COMPUTER PROGRAMMING

Course Objective

Knowledge and understanding:

Students learn problem solving by implementing algorithms in computer programs.
Students learn the basic concepts of programming languages.
Applying knowledge and understanding: Students are trained in writing small computer programs, and applying the concepts learned.

Making judgements: Students are trained in testing and judging the correctness of their own programs, and to correct programming mistakes.

Lifelong learning skills: After completing the course, students will be able to grasp future programming languages and environments by applying the basic concepts learned here.

Course Content

In this course, students learn the basic concepts of (problem solving with) computer programming. We are using the C++ programming language.

The course consists of six modules:

The basics: data types, operators, type conversions, expressions, statements, if, switch, loops
Vectors, user-defined functions with parameter passing, scopes
Error handling, exceptions, streams and input errors, file IO, command line parameters, assertions and unit testing
Recursion: functions that call themselves, understanding how functions work
Classes: DIY data types, interfaces and implementation, constructors and initialisation
Memory management: pointers, dynamic memory allocation, memory leaks; Tree data structures as recursive data types
The six modules build on top of each other. Students reach the course objectives by mastering the higher-level modules five and six.

Additional Information Teaching Methods

This course revolves around self-study with the interactive course e-book, in combination with class sessions for deepening understanding (2 times per week), and practical programming assignments (individual preparation and guided lab sessions 3 times per week).

Method of Assessment

Throughout the course, the practical programming assignments assess the student's learning progress. The midterm exam assesses proficiency with modules one, two and three. The final exam assesses proficiency with all six modules.

Both exams are computer-based and contain programming assignments. There will be a resit exam, covering all six modules.

The midterm exam and the final exam together result in the exam grade. Here, the midterm counts for 30% and the final exam for 70%. If taken, the resit exam replaces the combined exam grade from midterm and final.

All practical programming assignments together result in the grade for the computer practical (for details, see Canvas).

For the practical programming assignments, there will be no resit. (The multiple assignments can be used to mutually compensate each other.)

For passing the course, the exam grade must be at least 6.0. The overall grade is the weighted average of the exam grade (65%) and the practical programming assignments (35%). This overall grade must be 5.5 or better for passing the course. Students with an overall grade of 5.5 or better can raise their total grade by up to 1 bonus point by timely completion of the exercises in the interactive e-book.

Entry Requirements

Students must possess the ability to clear, analytical and abstract thinking, as indicated by the level of mathematics required for admission to the program.

Literature

Programming in C++, Frank Vahid and Roman Lysecki, interactive e-book, https://www.zybooks.com/catalog/programming-c-plus-plus

The contents of this e-book have been tailored to our course.

Additional Information Target Audience

Bachelor Computer Science ( )

Explanation Canvas

Canvas is the place where all communication takes place. Students must make sure they receive Canvas notifications in a timely manner.

Recommended background knowledge

Strictly speaking, prior programming experience is not required for following this class. But it certainly helps. Prospective students who have never programmed before may want to spend some time during summer trying out programming in a very user-friendly way: We have prepared some introductory exercises using the Hedy programming language, accessible via the following link: https://hedy.org/hedy/l/4nWLrAf

The use of Hedy is free of cost. Students must create a Hedy account before use.

ENGLISH LANGUAGE TEST
Course Objective
At VU Amsterdam, having a strong command of the language of your study programme is crucial to your academic success. Therefore, all first-year Bachelor students must take a Dutch or English language proficiency test at the beginning of their studies. The required test depends on the language of instruction for your Bachelor's programme.

See also: Language Proficiency Test - Vrije Universiteit Amsterdam (vu.nl)

Course Content
The test consists of multiplechoice and fill-in-the-blank questions on the following topics:

Grammar
Spelling and punctuation
Structuring
Vocabulary/usage
Phrasing
Reading comprehension
Pronunciation
Classroom English
Additional Information Teaching Methods
The language proficiency test will take place in September in the DigiTenT at the VU.

Method of Assessment
The language proficiency test is a digital test consisting of 200 closed questions.

Additional Information
The VU English Language Test is linked to a first-year course. The final grade for this course is awarded only after passing the English Language Test. Students who do not obtain a satisfactory result for the test have to take the refresher course offered by the Taalloket.

For more information on the VU English Language Test, please see also: Language Proficiency Test - Vrije Universiteit Amsterdam (vu.nl)

Custom Course Registration
You do not need to register for the test. The Educational Office of your faculty will ensure your registration for the language proficiency test. Once registered, you can find the test’s date and time on your personal timetable. If you have not been registered by 8

September, please contact the student desk. You can also check thetimetable of your programme for the test schedule.

Ensure you know your VUnet ID and corresponding password, as you will need these details to log in to the language test. Tip: change your passwordto something you can easily remember.

