BiCS
Bachelor in Computer Science
University of Luxembourg, LU

BiCS

Academic Year 2019/2020

Study Programme Annex Reference Document

bicshub.uni.lu

Thursday 27th February, 2020 - 14:58

Contents

1 Introduction

This document provides the necessary organization information concerning the BiCS - Bachelor in Computer Science. The targeted audience 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
- external students or instructors: to answer any questions they could have concerning the program content.

The BiCS promotion rules include, in addition to the ones included in this document, all the official promotion rules provided in the official documents of the university of Luxembourg which are:

1. The document [1]: which contains the "Règlement grand-ducal du 22 mai 2006 relatif à l'obtention du grade de bachelor et du grade de master 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 as of the 1st of August 2018".

Majority of legal documents are provided in French and not translated. It is up to the student to ensure he has access to the understanding of the content of the official documents. In case new versions of those documents are published they replace automatically the one listed above.

The BiCS is a program whose first promotion start date is September 2017 and thus it will offer the 6 study semesters only starting in September 2019. Furthermore it is expected major revisions of this document in its first development periods. It is thus encouraged to consult new versions of this document each semester.

2 Basic programme 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 **FSTC** faculty of sciences, technology and communication.

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 you need to provide an official certificate or a letter from your high school language professor(s) that certifies your level. The minimum CEFR equivalent level at application time is B2 for English and B1 for French or German (you decide yourself which secondary language you choose).

2.6 Registrations fees

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

3 Objectives and Programme 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

The bachelor in computer science targets some main learning outcomes. Upon successful completion of the BiCS programme, a graduate:

- 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 **60 students**.

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

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

The decision over admission offers is subject to a validation by the study programme adjunct director.

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 they are elligible and have a final grade satisfying the selection model thresholds constraints.

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

1. Motivation letter
2. Level in mathematics sicne 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)
9. Quality of the curriculum vitae

the grades are determined exclusively using the application file documents. There is no interview (phone, visio-conference or physical).

5 Organisation - General Rules

The programme has the following general organisation rules:

1. it requires to obtain at least **180 ECTS** to obtain the bachelor degree.
2. it is a **full time** programm which features a regular course load equivalent to full time studies (30 ECTS per semester). It cannot be followed by students that cannot participate to the programme activities from Monday to Friday and from 8am to 7pm.
3. there is no specialisation track
4. All modules contain only one course for all semesters except semester 6.

6 Organisation - Specific Rules

6.1 Rational

6.1.1 Course Types: Mandatory, Optional, Outside field, ...

The BiCS program contains different course types which are the following:

- **Mandatory (M):** this course must be validated by any student registered to the semester whose program includes this course.
- **Optional (O):** the student is free to choose or not this course. Specific constraints can nevertheless be added thus restricting the freedom of choice for the student.

The opening of an optional course depends 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.

The Bachelor is granted to a student if and only if:

- he succeeds to each of the mandatory courses
- he 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.
- he satisfied all the other rules included in the legal documents (including this one, the laws, ...)

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 in a prior semester or in the semester in which the student is registered to.
- AND the course is open for registration at semester start[1].

[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

As illustrations, with the current BiCS program (available in 2019/2020) is it possible for a student to get the bachelor 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	22	26	30	26	34
5	30	34	26	34	38	34
6	30	34	38	26	26	34
Total	180	180	180	180	180	>180

Table 1: Possibles ways to get at least 180 ECTS

The following subtypes 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 it is a CS subtype.

6.1.2 Outside Field Course rules

An outside field course (OF course or OFC for short) is a course with main objective to provide the student with the opportunity to acquire knowledge which belongs to a knowledge field outside the Computer Science field.
An outside field course has the following general properties. Additional individual properties can be provided in the OF course description provided in section 9 of this document

- Not more than three outside field courses can be validated during the whole bachelor program.
- The list of OF courses offered is included in this document and will be regularly updated.

6.1.3 Online Course

The optional courses list of the BiCS can include courses offered online (called BiCS Online Course and named "OL course" or "OLC" for short). In this part you can find the general rules that apply to OL courses.

Basic rules

1. Any student can chose up to three OL courses during his bachelor studies
2. A student registered in BiCS semester S_j can only register to online courses offered in semester S_i with $i <= j$.

3. An OL course must be selected among the ones proposed in section 9 of this document [2].

4. If a student wants to select the OL course ol_i for semester S_j, he must proceed to the following process:

   (a) before end of semester S_j he must fill the online form.

   (b) before end of semester S_j he must inform by email to the BiCS program director that he filled the online form and includes in his email: his student's identification; for each online courses selected, the title of the online course and the semester for which he wants this online course to be allocated.

   (c) during semester S_j he must proceed to the administrative tasks (i.e. registrations, exam,...) for all the online courses he wants to be evaluated for. THIS PROCESS IS MANDATORY AND HANDLED BY THE PROGRAM SECRETARY AND NOT BY THE COURSE RESPONSIBLE.

