
1.3. Spectral Applications
1.3.1. Convolution and deconvolution
1.3.2. Correlation and autocorrelation
1.3.3. Wavelets
1.4. Modeling of Data
1.4.1. Least squares as a Maximum Likelihood Estimator
1.4.2. Straight line with errors
1.4.3. Non-linear models: Levenberg-Marquardt
1.4.4. Confidence Limits
1.4.5. Support vector machines and K-means clustering
1.4.6. Markov Chain Monte Carlo
1.5. Overview of Application fields related to research activities at UL
1.5.1. Overview of Application fields related to research activities at UL

12.39 Semester 6 - Intelligent Systems 2

Responsible	Dr. TBD

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
6	2	6	O	4	42 h.	120 h.

Prerequisites	none
Description	Many intelligent systems such as autonomous robots or vehicles are embedded in a complex dynamic environment. In order to perceive this environment and intelligently interact with it, they need sensing and sensor processing, where one of the most important approaches is based on optical sensors (such as cameras) and computer vision. This course covers the basic principles of computer vision, the image acquisition and formation, image processing, feature extraction up to the usage of multiple images and recent approaches for computer vision based on artificial intelligence and machine learning.
Evaluation	- 3 projects / assignments (75% of the grade) - final exam (25% of the grade) - REMOTE with an assignment/project - students having failed the course will be given a new project which will then count for 100% of the grade

Bibliography	Corke, P.: Robotics, Vision and Control. Springer, 2013.

Content

1. Intelligent Systems (IS)
1.1. Introduction to Computer Vision
1.1.1. application examples, basic objectives
1.2. Light and Color
1.2.1. Spectral Representation of Light, Color
1.3. Image Formation
1.3.1. Perspective Transform
1.3.2. Camera Calibration
1.3.3. Non-Perspective Imaging Models
1.3.4. Unified Imaging
1.4. Image Processing
1.4.1. Obtaining an Image
1.4.2. Monadic Operations
1.4.3. Diadic Operations
1.4.4. Spatial Operations: Convolution, Template Matching, Non-Linear Operations
1.4.5. Shape Changing: Cropping, Image resizing, image warping
1.5. Image Feature Extraction
1.5.1. Region Features: Classification, representation, description
1.5.2. Line Features
1.5.3. Point Features, corner detectors
1.6. Using Multiple Images
1.6.1. Feature Correspondence
1.6.2. Stereo Vision
1.6.3. Structure and Motion

12.40 Semester 6 - Online Course (OL)

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
6	2	7	O	4	120 h.	120 h.

Prerequisites	**Consult the 'Online Course' section of the BiCS Study Programme Annex Reference Document**
Description	The BiCS program offers in the list of optional courses the possibility to follow selected 'Online Courses' (OL Courses). The rules that must be followed concerning the selection / execution and evaluation of online courses are indicated in the 'Online Course' section of the BiCS Study Programme Annex Reference Document. The list of possible online courses (provided in this course card) can change each semester and it is advised to consult the following the course folder to get details and access to the available courses descriptions and registrations. The course folder path in the BiCS students chared folder (https://dropit.uni.lu/invitations?share=99a33b312f6853c03ed0) is Courses/OnlineCourses
Evaluation	Consult the 'Online Course' section of the BiCS Study Programme Annex Reference Document

Bibliography	N.A.

Content

13 Legends

- BiCS Program Acronyms legends
  - Types
    - M: course is mandatory for each student
    - O: course is an optional
    - OF: course is an outside field course
  - MR is an abbreviation for Module Reference code
  - CR is an abbreviation for Course Reference code
  - L. is an abbreviation for Language
  - S. is an abbreviation for Semester
  - W. is an abbreviation for workload. Indicates the number of working hours expected to be executed by the student including all activities.

References

[1] du Grand-Duché de Luxembourg, J.O.: Loi du 27 juin 2018 ayant pour objet l'organisation de l'Université du Luxembourg. (MÉMORIAL A - N°587) (July 2018)

[2] du Grand-Duché de Luxembourg, J.O.: Arrêté ministériel du 5 mai 2020 portant approbation du règlement des études de l'Université du Luxembourg. (MÉMORIAL B - N°1785) (May 2020)

[3] du Grand-Duché de Luxembourg, J.O.: Arrêté ministériel du 3 décembre 2019 portant approbation du règlement d'ordre intérieur de l'Université du Luxembourg. (MÉMORIAL B - N°4169) (December 2018)

[4] ACM/IEEE-CS Joint Task Force on Computing Curricula: Computer science curricula 2013. (December 2013)

[5] Bachelor in Computer Science: BiCS Semester Projects Reference Document. Technical report, University of Luxembourg (2020)