LOGIC AND SETS FOR CS
Course Objective
After completing this course, the student can:

express logical statements in propositional and predicate logic (Knowledge and understanding) (Applying knowledge and understanding)
reason about the meaning of such formulas through truth tables and models (Applying knowledge and understanding) (Making judgements)
argue formally whether one formula implies another one, or that they are equivalent (Applying knowledge and understanding) (Making judgements)
reduce a propositional formula to disjunctive or conjunctive normal form (Knowledge and understanding) (Applying knowledge and understanding)
express propositional formulas in logic circuits and OBDDs (Knowledge and understanding) (Applying knowledge and understanding)
Furthermore, the student is able to:

reason about set constructions through Venn diagrams and the algebra of sets (Applying knowledge and understanding) (Making judgements)
construct and interpret formal, graphic, and matrix representations of sets, relations, and functions (Applying knowledge and understanding) (Making judgements)
determine and argue whether: (Applying knowledge and understanding) (Making judgements)
a. a relation is reflexive, transitive, symmetric, or antisymmetric.
b. a relation is an ordering relation, equivalence relation, or a function
c. a function is injective or surjective
construct and interpret compositions of relations (or functions) and their inverses (Applying knowledge and understanding) (Making judgements)
construct a proof by mathematical induction (Applying knowledge and understanding)
Course Content
The sets part of the course starts by introducing the concepts of sets, Venn diagrams, product sets and relations. The student then learns the main characteristics and properties of three particular types of relation: ordering relations, equivalence relations and functions. The sets part concludes with a study of the principle of mathematical induction.

The logic part focuses in the first place on propositional logic: truth tables, boolean operators, functional completeness, logical puzzles, SAT-solving, logic circuits and OBDDs. In addition the student will learn the meaning and use formulas of predicate logic, to express mathematical properties and sentences from natural language.

Additional Information Teaching Methods
Every week, there is one lecture and one exercise class for the logic part of the course, and one lecture and one exercise class for the sets part of the course.

Method of Assessment
A written midterm exam (40% of the grade) and a written final exam (60% of the grade). For both the midterm and the final exam, at least a 5.0 must be achieved (and the overall mark must be at least 5.5.).

The resit exam covers all material of the course.

It is not possible to resit only the midterm exam or only the final exam of the course.

Literature
All course materials are provided via Canvas.
Additional Information Target Audience
Bachelor Computer Science ( )

DATA STRUCTURES AND ALGORITHMS FOR CS
Course Objective
After taking this course, you:

have basic knowledge of data structures and algorithms,
know that different data structures are suitable for different computational problems, and
are able to analyze the performance of algorithms, with regard to their time and space requirements.
The following list presents the course objectives according to Dublin Descriptors.

Learn the notations and the language of data structures and algorithms. (Knowledge and understanding.)
Learn some basic data structures including linear data structures (e.g., arrays, linked lists, queues, and stacks) and several efficient dictionary data structures (e.g., binary search trees, AVL trees, and hash tables). Also, familiarize with performances of the different data structures with regard to their query and storage requirements. (Knowledge and understanding.)
Learn algorithms for solving some basic problems including sorting, searching, and a few graph-related algorithms like BFS and DFS.
Be able to use the data structures and the algorithms learned from the course to solve an algorithmic problem. (Applying knowledge.)
Given an algorithmic problem, pick the most appropriate data structure in order to speed up a designed algorithm for the problem and analyze its performances. (Making judgements.)
Familiarize with few real-life problems that can be modeled as an algorithmic problem and can be thus studied using algorithms and data structures. (Communication.)
Course Content
You will be acquainted with some basic linear data structures (e.g. arrays, lists, queues, and stacks), and several efficient dictionary data structures (e.g., binary search trees, AVL trees, hash tables). Although our lectures and discussions will be from a theoretical perspective, we will also discuss their implementation details whenever needed. Some fundamental algorithms will be discussed to solve basic problems: sorting, searching, and a few graph algorithms like BFS and DFS. It will be also explained how the time and space complexity of algorithms can be analyzed.

Additional Information Teaching Methods
There will be two lectures and one exercise sessions per week. Attendance is not mandatory, but it is strongly recommended.

Method of Assessment

There will be one midterm exam (50%),and one final exam (50%). Roughly speaking, the midterm exam covers the first half of the course and the final exam covers the second half of the course. Your final grade is simply the sum of the grades you have obtained for the two exams. There is no minimum grade requirements for two exams in order to pass the course, except that their sum has to be at least 55.

The resit of the course covers the entire course and it replaces both the midterm exam and the final exam.

Entry Requirements
Officially speaking, the course does not have any entry requirements, but the knowledge of the following topics is strongly advised in order to be able to follow the course.

calculus and in particular geometric and arithmetic series, and
discrete mathematics, and in particular mathematical induction.
Literature
Introduction to Algorithms (3rd edition), by Thomas H. Cormen, Charles, E. Leiserson, Ronald L. Rivest, and Clifford Stein, MIT Press 2009.

Explanation Canvas
We will use Canvas for our communications, posting announcements, publishing materials of the course (e.g., course slides, weekly exercises, and any other additional related materials).

Recommended background knowledge
As mentioned earlier, a good knowledge in calculus and discrete math is strongly advised in order to be able to follow the course.

PERVASIVE COMPUTING
Course Objective
We expect that by the end of this course, students will be able to:

Design a realistic smart system with the potential to benefit human lives. The system acquires and processes audio and video data and uses pattern recognition to make decisions that affect the environment accordingly. (Knowledge and understanding) (Applying knowledge and understanding) (Making judgments) (Lifelong learning skills)
Build a simplified version of this system, based on programmable microcontrollers. (Applying knowledge and understanding) (Making judgments) (Lifelong learning skills)
Work together in a team, collaboratively identifying not only the technical, but also the safety or ethical issues with their designs, and then sharing their challenges and discoveries through reports, presentations, and in-class demonstrations. (Lifelong learning skills) (Communication)
Course Content
Pervasive (or ubiquitous) computing is a trend based on Mark Weiser's vision of computers available "always and everywhere", embedded in everyday life. This course is an introduction to pervasive computing systems that assist people in their daily lives. Think about a self-driving car, a speech-controlled wheelchair, an automatic insulin pump for diabetic patients, or a navigation system for visually impaired pedestrians.