   (d) he must submit his deliverables for each online course using the moodle assignment. By default all deliverables must be submitted in one compressed file using Moodle. In case the Moodle submission fails, a cloud service must be used and all necessary information to download the file must be communicated to the course responsible in due time.

OL Evaluation

The OL course responsible, sets a grade for each OL followed by a student which is under his responsibility.
For each OL course followed by a student, the following deliverables are to be provided by the student to the OL course responsible:

1. the OL account access information provided using the online form (required for verification by the university of Luxembourg)

2. a validated successful course certificate

3. a written detailed report that must include the following parts:

   - personal detailed description of the activities made to acquire the course knowledge (reuse of course material provided by the online course website is forbidden).
   - a work-time report that should demonstrate that the student spent the adequate (i.e. ECTS quantity x 30 hours) amount of work time to acquire the knowledge contained in the OL course.

4. a 10' screencasted video and audio presentation from the student that should demonstrate the work done by the student to acquire the knowledge contained in the OL course.

[2] You can propose other online course to the BiCS director if you want him to evaluate the possibility to include it in the official list

The OL course responsible provides a grade between 0 and 20 based on the deliverables described above evaluating the activities made by the student to acquire the targeted knowledge as defined by the online course with the adequate amount of work.
The OL course responsible work is counted for 2 TU (teaching unit) per student.

OL Fees
The majority of OL courses selected for the BiCS program imply the payment of registration fees. Those fees are to be covered by the student (e.g. "Coursera" asks around 45€/month for a specialization). Thus selecting a OL course implies the commitment by the student to pay the necessary fees for the selected course. Depending on the BiCS budget it might be the case that a limited number of OL course fees will be paid by the BiCS. In this case, a quantity of OL course paid positions will be communicated to the students and allocated following a selection process.
In any case, OL are optional and not mandatory, thus a student will never be obliged to select a non free online course during his bachelor.

Available Courses
The list of possible OL courses can change each semester and it is advised to consult the following webpage to get details and access information to the available courses.

6.1.4 Mobility rules Outgoing Students

The BiCS bachelor program includes a mobility semester in accordance with the University of Luxembourg regulation documents.

- Warnings

  1. The mobility semester is by default the fourth semester of the program.
  2. The BiCS jury can decide to postpone the mobility semester depending on each student specific situation.
  3. **If a student fails to go in mobility during the 4th semester** it implies that the student cannot get his bachelor degree in three years and thus will not be able to register to a master degree until he validates his mobility semester and gets his bachelor degree.

All the rules, at university level, concerning the mobility semester are communicated to the students by the Mobility service of the University of Luxembourg.
The 4^{th} semester program presented in this document thus may concern only 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 don't have to select an OFC course in the 4th semester.

6.1.5 Mobility rules for Incoming Students

Incoming student doing their mobility semester at university of Luxembourg can select courses from the BiCS Bachelor in Computer Science.
Optional courses may not be always open each year and each semester, thus you need to contact the faculty single point of contact for mobility to be informed about the open courses (**Mobility Contact**).
The following rules are specific to the BiCS Bachelor in Computer Science to determine the courses that can be included in the learning agreement:

- **Bachelor Semester Project Courses**
  - You must be physically present at the first session at semester start at the University of Luxembourg
  - You cannot be registered to another course which is planned during the Mondays. Monday is a full day dedicated to your project 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 Bachelor Semester Project from your learning agreement.
  - Bachelor Semester Projects necessitate a huge amount of work (10 ECTS = 300 hours of work). So be prepared for this if you select this course.

6.1.6 Languages

The BiCS Bachelor implements applied multilingualism and each student will have his studies in which he will practice English as primary language and a secondary language freely chosen between French and German. During each semester, students practice along all their studies on their language deliverables related to their **computer science projects**.
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 secondary language must be 45, then each student has to select the courses of his studies in a way that will ensure him to have around 25% of his ECTS obtained using courses labelled according to his secondary language. The possibilities to do so are the following:

- Do a mobility semester in a university providing a program in the secondary language of the student
- Choose BSPs (Bachelor Semester Projects) labelled with his secondary language
- Choose outside field courses labelled with his secondary language

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

6.2 Promotion Rules

6.2.1 Course Examination Registration Pre-requisites (CERP)

Each student must be registered to the examination for each course he wants to be evaluated for. Not being registered to a course examination implies that the student will not be evaluated for that course at the next jury session and thus it will not be possible to get the course associated ECTS.
To be allowed to register to the examination for a course, each student must satisfy the pre-requisites stated for this course. Course examination registration pre-requisites (CERP) are available in this document and are updated and published after each official jury (winter and summer).
It is thus the responsibility of the student to consult the CERP for all the courses he is concerned by before the start of each semester to handle in the best possible way his studies. CERP are provided in two ways:

1. **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.

2. **Local CERP** that apply to only one course and that are provided inside each course card given in section 9. If the term **MANDATORY** is used then the condition must be satisfied to be allowed to register to the examination of the concerned course. IF the term **RECOMMENDED** is used then the student is free to request his registration to the examination for this course but the course responsible indicated that there are high risks of failure if the recommended conditions are not satisfied. This is to help the student to plan his studies in the best possible way especially if he has to retake some course examination.

6.2.2 Global CERP

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_1 **(resp.** $BSP\ S_1$**),... for short**.

The following default rules hold for any student registered to a BSP (see exceptions section in [5]):

1. The following jury sessions 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)
2. by default a jury session cannot validate the grades of two different BSP.
3. by default a jury session cannot validate the grade for BSP_i (i $>$ 1) if the ECTS for the BSP_{i-1} were not previously acquired.

4. the Late Summer jury 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.

- Main Exceptions:

  1. If you encountered health issues for which you have official medical documents you can enter a specific request by sending a letter to the program director through the secretary (which should give you a receipt). This letter should describe your request and should be received, if possible, at least 2 weeks before the jury that should validate your grade.
  2. If you failed to a BSP_i and you and your tutor believe that the necessary work to validate the BSP is minimal then you can enter a request (at last the first week of the semester) to the **BiCS** director to be allowed to register to the BSP_i and BSP_{i+1} during the same semester.

More details on the BSP are the provided in the BSP reference document [5].

7 General Regulation Rules

We would like to highlight some important general regulation rules that concern the program and that, for some of them, are already provided in general documents available to students:

1. Plagiarism is a breach of the law.
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 (cf. section VI of [1]).
4. If plagiarism is detected it is possible that all students having common parts will support the sanction.
5. By default, any artifact produced by a student and requested by an instructor is the result of an individual production.
6. In case of individual homework or project, students must be very careful in not being convinced of plagiarism. Especially in case some cooperation with other student has been made as a preparatory work for the homework or project.

8 Study Plan Overview

Below are given the summary tables providing essential information about each semester [3].

8.1 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/FR/DE	Bachelor Semester Project 1	10	300

8.2 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

8.3 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	3	90
3	3	1	M	EN	Algorithms and Complexity	3	90
3	4	1	M	EN	Information Management 1	5	150
3	5	1	M	EN	Security 1	5	150
3	6	1	M	EN/FR/DE	Bachelor Semester Project 3	10	300

[3] *See appendix here for legends*

8.4 Semester 4

S.	MR	CR	Type	L.	course	ECTS	W.
4	1	1	M	EN	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

8.5 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	Intelligent Systems 2	4	120
5	5	1	O	EN	Information Management 3	4	120
5	6	1	O	EN	Computational Science 1	4	120
5	7	1	O	EN	Online Course (OL)	4	120
5	8	1	O	EN	Natural Language Processing	4	120
5	9	1	O	EN	Theoretical Computer Science 3	4	120

8.6 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	Sofware 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 2	4	120
6	2	6	O	EN	Computational Science 3	4	120
6	2	7	O	EN	Online Course (OL)	4	120

9 Study Plan

9.1 Semester 1 - Linear Algebra 1

Responsible	**Dr. HE**

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	**Mid-term / written 30%** **final exam / written 50%** **Assignments 20%.** **Redoing session: 100% written exam, no kept grade.** **(all evaluations are individual by default)**

Bibliography	To be defined

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

9.2 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	**Mid-term exam / written 35%** **final exam /written 55%** **assignments 10%** **Redoing session: 100% written exam, no kept grade.**

Bibliography	To be defined

Content

1. Analysis for Applications 1
1.1. Functions
1.1.1. Basic functions: polynomials, 1/exp(x,k) , exp, log, cos, sin, tan, cosh, sinh, tanh.
1.2. Sequences of real numbers
1.2.1. Sequences of real numbers
1.2.2. Limit of a sequence
1.3. Continuous real functions
1.3.1. Limit of a function
1.3.2. Continuous functions
1.3.3. Limits at infinity, limits of basic functions
1.4. Derivatives
1.4.1. Definition of derivatives
1.4.2. Derivatives of basic functions
1.5. Taylors expansions
1.5.1. Definition and existence of Taylor expansions
1.5.2. Taylor expansions of basic functions
1.5.3. Finding limits using Taylor expansions
1.6. Metric spaces
1.6.1. Metric spaces, sequences in metric spaces, limits
1.6.2. Continuous function between metric spaces
1.6.3. Compact and connected metric spaces

9.3 Semester 1 - Discrete Mathematics 1

Responsible	**Dr. Van der Torre and Libal**

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

Prerequisites	**none**
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	**Homework / submission: 10%** **Midterm exam / written: 30%** **Final exam / written: 60%** **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

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

9.4 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: 60%** **Practicals - continous: 40%** **Redoing evaluation:** **If (last practical grade >= 10)** **then** **- written exam: 60%** **- last practical grade: 40%** **else** **- final exam: 100%**

-

Bibliography	

-

Content

9.5 Semester 1 - Bachelor Semester Project 1

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
1	5	1	M	10	300 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 a valid (description and tutor) validated by the tutor and the director 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=5ed11de976552dad0af3**
Description	During this project the students will: 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. At the end of the first bachelor semester project it is expected that the student is autonomous in finding a subject and a tutor for his next bachelor semester project.
Evaluation	**see full official bachelor semester project (BSP) reference document available here:** **https://dropit.uni.lu/invitations?share=5ed11de976552dad0af3**