These systems:

sense the context (time, user's location, blood sugar concentration, road scenery, etc), recognize data patterns, reason and make intelligent decisions, and
act upon the environment, by controlling the wheels or the pump motor, suggesting the best route, or just notifying a caretaker.
The main components of such a system are sensors, controllers, and actuators. In this course, the students will learn different techniques to acquire signals from the environment, process these raw signals in order to infer context by using machine learning, and to write software agents for control. During the practical sessions, the students will get experience with these techniques and build their own microcontroller-based smart system. Programming is done in MATLAB and C++.

Guest lectures, given by researchers working in relevant fields are planned as well.

Additional Information Teaching Methods
Lectures, mandatory practical sessions.
Method of Assessment

Compulsory practical assignments and a digital exam in TestVision. The final grade is calculated as Final grade = (0.5*PRAC) + (0.5*EXAM). A pass requires both components to be >=5.5. It is possible to resit the exam, but not the practical.

Entry Requirements
Knowledge of programming in C++ is recommended.

Literature
Silvis-Cividjian, N. (2017), Pervasive Computing - Engineering Smart Systems, Springer International Publishing, ISBN 978-3-319-51654-7

Explanation Canvas
All materials can be found in Canvas.

WEB TECHNOLOGY
Course Objective
Everyone uses the Web, but how was it originally built, how has it evolved to what it is now, and how might it further develop in the future? The objectives of this course are to enable the student to formulate answers to these questions. While the course is primarily focused on technological aspects, as a professional in academia or industry, you need to be able to relate technological developments on the Web to relevant trends in science and society in general.

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

Knowledge and understanding: Understand, on an introductory level, the key ideas, languages, and protocols underlying the Web.
Applying knowledge and understanding: Be able to apply this knowledge by designing and implementing a basic Web application.
Making judgements: Be able to validate and reflect on the impact of technical design decisions on the functionality and usability in a Web context.
Course Content
The lectures and online materials teach you the key ideas languages and protocols underlying the Web. These ideas include separation of concerns in terms of content versus style and client versus server, and the notion of device independence. Important Web languages and protocols include URI, HTTP, HTML, CSS, and JavaScript. By participating in the practical sessions and by making the assignments, you will develop the skills you need to apply these techniques to create basic Web content yourself and to systematically validate and assess more advanced Web content made by yourself and others. The course will teach you the basic building blocks of the Web as it is today, and understanding how they are related will also help you understand future developments. During the course, you will:

create your own HTML content and formally validate it on syntactical correctness
create your own CSS style sheets to render the same HTML content differently on different devices
apply JavaScript code made by yourself or others and use it to enrich the interactive behavior of your own Web content, and learn how to debug such applications
build a Web server with a RESTfull HTTP API
write a technical report in which you evaluate your own website on international accessibility criteria and reflex on your previous assignments.
Additional Information Teaching Methods

Lectures (4x2 hours p/week), Practical sessions (2x4 hours p/week, attendance mandatory), homework (preparation lectures, preparation practical sessions, making assignments, preparing the exam) (26 hours p/week).

Method of Assessment
The final grade is determined from the following components:

Individual exam counts for 30%.
Three or four (group) assignments count for 60% (4x15% or 3x20%).
Making the exercises in the online textbook counts for 10%.
To pass the course both the final grade and the individual exam need to be equal or above 5.5.

There will be a resit for the individual exam only.

Literature
Provided online via ZyBook & Canvas.
Additional Information Target Audience
Bachelor Computer Science ( )
Custom Course Registration
Registration is compulsory at least 4 weeks before the course starts.
Group enrolment takes place in Canvas.
Explanation Canvas
Further information for this course will be made available online (Canvas). All students must be enrolled in the course Canvas community.

Recommended background knowledge
An introductory programming course, being able to use a command line interface such as bash or Powershell.

COMPUTER ORGANIZATION

Course Objective

Explain the basic concepts, historical objectives, and modern functions of digital computers. (Knowledge and understanding)

Describe the basic architecture and operation of digital computers. (Knowledge and understanding)

Use proficiently binary data representation, number representation, and arithmetic and data conversion. (Knowledge and understanding) (Applying knowledge and understanding)

Explain at a proficient level the architecture and operation of each of the main components of a digital computer: the basic processing unit, the hierarchical memory system, the I/O system, and the interconnectionsystem. (Knowledge and understanding) (Communication)

Explain at a basic level various system mechanisms for building faster single-node systems, such as pipelining and caching, and large-scale systems. (Knowledge and understanding) (Communication)

Demonstrate proficiency in implementing basic operations of digital computers in realistic scenarios. (Knowledge and understanding) (Applying knowledge and understanding) (Lifelong learning skills)

Analyze at a basic level the tradeoffs inherent to the design of digital computers, concerning among others performance (simple modeling), scalability (Amdahl's Law), availability, energy consumption, and cost. (Applying knowledge and understanding) (Making judgements) (Lifelong learning skills)

Course Content

Computers are everywhere, in industry, academia, governance, and many other activities that impact our society. But what are they? How do they work? Why do they currently work the way they do? How to analyze them and to improve their performance?

Matching the requirements of the IEEE/ACM CS Curriculum, topics for this course include: the architecture, the structure, the operation, and the interconnection of computer components into computer systems, including modern architectures, data representation, assembler programming, virtual machines, the structure of translators, compiling and loading, basic operating systems concepts (I/O, interrupt handling, process).

Additional Information Teaching Methods

Lectures 4h/week.

Tutorial (Instructie) 2h/week.

Practical work (Lab) 4h/week, from week 3.

Self-study in teams.
This course uses dr. Iosup's method for gamification.
Method of Assessment
(Mandatory) basic lab assignments. (Turn in to SAs)
(Mandatory) final exam, written, multiple-choice.
(Optional) in-class exercises, oral and written.
(Optional) mid-term exam, written, multiple-choice. The results of the mid-term exam count only if the final exam is also taken by the student, and only if it increases the final grade of the student.
(Optional) self-study booklet and exam pack. (Turn in to TA)
(Optional) advanced lab assignments. (Turn in to SAs)
All partial results (including the lab, and the mid-term and final exams) are only valid during one academic year.
The end grade is the total number of points accumulated across all assessment possibilities scored divided by 1000.
It is possible to score a perfect 10 as the final grade.

The different course activities are graded as follows:

Exam, multiple-choice questions (max 7,500 points)
Self-study assignments, including exam preparation (max 2,000 points)
Lab assignments (max 4,000 points)
In-class activity (max 50 points per session)
Various bonus activities.
Students do not have to complete every course activity but are free to choose their own path of advancement.

There is only one resit opportunity for the exam.

Literature
Books (students can pick any):

Carl Hamacher and Zvonko Vranesic, Computer Organization, 6th edition, McGraw-Hill Education, 2011. ISBN-13: 978-0073380650
David A. Patterson and John L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 5th edition, Morgan Kaufmann, 2013. ISBN-13: 978-0124077263
Additional Study Materials:

Course and Lab guides are also provided via Canvas.

Additional Information Target Audience
Bachelor Computer Science ( ).
Bachelor Mathematics ( )
Custom Course Registration
The course includes two types of group activities: lab and self-study.

Group enrolment takes place in Canvas. In the past few years, students could choose to attend lab sessions on one of two separate days. Choices are made available at the start of the course.

Recommended background knowledge
Mathematics students should have taken at least one prior course in computer programming.

REQUIREMENTS ENGINEERING

Course Objective

In this course, you will learn about requirements engineering practices. for eliciting, documenting, and managing requirements (Knowledge and understanding).

Through theory and hands-on exercises, you will gain insights into how these practices can be applied during software development (Applying knowledge and understanding).

Additionally, you will learn about different software lifecycle models and how requirements engineering activities fit into these models.

(Knowledge and understanding) (Applying knowledge and understanding)

After following the course, you will be able to reason and make decisions about adequate requirements engineering elicitation, documentation, and management techniques taking into consideration specific software project characteristics (Making judgements).

Additionally, you will be able to apply these techniques in concrete project examples. (Applying knowledge and understanding) (Lifelong learning skills)

Course Content

A good requirements specification is critical for software project success. This course gives an introduction to processes, methods, and representation forms for specifying and managing requirements. It also gives an overview of the whole software development lifecycle.

Topics include requirements elicitation, analysis, modeling, and validation. Current research topics such as continuous user participation and requirements mining will also be covered. Part of the course entails assignments performed in groups and individual non-graded exercises.

Additional Information Teaching Methods

The course will be composed of theoretical lectures and practical lectures in which individual exercises will be solved by students and discussed.

Method of Assessment

Assignments (group): 70%
Final written exam (individual): 30%
Quizzes (individual): pass/fail

There is a resit for the final written exam. There are no resits for any of the graded assignments or quizzes.

Note: should the lecture be given partially online, the assignments will count for a 100% of the grade and the final written exam will be in apass-fail modality.

Literature
Selected chapters (to be announced in lecture) of the following books:

Mandatory (We will only read specific chapters from this book; please, wait for the first lecture to decide if you should buy the book):

Requirements Engineering, Axel van Lamsweerde, Wiley, 2009.
Selected chapters of the following books (available in Canvas):

Karl Wiegers and Joy Beatty, Software Requirements, Microsoft Press, 2013.
Martina Seidl, Marion Scholz, Christian Huemer, Gerti Kappel, UML@Classroom: An Introduction to Object-Oriented Modeling, Springer, 2015.
Bernd Brügge and Allen H. Dutoit, Object-Oriented Software Engineering, Pearson, 2010.
Additional Information Target Audience
Bachelor Computer Science ( )
Recommended background knowledge
Basic programming experience. Some knowledge about the software lifecycle is of advantage.

COMPUTER NETWORKS
Course Objective
After successfully completing the Computer Networks course, you are able to:

Explain the basic principles and modern functions of computer networks and data communication. (Knowledge and understanding)
Describe the layered network architecture and explain the essential function(s) in these layers. (Knowledge and understanding)
Apply basic physics, theorems from information theory, and network protocol properties to calculate network properties such as maximum bit rate, minimum round-trip latency, window size, etc. (Applying knowledge and understanding)
Apply networking mechanisms and algorithms to detect and correct transmission errors, compute routing tables, set window sizes, etc. (Applying knowledge and understanding)
Explain the concept of Quality of Service, why different applications have different networking requirements, and how these requirements translate to network and protocol properties. (Knowledge and understanding) (MakingJudgements)
Explain issues related to performance and scalability in today's popular networking protocols. (Knowledge and understanding) (Making Judgements)
Demonstrate proficiency in socket programming by implementing basic networked applications. (Knowledge and understanding) (Applying knowledge and understanding)
Course Content
The emphasis in this course is on fundamental concepts in digital communication. In modern computer networks, data communication takes place by sending data from A to B via a layered architecture where each. layer implements a different abstraction. The higher layers are responsible for handling web pages, emails, and similar things, that are translated into packets, bits, and eventually digital signals on physical links (e.g., light pulses, electrical signals in copper wires, radio waves).