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

9.6 Semester 2 - Linear Algebra 2

Responsible	**Dr. Perucca**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
2	1	1	M	4	56 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	**final exam / written 80%** **assignments 20%.** **Redoing session: written 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

9.7 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

9.8 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: 25%.** **- Mid-term exam: 35%.** **- Final exam: 40%.** **- 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

9.9 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

9.10 Semester 2 - Programming Fundamentals 2

Responsible	**Dr. Rothkugel**

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

Prerequisites	**RECOMMENDED: Programming Fundamentals 1**
Description	Building upon the basic skills acquired in Programming Fundamentals 1, this course covers the object-oriented programming paradigm, both at a theoretical as well as at a practical level. Students will learn the fundamental concepts of object-orientation, and how to apply them for designing and implementing applications of average complexity.
Evaluation	**continuous control / submission: 30%** **presentation: 10%** **final exam / written: 60%** **exam redo / written: 100%**

Bibliography	To be defined

Content

1. Programming Languages (PL)
1.1. Object-Oriented Programming
1.1.1. Object-oriented design: Decomposition into objects carrying state and having behavior, Class-hierarchy design for modeling
1.1.2. Definition of classes: fields, methods, and constructors
1.1.3. Subclasses, inheritance, and method overriding
1.1.4. Dynamic dispatch: definition of method-call
1.1.5. Subtyping (cross-reference PL/Type Systems): Subtype polymorphism, implicit upcasts in typed languages, Notion of behavioral replacement: subtypes acting like supertypes, Relationship between subtyping and inheritance
1.1.6. Object-oriented idioms for encapsulation: Privacy and visibility of class members, Interfaces revealing only method signatures, Abstract base classes
2. Software Development Fundamentals (SDF)
2.1. Algorithms and Design
2.1.1. Fundamental design concepts and principles: Abstraction: Program decomposition: Encapsulation and information hiding: Separation of behavior and implementation
2.2. Development Methods
2.2.1. Debugging strategies
2.2.2. Simple refactoring
2.3. Advanced Programming Constructs
2.3.1. Exception Handling
2.4. Basic Type Systems
2.4.1. Generic types (parametric polymorphism): Definition, Use for generic libraries such as collections, Comparison with ad hoc polymorphism (overloading) and subtype polymorphism
2.4.2. Using collection classes, iterators, and other common library components
3. Software Engineering (SE)
3.1. Software Design
3.1.1. Design Patterns
3.2. Functional Programming
3.2.1. Effect-free programming: Function calls have no side effects, facilitating compositional reasoning, Variables are immutable, preventing unexpected changes to program data by other code, Data can be freely aliased or copied without introducing unintended effects from mutation
3.2.2. First-class functions (taking, returning, and storing functions)

9.11 Semester 2 - Bachelor Semester Project 2

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
2	6	1	M	10	300 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 a valid (description and tutor) validated by the tutor and the director 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=5ed11de976552dad0af3**
Description	During this project the students will: 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. At the end of the first bachelor semester project it is expected that the student is autonomous in finding a subject and a tutor for his next bachelor semester project.
Evaluation	**see full official bachelor semester project (BSP) reference document available here:** **https://dropit.uni.lu/invitations?share=5ed11de976552dad0af3**

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

9.12 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 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

9.13 Semester 3 - Programming Fundamentals 3

Responsible	**Dr. Rothkugel**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	2	1	M	3	42 h.	90 h.

Prerequisites	**RECOMMENDED Programming Fundamentals 1** **RECOMMENDED Programming Fundamentals 2** **RECOMMENDED Computing Infrastructures**
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	**midterm exam / practical: 40%** **final exam / written: 60%** **exam redo / written: 100%**

-

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

9.14 Semester 3 - Algorithms and Complexity

Responsible	**Dr. Kelsen**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	3	1	M	3	42 h.	90 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: 60%** **Practicals - continous: 40%** **Redoing evaluation:** **If (last practical grade >= 10)** **then** **- final exam: 60%** **- last practical grade: 40%** **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

9.15 Semester 3 - Information Management 1

Responsible	**Dr. Theobald**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	4	1	M	5	70 h.	150 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)

9.16 Semester 3 - Security 1

Responsible	**Dr. Cardoso dos Santos**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	5	1	M	5	72 h.	150 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	**First Half (security and symmetric crypto)** **Homeworks / submission: 50%** **Second Half (public key crypto)** **Homeworks / submission: 50%** **Final grade is the average of both halves.** **Redoing session:** **During the Winter semmester: Attend the course again, delivering homework and.** **During the Summer Semmester: Final Exam.**

-

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

9.17 Semester 3 - Bachelor Semester Project 3

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
3	6	1	M	10	300 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 a valid (description and tutor) validated by the tutor and the director 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=5ed11de976552dad0af3**
Description	During this project the students will: 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. At the end of the first bachelor semester project it is expected that the student is autonomous in finding a subject and a tutor for his next bachelor semester project.
Evaluation	**see full official bachelor semester project (BSP) reference document available here:** **https://dropit.uni.lu/invitations?share=5ed11de976552dad0af3**