This layered architecture with increasing levels of abstraction and separation of concerns, is a fundamental approach that you will encounter in all aspects of computer science (and beyond). Within this architecture, we will concern ourselves with questions like: what route should the data follow through the network, what do we do when errors occur, how do we interconnect two networks that have completely different properties, etc.

Following the IEEE/ACM Computer Science curriculum, topics to be discussed include the layered network architecture, different types of networks (e.g., wired and wireless, LAN and WAN), multiplexing, error control, flow control, routing and forwarding, names and addresses, high-level architectures of networked applications (peer-to-peer, client/server, etc), performance issues, and scalability issues. These issues are discussed while exploring the technology behind the Internet and its popular protocols (e.g., TCP, UDP, Ethernet, Wifi, etc.).

Additional Information Teaching Methods
Lectures, labs, and tutorials. This course is gamified.
Method of Assessment
(Mandatory) Basic lab assignments. (Turn in to SAs)
(Mandatory) Final exam, multiple-choice.
(Optional) Mid-term, multiple choice. The results of the mid-term exam count only if the final exam is also taken by the student, and only if it increases the final grade of the student.
(Optional) Optional lab assignments. (Turn in to SAs)
(Optional) Self-study assignment. (Turn in to TAs)
(Optional) In-class exercises, oral and written.

All partial results (including the lab, and the mid-term and final exams) are only valid during one academic year.
The end grade is the total number of points accumulated across all assessment possibilities scored divided by 1000.
It is possible to score a perfect 10 as the final grade.

The different course activities are graded as follows:

Exam, multiple-choice questions (max 7,500 points)
Self-study assignments (max 2,000 points)
Lab assignments (max 4,000 points)
In-class activity (max 50 points per session)
There is only a resit opportunity for the exam.

Literature
Andrew S. Tanenbaum and David Wetherall, Computer Networks, 5th or 6th ed.

Additional Information Target Audience
Bachelor Computer Science ( )
Additional Information
Current information can be found on Canvas: canvas.vu.nl
Recommended background knowledge
University-level programming course.
Basic knowledge of the Python programming language.

DISCRETE MATH AND CALCULUS
Course Objective
The student knows basic concepts from graph theory and can solve problems about and with those in explicit situations. (Knowledge and understanding) (Applying knowledge and understanding)
The student knows basic theorems and algorithms from graph theory and can use these to compute and/or prove certain properties in explicit situations. (Knowledge and understanding) (Applying knowledge and understanding)
The student knows basic concepts from combinatorics and can solve problems about and with those in explicit situations. (Knowledge and understanding) (Applying knowledge and understanding)
The student knows basic theorems and techniques from combinatorics and can use these to compute and/or prove certain properties in explicit situations. (Knowledge and understanding) (Applying knowledge and understanding)
The student knows basic concepts from calculus and can solve problems about and with those in explicit situations. (Knowledge and understanding) (Applying knowledge and understanding)
Course Content
This course is about three subjects: graph theory, combinatorics and calculus.

Additional Information Teaching Methods
Lectures and tutorials.

Method of Assessment
Written exam.

Additional Information Target Audience
Bachelor Computer Science ( )

SYSTEMS PROGRAMMING PROJECT

Course Objective

Students learn to apply programming concepts in the context of a larger, individual project. After completing the course, students will be able to plan, develop, and test larger programs for correct behavior according to a given specification. (Knowledge and understadning) (Applying knowledge and understanding) (Making Judgements)

Course Content

In this course, you will implement an emulator capable of executing IJVM byte-code. The implementation will be done in C. The assignment is split into several smaller parts that build up to the final deliverable.

Additional Information Teaching Methods

One introductory lecture in the first week. 4 weeks of practical work.

Method of Assessment

The student's knowledge will be assessed by means of a practical programming assignment and an oral evaluation.

Both the practical assignment and the oral evaluation require a passing grade. The final grade consists of 100% of the grade of the practical programming assignment. The oral evaluation is graded with pass/ fail.

There will be no resits for either of the assessments.

Literature

The course syllabus.

Extra (optional) reading:

Ritchie, D. M., Kernighan, B. W., & Lesk, M. E. (1988). The C programming language. Englewood Cliffs: Prentice Hall.

Andrew, S. Tanenbuam. "Structured Computer Organization". Prentice-Hall, Inc. New Jersey (1999).

Recommended background knowledge

This course is a Programming practical so we expect you to be proficient at the level of Computer Programming (XB_40011). A good grasp of systems architecture is also recommended.

OBJECT-ORIENTED AND FUNCTIONAL PROGRAMMING

Course Objective

The goal of this course is to obtain familiarity and experience with advanced programming language concepts, such as inheritance and pattern matching, as well as improving general programming skills.

After taking this course, you will be able to:

Understand & apply concepts from object-oriented programming such as subtyping and inheritance. (Knowledge and understanding) (Applying knowledge and understanding)
Understand & apply concepts from functional programming such as pattern matching and higher-order functions. (Knowledge and understanding) (Applying knowledge and understanding)
Design and implement a moderately large program from scratch. (Applying knowledge and understanding) (Making judgements)
Produce clear, readable code. (Applying knowledge and understanding) (Making judgements) (Lifelong learning skills)

Course Content

The lectures cover various programming concepts from Object-oriented and Functional programming and how they occur in various programming languages. These topics include inheritance, subtyping, parametric polymorphism, side effects, higher-order functions, and pattern matching. The lectures also cover how to produce clear, readable code.