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

9.18 Semester 4 - Bachelor Semester Project 4

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
4	1	1	M	10	300 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 a valid (description and tutor) validated by the tutor and the director 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=5ed11de976552dad0af3**
Description	During this project the students will: 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. At the end of the first bachelor semester project it is expected that the student is autonomous in finding a subject and a tutor for his next bachelor semester project.
Evaluation	**see full official bachelor semester project (BSP) reference document available here:** **https://dropit.uni.lu/invitations?share=5ed11de976552dad0af3**

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

9.19 Semester 4 - Information Management 2

Responsible	**Dr. Theobald and Libal**

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	**Regular 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 students:** **- 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)

9.20 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 tempeoral logics (LTL and CTL), and their model chekcing algorithms will be introduced. The course also covers techniques that deal with the state-space explosion problem, if possible.
Evaluation	**First exam session:** **Written exam: 60%** **Practical exercises: 40% (the best 80% of all exercises)** **Resit:** **Written exam: 60%** **Grade for practical exercises from first exam session: 40%**

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

9.21 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	**intermediate exam / oral in mandatory weekly course sessions: 20%** **continuous control in mandatory weekly course sessions / submission: 20%** **final exam / written: 60%** **Redoing session:** **If (previous Intermediate exam + previous continuous control)/2 >= 10 then** **- written exam: 60%** **- previous Intermediate exam 20%** **- previous continuous control 20%** **else written exam: 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.
1.3.3. Separate compilation, linking.
1.4. Program Analysis
1.4.1. Static analysis and formal validation/verification.

9.22 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

9.23 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 possiblity 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 webpage to get details and access to the available courses: https://goo.gl/wvSLvj
Evaluation	**Consult the 'Online Course' section of the BiCS Study Programme Annex Reference Document**

Bibliography	N.A.

Content

9.24 Semester 5 - Bachelor Semester Project 5

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	1	1	M	10	300 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 a valid (description and tutor) validated by the tutor and the director 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=5ed11de976552dad0af3**
Description	During this project the students will: 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. At the end of the first bachelor semester project it is expected that the student is autonomous in finding a subject and a tutor for his next bachelor semester project.
Evaluation	**see full official bachelor semester project (BSP) reference document available here:** **https://dropit.uni.lu/invitations?share=5ed11de976552dad0af3**

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

9.25 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:** **- mandatory participation to weekly course sessions** **- if final exam grade < 6** **then final grade = final exam grade** **else final grade = final exam grade * 0.5 + continuous control * 0.5** **Redoing evaluation:** **If (previous continuous control >= 10)** **then** **[if final exam grade <= 6** **then final grade = final exam grade** **else final grade = final exam grade * 0.5 + continuous control * 0.5** **]** **else** **- written exam: 100%**

-

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)

9.26 Semester 5 - Human-Computer Interaction (HCI)

Responsible	**Dr. Distler**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	3	1	M	4	65 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 (cognition, emotion, needs, perception...), HCI principles and key terms (user interface, usability, user experience, accessibility, ...), and selected sub-disciplines. The second part teaches methods of user-centered design. Every session is an interactive training session, comprised of theoretical input from the lecturers and training activities where students discuss and apply the topics of the session (e.g. hands-on experiences, discussions, work on sub-projects). [1] Churchill, Bowser, Preece (2013): Teaching and Learning Human-Computer Interaction. https://interactions.acm.org/archive/view/march-april-2013/teaching-and-learning-human-computer-interaction
Evaluation	**After being introduced to the fundamentals of HCI in the first half of the course, students will form groups and define a topic with 5 sub-projects (4 short written documents for each step in the user-centered design process & one final presentation with reflections on the topic) to learn about HCI methodology. In each session of the second half, they will work collaboratively on their topic, following a user-centered design process.** **As an evaluation, each group will provide four short written documents (about 1-2 pages, to be delivered throughout the second half of the semester) about how they applied the methods of a user-centered design process. At the final session, they will present their results and reflect on the process of working on their topic.** **The four documents and the final presentation will be graded independently. The final grade will be the average of these four documents and the final presentation.** **Redoing conditions: an individual topic will be provided to registered redoing students at beginning of the semester on explicit request to the responsible professor. The four deliverables will be requested and an oral presentation will be made.**

Bibliography	Students receive a suggested reading list in the first session. An example of a general HCI textbooks for interested students is: MacKenzie, I. Scott (2013). Human-computer interaction : An empirical research perspective. San Francisco: Morgan Kaufmann.

-

Content

1. Human-Computer Interaction (HCI)
1.1. Foundations
1.1.1. Contexts for HCI (anything with a user interface, e.g., webpage, business applications, mobile applications, and games)
1.1.2. Processes for user-centered development, e.g., early focus on users, empirical testing, iterative design
1.1.3. Different measures for evaluation, e.g., utility, efficiency, learnability, user satisfaction
1.1.4. Usability heuristics and the principles of usability testing
1.1.5. Physical capabilities that inform interaction design, e.g., color perception, ergonomics
1.1.6. Cognitive models that inform interaction design, e.g., attention, perception and recognition, movement, and memory, gulfs of expectation and execution
1.1.7. Social models that inform interaction design, e.g., culture, communication, networks and organizations
1.1.8. Principles of good design and good designers, engineering tradeoffs
1.1.9. Accessibility, e.g., interfaces for differently-abled populations (e.g., blind, motion-impaired)
1.1.10. Interfaces for differently-aged population groups (e.g., children, 80+)
1.2. Designing Interaction
1.2.1. Principles of graphical user interfaces (GUIs)
1.2.2. Task analysis, including qualitative aspects of generating task analytic models
1.2.3. Low-fidelity (paper) prototyping
1.2.4. Quantitative evaluation techniques, e.g., keystroke-level evaluation
1.2.5. Help and documentation
1.2.6. Handling human/system failure
1.2.7. User interface standards
1.3. Programming Interactive Systems
1.3.1. Interaction Design Patterns: visual hierarchy, navigational distance
1.3.2. Choosing interaction styles and interaction techniques
1.3.3. Design for resource-constrained devices (e.g. small, mobile devices)
1.4. User-Centered Design & Testing
1.4.1. Approaches to, and characteristics of, the design process
1.4.2. Functionality and usability requirements (cross-reference to SE/Requirements Engineering)
1.4.3. Techniques for gathering requirements, e.g., interviews, surveys, ethnographic and contextual enquiry
1.4.4. Techniques and tools for the analysis and presentation of requirements, e.g., reports, personas
1.4.5. Prototyping techniques and tools, e.g., sketching, storyboards, low-fidelity prototyping, wireframes
1.4.6. Evaluation without users, using both qualitative and quantitative techniques, e.g., walkthroughs, GOMS, expert-based analysis, heuristics, guidelines, and standards
1.4.7. Evaluation with users, e.g., observation, think-aloud, interview, survey, experiment
1.4.8. Challenges to effective evaluation, e.g., sampling, generalization
1.4.9. Reporting the results of evaluations
1.4.10. Internationalization, designing for users from other cultures, cross-cultural
1.5. New Interactive Technologies
1.5.1. Choosing interaction styles and interaction techniques
1.5.2. Representing information to users: navigation, representation, manipulation
1.5.3. Approaches to design, implementation and evaluation of non-mouse interaction: Touch and multi-touch interfaces ,Shared, embodied, and large interfaces ,New input modalities (such as sensor and location data) ,New Windows, e.g., iPhone, Android ,Speech recognition and natural language processing (cross reference IS/Natural Language Processing) ,Wearable and tangible interfaces ,Persuasive interaction and emotion ,Ubiquitous and context-aware interaction technologies (Ubicomp) ,Bayesian inference (e.g. predictive text, guided pointing) ,Ambient/peripheral display and interaction
1.6. Human Factors & Security
1.6.1. Applied psychology and security policies
1.6.2. Security economics
1.6.3. Regulatory environments – responsibility, liability and self-determination
1.6.4. Organizational vulnerabilities and threats
1.6.5. Usability design and security
1.6.6. Trust, privacy and deception
1.7. Design-Oriented HCI
1.7.1. Intellectual styles and perspectives to technology and its interfaces
1.7.2. Consideration of HCI as a design discipline: Sketching ,Participatory design
1.7.3. Critically reflective HCI: Critical technical practice ,Technologies for political activism ,Philosophy of user experience ,Ethnography and ethnomethodology
1.7.4. Indicative domains of application: Sustainability ,Arts-informed computing

9.27 Semester 5 - Intelligent Systems 2

Responsible	Dr. Voos

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	4	1	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	**- 4 projects / exercices (group of 3 students): 75%** **- final exam: 25%** **- 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

9.28 Semester 5 - Information Management 3

Responsible	**Dr. Theobald**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	5	1	O	4	60 h.	120 h.

Prerequisites	**RECOMMENDED Information Management 1 & 2**
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. Data Warehousing
1.1. OLTP/OLAP
1.1.1. Data warehousing concepts
1.1.2. Extract-transform-load (ETL)
1.1.3. Online transaction processing (OLTP)
1.1.4. Online analytical processing (OLAP)
1.1.5. Multidimensional data model & data cubes
1.1.6. OLAP extensions to SQL

1.2. Business Intelligence
1.2.1. Data mining concepts and techniques
1.2.2. Associative and sequential patterns
1.2.3. Market basket analysis
1.2.4. Clustering and classification
1.2.5. Data cleaning
1.2.6. Data visualization (cross-reference GV/Visualization and CN/Interactive Visualization)

9.29 Semester 5 - Computational Science 1

Responsible	**Dr. Navet**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	6	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 used to make design choices. The course relies on group projects where you will be asked to solve real-world problems. You will be able to evaluate 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, * Exact and approximate optimization techniques such as constraint programming and nature-inspired heuristics.
Evaluation	**- Three projects (groups of two students): 75%** **- Final exam (MCQ): 25%.** **- Students having failed the course will be given a new project, which will then count for 100% of the grade.**