We will use the programming language Scala, as it supports a wide array of programming techniques.

There are several individual programming exercises that offer practice with writing programs with the concepts discussed in the lectures. These give the students experience with designing and implementing moderately large programs from scratch, train the students in producing clear, readable code, and provide an opportunity to use the advanced concepts from the lectures.

There are also several multiple-choice exercises that offer practice with reading programs that involve the concepts discussed in the lectures. These typically present a small program, with accompanying multiple-choice questions such as what is the output of this program, what is the type of certain function, what type of error is present, etc? These programs help with understanding the details of how the program concepts work. Such questions will be presented on canvas and in the lectures. The exam is a digital multiple-choice proctored exam taken at the VU which mainly presents these types of questions.

Additional Information Teaching Methods

Lectures: 4 hours per week (in total 28 hours). Practical sessions: 4 hours per week (in total 28 hours).

Method of Assessment

Written exam (40% of grade) and programming exercises (60% of grade).

Both grades have to be at least 5.5. There is no resit opportunity for the programming exercises.

Entry Requirements
Computer Programming (XB_40011) or Systems Programming Project (XB_0123) (formerly known as Computer Programming Project (X_400556)).

Literature
Materials will be provided through Canvas.
Additional Information Target Audience
Bachelor Computer Science ( )

STATISTICAL METHODS
Course Objective
After this course, the student should be able to:

understand the setup of a probability space, events, probability, and conditional probability; (Knowledge and understanding)
use the basic rules of probability theory (sum rule, product rule, law of total probability, Bayes' rule); (Applying knowledge and understanding)
understand what a random variable is, and what its distribution is, both for discrete and continuous random variables (this includes the concepts of probability mass function, probability density function, and cumulative distribution function); (Knowledge and understanding)
compute the expectation and variance of a random variable; (Applying knowledge and understanding)
compute the mean and variance of a sum of independent random variables; (Applying knowledge and understanding)
derive discrete probability distributions for probabilistic experiments associated with the Bernoulli, Binomial, Negative Binomial, and Poisson distributions; (Applying knowledge and understanding)
compute probabilities for continuous random variables, specifically the uniform, exponential, and normal distribution; (Applying knowledge and understanding)
do calculations based on the law of large numbers and central limit theorem; (Applying knowledge and understanding)
reproduce the definition of an estimator, do elementary computations on point estimators; (Applying knowledge and understanding)
understand what a confidence interval is and derive it in simple settings using a pivot; (Knowledge and understanding) (Applying knowledge and understanding)
understand the setup of classical hypothesis testing (critical region, p-value, type of errors) and apply this to simple hypothesis testing problems; (Knowledge and understanding)
derive the likelihood ratio test in simple settings; (Applying knowledge and understanding)
get basic knowledge of the statistical software package R; (Knowledge and understanding)
apply concepts introduced in the course in 2 assignments which are to be implemented using the statistical software package R. (Applying knowledge and understanding)
Course Content
From the book by From Pishro-Nik:

Chapter 1: Basic concepts
Chapter 3: Discrete random variables, Bernoulli, Binomial, Negative Binomial, Poisson
Chapter 4: Continuous random variables; sections 4.0, 4.1 and from 4.2 uniform, exponential and normal random variables
Chapter 7: Law of large numbers and central limit theorem; sections 7.0 and 7.1
Chapter 8: Classical statistics (point estimation, interval estimation, hypothesis testing); sections 8.1, 8.2, 8.3 and 8.4
Additional Information Teaching Methods

6 weeks with 4 hours lecturing each week. Exercise classes. Two R assignments, to be done in groups of 2.

Attendance to lectures and classes is not mandatory but strongly recommended.

Method of Assessment
The exam counts for 80%. Each of the assignments counts for 10%. To pass the course, the (unrounded) grade of the exam needs to be at least 5.

Entry Requirements
Calculus (functions, differentiation and integration) and elementary set theory.

Literature
Hossein Pishro-Nik, Introduction to Probability, Statistics and Random Processes.

The book can be ordered but is also freely available on www.probabilitycourse.com.

Additional Information Target Audience
Bachelor Computer Science ( )

NETWORKS AND GRAPHS
Course Objective
After taking this course, student will be able to formally reason about graphs and use them to solve fundamental computational tasks.

The course introduces basics of graph theory and graph algorithms. In particular, you will be able to:

Mathematically analyze basic properties of graphs
Apply graph algorithms in computational tasks such as shortest paths and spanning trees
Analyze performance of such algorithmic tasks
Learning goals:

G1: Learning basic concepts in graph theory. (Knowledge and understanding)
G2: Learning and reasoning about graphs and their properties (Knowledge and understanding)
G3: Familiarity with standard graph algorithms (Knowledge and understanding)
G4: Analyze performance and correctness of different algorithms (Knowledge and understanding) (Making judgements)
G5: Applying fundamental graph algorithms for solving computational tasks (Applying knowledge and understanding) (Communication)
G6: Developing problem solving skills by using algorithmic techniques such as dynamic programming and greedy (Applying knowledge and understanding)

Course Content
Many real world problem involve interactions on a network, which is mathematically represented by "graphs". Graphs are fundamental objects appearing many computational tasks all across computer science (e.g. computer networks, data analysis, databases, machine learning, etc). This course focuses on graph theory, graph algorithms, and their applications. We reason about graphs mathematically and along the way develop further skills to formally analyze algorithms.