Bibliography	- 'Reinforcement learning in motion', Phil Tabor, available on a-z.lu - 'Search and Optimization by Metaheuristics - Techniques and Algorithms Inspired by Nature', Ke-Lin DuM. N. S. Swamy, available on a-z.lu

Content

1. Computational Science (CN)
1.1. Processing
1.1.1. Introduction to Neural networks and Deep Neural Networks. Application to a case-study (project 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 - project 2)
1.1.3. Numerical methods: algorithms for numerically fitting data, architectures for numerical computation including parallel architectures
1.1.4. Introduction to optimization algorithms: optimal techniques (convex optimization, integer programming, constraint programming) and heuristics (e.g. nature inspired algorithm, simulated annealing). Multi-objective optimization.
1.1.5. 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. Formal models and modeling techniques: mathematical descriptions involving simplifying assumptions and avoiding detail. Examples of techniques include: Monte Carlo methods, Queuing theory, game theory. Application to market risk evaluation with Monte Carlo simulations (project 3).
1.3.6. Assessing and evaluating models and simulations in a variety of contexts, verification and validation of models and simulations
1.3.7. Important application areas including economics and finance, city and urban planning, science and engineering
1.3.8. 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

9.30 Semester 5 - Online Course (OL)

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	7	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 possiblity 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 webpage to get details and access to the available courses: https://goo.gl/wvSLvj
Evaluation	**Consult the 'Online Course' section of the BiCS Study Programme Annex Reference Document**

Bibliography	N.A.

Content

9.31 Semester 5 - Natural Language Processing

Responsible	**Dr. Schommer**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	8	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] To be defined
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.

9.32 Semester 5 - Theoretical Computer Science 3

Responsible	**Dr. Pang**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
5	9	1	O	4	42 h.	120 h.

Prerequisites	**Previous knowledge in either of the two research areas (formal verification and machine learning) is helpful but not mandatory. RECOMMENDED: Theoretical Computer Science 1 / Theoretical Computer Science 2/ Linear Algebra 1/ Linear Algebra 2/ Intelligent Systems 1**
Description	Formal verification aims to guarantee correctness properties of software and hardware systems. The modelling and verificaiton of such systems is critical, especially for safety-critical systems. Formal verification attempts to answer the question: 'does a system model satisfy a specification, which consists of properties of interest describing the desired behaviours of the system? '. Typical verification techniques include theorem proving (automated deduction) and model checking. On the other hand, machine learning aims to learn patterns from training data and it can be seen as a systematic way of solving a problem by optimising some objective function using training data. Machine learning tasks can be classified into supervised learning and unsupervised learning, as well as reinforcement learning. The course focuses on the connections of these two research areas and it will cover the following topics (but not limited to): (1) formal verification of neural networks, (2) model learning, (3) synthesis of programs and algorithms, and (4) invaraint learning. A seed lession on the research areas will be given at the beginning of the course, and the students will then select one topic on combing formal verification and machine learning. The remainder of the course will be composed of seminar sessions organised with the students on the selected topics.
Evaluation	**Continuous control: 30%** **Report: 40%** **Presentation: 30%** **Redoing session:** **Report: 50%** **Presentation: 50%**

-

Bibliography	Examples of papers to be studied during the course: 1. Safety verification of deep neural networks (Xiaowei Huang, Marta Kwiatkowska, Sen Wang, Min Wu. Proceedings of CAV 2017) 2. Model learning (Frits W. Vaandrager. Commun. ACM 60(2):86-95. 2017) 3. Learning a static analyzer from data (Pavol Bielik, Veselin Raychev, Martin Vechev. Proceedings of CAV 2017) 4. ICE: A robust framework for learning invariants (Pranav Garg, Christof Loding, P. Madhusudan, Daniel Neider. Proceedings of CAV 2014)

-

Content

1. Theoretical Computer Science
1.1. Model Checkers
1.1.1. The state-space explosion problem, LTL and CTL model checking algorithms
1.2. Formal Methods
1.2.1. Roles of formal methods
2. Software Engineering (SE)
2.1. Formal Methods
2.1.1. Model checking, formal verification
2.2. Software Verification and Validation
2.2.1. Verification and validation concepts, static and dynamics approaches to verification
3. Artificial Intelligence (AI)
3.1. Machine learning
3.1.1. Basic concpets, algorithms, and statistical methods
3.2. Deep learning
3.2.1. Neural networks, optimisation

9.33 Semester 6 - Bachelor Semester Project 6

Responsible	**Dr. Guelfi**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
6	1	1	M	10	300 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 a valid (description and tutor) validated by the tutor and the director 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=5ed11de976552dad0af3**
Description	During this project the students will: 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. At the end of the first bachelor semester project it is expected that the student is autonomous in finding a subject and a tutor for his next bachelor semester project.
Evaluation	**see full official bachelor semester project (BSP) reference document available here:** **https://dropit.uni.lu/invitations?share=5ed11de976552dad0af3**