The topics include basic concepts and techniques in graph theory and graph algorithms:

Intro to discrete math and review of proof techniques
Graph theory concepts
Directed graphs
Important graph classes (bipartite graphs, Eulerian graphs, Hamiltonian graphs, random graphs)
Connectivity
Shortest path algorithms
Minimum spanning tree algorithms
Cuts and flows
Additional Information Teaching Methods
There are two main lectures (at VU campus) per week. Moreover, there is one exercise class per week. Attendance is not mandatory but bonus points will be given based on active participation in exercise classes.

Method of Assessment
The final grade is determined by two written exams: The midterm and the final exam. The midterm accounts for 40 percent and the final exam accounts for 60 percent of the grade. Active participation in interactive exercise sessions will result in a bonus of up to %5 of the total grade. In these sessions students will work in groups to solve problems.

Literature
Lecture slides will be provided and textbook is not required, however the following textbooks are recommended:

Graph Theory, by J. A. Bondy , U. S. R. Murty (Hardcover ISBN: 978-1-84628-969-9) [available online at VU library]

Introduction to Algorithms, Fourth Edition, by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein (ISBN: 978-0262046305)

Additional Information Target Audience
Bachelor Computer Science ( )

Recommended background knowledge
Data Structures and Algorithms, Logic and Sets, Statistical methods.

OPERATING SYSTEMS
Course Objective
Get insight into the internals of modern operating systems. Both theory (architecture, design principles, interfaces, etc.) and practice (operating system implementation aspects, user-level systems programming in C, etc.).

Upon completion of this course, you will:

have a basic understanding of the operating systems design and implementation principles adopted by mainstream operating systems such as Linux (K&U)
have knowledge of key operating systems concepts, including: history and evolution of operating systems, core operating systems abstractions, system calls, virtualisation, basic and advanced operating systems architectures (K&U, AK&U, LS)
have knowledge of process management principles (K&U, AK&U, MJ)
have knowledge of memory management principles (K&U, AK&U, MJ)
have knowledge of file systems principles (K&U, AK&U, MJ)
have knowledge of input/output principles (K&U, AK&U, MJ)
have knowledge of concurrency principles (K&U, AK&U, MJ)
The acronyms above refer to the Dublin Descriptors covered by each learning objective, specifically:

Knowledge and Understanding (K&U),
Applied Knowledge and Understanding (AK&U),
Making Judgment (MJ),
Learning Skills (LS), and
Communication (C).
Course Content
This course gives an introduction to the internals of operating systems. The following topics are covered: operating system architectures, processes, threads, synchronisation, memory management, file systems, input/output, and virtualisation.

The course uses real-world operating systems such as Linux and MINIX 3 as examples, providing insights into both the theory and practice of modern operating systems.

Additional Information Teaching Methods
Lectures and practicals.
Method of Assessment
Written exam on the material covered in lectures and four individual programming assignments in C. The final grade is the weighted average of the exam grade (50%) and the assignments (50%).

To pass the course, the exam grade must be at least 5.5, each assignment grade must be at least 4.0, and the final grade must be at least 5.5. There is no resit opportunity for the programming assignments.

Entry Requirements
XB_40011 Computer Programming or XB_0123 Systems Programming Project (formerly known as X_400556 Computer Programming Project) .

Literature
Tanenbaum, A.S., Bos, H.J., "Modern Operating Systems", Fifth edition, Pearson.

https://www.pearson.com/en-us/subject-catalog/p/modern-operating-systems/P200000003295/9780137618880

Additional Information Target Audience
Bachelor Computer Science ( )
Recommended background knowledge
Good programming skills.

HISTORY OF SCIENCE FOR CS
Course Objective
Students acquire knowledge and understanding about the history of computing from various perspectives: computing as a scientific goal, computing as a government (administrative or military) objective / ideal, computing as an economic enterprise (Knowledge and understanding).

Students can apply that knowledge and understanding to describe the meanings digital culture has or may have for various people in contemporary society (Applying knowledge and understanding).

Students are able to communicate their knowledge and understanding in an effective way, and from various perspectives - thereby they will be better equipped to reflect on their subject of study (Communication) (Making judgements)

Course Content
Various subjects from the history of computing will be treated. Several highlights will be discussed and placed within the social context of its time. By discussing these highlights from several points of view the history of computing will serve as a way to illustrate the various roles of computing in society.

Additional Information Teaching Methods
Lectures and short assignments. Workgroup sessions. Optional quizzes helping to study; extracurricular activities for more in-depth info.
Method of Assessment
Written exam; assignments yield a bonus of 1 point if all three assignments are graded and the average score is 6 or better.

Literature
Thomas Haigh and Paul Ceruzzi, A new history of modern computing, MIT (2021)

Additional Information Target Audience
Bachelor Computer Science ( )
Additional Information
More information with the course coordinator: W&N room A652, d.j.beckers@vu.nl.

Explanation Canvas
All info about the course is available through Canvas. The "Modules" page offers a chronological overview of the lectures, indicating when to prepare what, and optional extras.