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

9.34 Semester 6 - Sofware 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

9.35 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: - Security protocols - Risk assessment - Hardware separation - Multi-party computation - Blockchain operation and security - Fair exchange on the blockchain 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). In this course, the students will not simply consume the information offered by the lecturers. They will have to actively read academic papers, perform background research, present their findings and engage in discussion with other students.
Evaluation	**For each of the 6 topics, the students will have to perform a number of tasks, which will determine the grade for each topic. The final grade will be the average of the grades for the 6 topics.**

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

Content

9.36 Semester 6 - User Centered Design

Responsible	**Dr. Niess**

Sem.	Module ref.	Course ref.	Type	ECTS	Volume	Workload
6	2	3	O	4	65 h.	120 h.

Prerequisites	**none**
Description	This course consists of a mix of theoretical and practical parts. You will gain more advanced knowledge about several user-centered design techniques and methods that you can implement to design and evaluate positive user experiences. The course will include 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.
Evaluation	**At the beginning of the course students will form small groups. Throughout the course, students will work collaboratively on their projects. As an evaluation, each group will give four intermediate presentations plus a final project presentation. Each presentation is also to be handed to the lecturer in PDF format. Each presentation and the connected PDF document will be graded independently.**

Bibliography	Students will receive a reading list in the first session of the course. Recommended books: Lazar, J., Feng, J. H., & Hochheiser, H. (2017). Research methods in human-computer interaction. Morgan Kaufmann. Peerce, J., Rogers, Y., & Sharp, H. (2002). Interaction design beyond human-computer interaction, vol 25 (11).

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. User-Centered Design / design research
1.2.1. User-centered design applied (study design, data analysis, methods,. . .), planning and conducting design research
1.3. User-Centered Design / idea generation
1.3.1. Design methods, design exercises, focus on idea generation (e.g. reframing exercise)
1.4. User-Centered Design / advanced prototyping methods
1.4.1. Scenarios, user journeys, low-fidelity prototypes (variation in materials)
1.5. User-Centered Design / evaluation methods
1.5.1. Initial evaluation of prototypes, conceptualising study design
1.6. User-Centered Design / ethical design, dark patterns,
1.6.1. Data analysis, considering unethical alternatives, design fiction
1.7. User-Centered Design / design fiction
1.7.1. Integrating design fictions in the interaction with users
1.8. User-Centered Design / ethical prototypes
1.8.1. Data analysis, higher fidelity protypes => avoiding ethical fallacies in interaction design
1.9. User-Centered Design / advanced research methods
1.9.1. Conceptualising mixed-method study to evaluate high-fidelity prototypes
1.9.2. Analysing mixed-method study to evaluate high-fidelity prototypes
1.10. Preparation for final presentation
1.10.1. Preparation of prototypes, presentations, posters, time for open questions
1.11. Final presentation
1.11.1. Final presentation

9.37 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

9.38 Semester 6 - Computational Science 2

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	**intermediate exam / written: 20%** **continuous control / submission: 30%** **final exam / written: 50%** **Redoing evaluation:** **If continuous control < 10, then final grade = redoing exam grade** **If continuous control >= 10, then** **final grade = max(redoing exam grade, redoing exam grade*0.7+ continuous control*0.3)**

-

Bibliography	Communicated during lecture.

-

Content

9.39 Semester 6 - Computational Science 3

Responsible	**Dr. Beex**

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

Prerequisites	**RECOMMENDED: 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, 3 exercises will be evaluated (around week 5, 9 and 13). At the end of the course, each student must give a 10min presentation about the exercises. The final grade is the average of the grades for the exercises and the grade for the presentation.** **Redoing students:** **- have to submit the exercise solutions on weeks 5, 9 and 13 using the Moodle assignments tool (they will receive the exercise subjects using Moodle).** **- have to do a presentation during the exam session** **- have their final grade as the average of the four grades received**

-

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

9.40 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 possiblity 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 webpage to get details and access to the available courses: https://goo.gl/wvSLvj
Evaluation	**Consult the 'Online Course' section of the BiCS Study Programme Annex Reference Document**

Bibliography	N.A.

Content

10 Courses Maintenance

In this section you'll find information on courses that were stopped and which necessitate some maintenance actions.
Any stopped course is maintained for its examination until there are no more students still needing to pass those courses.
You will find below for each of those courses the necessary information concerning examination of stopped courses.

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 primary language deliverables provided for his 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 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.

11 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] : Loi du 27 juin 2018 ayant pour objet l'organisation de l'Université du Luxembourg. Journal Officiel du Grand-Duché de Luxembourg (MÉMORIAL A - N°587) (2018)

[2] University of Luxembourg: Règlement des études de l'Université du Luxembourg. Technical report, University of Luxembourg (2018)

[3] University of Luxembourg: Arrete ministeriel du 29 aout 2018 portant approbation du reglement d'ordre interieur de l'Universite du Luxembourg. Technical report, JOURNAL OFFICIEL N° 2499 du 14 septembre 2018 (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 (2019)