LINEAR ALGEBRA
Course Objective
All that's below falls under:

Knowledge and understanding
Applying knowledge and understanding
Making judgements
Communication
Lifelong learning skills (since Linear Algebra appears in all of science and industry)
Besides being able to explain, interrelate, know the basic properties of, and construct simple arguments with the concepts listed above, the student will learn the following skills (organized by topic):

Linear systems:

Can solve systems of linear equations using row-reduction
Can determine the number of solutions of a linear system
Can prove or disprove simple statements concerning linear systems
Linear transformations:

Can determine if a linear transformation is one-to-one and onto
Can compute the standard matrix of a linear transformation
Can use row-reduction to compute the inverse of a matrix
Can prove or disprove simple statements concerning linear transformations
Subspaces and bases:

Can compute bases for the row and column space of a matrix
Can compute the dimension and determine the basis of a subspace
Can prove or disprove simple statements concerning linear systems
Eigenvalues and eigenvectors:

Can compute the eigenvalues of a matrix using the characteristic equation
Can compute bases for the eigenspaces of a matrix
Can diagonalize a matrix
Can prove or disprove simple statements concerning eigenvalues and eigenvectors
Orthogonality:

Can compute the orthogonal projection onto a subspace
Can determine an orthonormal basis for a subspace using the Gramm-Schmidt algorithm
Can solve least-squares problems using an orthogonal projection
Can orthogonally diagonalize a symmetric matrix
Can compute a singular value decomposition of a matrix
Can prove or disprove simple statements concerning orthogonality
Course Content

The topics that will be treated are listed below. For every topic, the relevant concepts are listed.

Linear systems:

linear system (consistent/inconsistent/homogeneous/inhomogeneous), (augmented) coefficient matrix, row equivalence, pivot position/column, (reduced) echelon form, basic/free variable, spanning set, parametric vector form, linear (in)dependence.

Linear transformations:

linear transformation, (co)domain, range and image, standard matrix, one-to-one and onto, singularity, determinant, elementary matrices.

Subspaces and bases:

subspace, column and null space, basis, coordinate system, dimension, rank.

Eigenvalues and eigenvectors:

eigenvalue, eigenvector, eigenspace, characteristic equation/polynomial, algebraic multiplicity, similarity, diagonalization and diagonalizability.

Orthogonality:

dot product, norm, distance, orthogonality, orthogonal complement, orthogonal set/basis, orthogonal projection, orthonormality, orthonormal basis, Gramm-Schmidt process, least squares problem/solution, orthogonal diagonalization, singular value/vector, singular value decomposition, Moore-Penrose inverse.

Additional Information Teaching Methods
The course is spread over a period of seven weeks. Each week there will be two theoretical classes of 90 minutes each and two exercise classes of 90 minutes each.
Method of Assessment
There is a written exam at the end of the course.
Literature
Linear Algebra and its Applications, by David C. Lay, Steven R. Lay and Judi J. McDonald, global edition (sixth edition), Pearson or other material provided in the course.

Additional Information Target Audience
Bachelor Artificial Intelligence ( )
Bachelor Computer Science ( )

SOFTWARE DESIGN
Course Objective
The main objective of the course is to let you master model-based design methodologies and techniques, obtain insights and knowledge about recurrent software design problems and object orientation. (Knowledge and understanding)

In addition, you will develop critical reasoning skills: for a given software problem, you will be able to select and apply design principles and the most appropriate object-oriented design patterns. (Making judgements) (Applying knowledge and understanding) (Lifelong learning skills)

Course Content
Developing real software systems is complex; they are large, and their development often starts when their requirements are still not fully specified. The goal of software design is to model complexsoftware systems in a systematic manner. The lectures will cover and apply a number of software modeling techniques and principles. You will learn which technique is the most appropriate for which problem, how to describe a (software) problem using models, and how to use such models to reason about and communicate key aspects of your software.

The course also introduces several design patterns for creating robust, better-organized, and maintainable software systems. Design patterns can be considered as standardized blueprints for solving recurrent design problems regarding object-oriented software systems.

The course makes use of the Unified Modelling Language (UML).

Additional Information Teaching Methods
Lectures (H). Practical sessions (W).

Method of Assessment
In this course the assessment is composed of two components:

- Team project (70% of the final grade): it will be carried out throughout the whole course by groups of students; the required methods and principles for the project will be covered during the lectures and laboratory sessions throughout the whole duration of the course.

The result of the team project is composed of two parts: (i) a modelling part, and (ii) an implementation part.

Both the modeling and implementation parts will be evaluated by the teaching team according to a shared assessment rubric. Within the team project, each of you will be responsible for a certain part of the project; as a team, you will report the responsibilities that each team member took in the project.

- Written exam (30% of the final grade): it consists of a set of multiple-choice questions. The exam aims to assess your understanding and knowledge of the methods, principles, and practical insights discussed in the lectures.

Literature
Martina Seidl, Marion Scholz, Christian Huemer, Gerti Kappel, "UML@Classroom: An Introduction to Object-Oriented Modeling", 2015.
John Ousterhout, “A philosophy of software design”, Yaknyam Press, 2018.
Martin P. Robillard, “Introduction to Software Design with Java”, Springer, 2019.
Additional Information Target Audience
Bachelor Computer Science ( )
Recommended background knowledge
Object-oriented principles in any programming language (for instance in Scala, Java, C/C++, Python).

DATABASES
Course Objective
The students will learn:

to design and evaluate database schemas using Entity-Relationship diagrams, (Applying knowledge and understanding) (Making judgements)
to understand the relational model (including integrity constraints such as key constraints and foreign key constraints), (Knowledge and understanding)
to translate Entity-Relationship diagrams into relational schemas (and to enforce cardinality constraints by design), (Applying knowledge and understanding)
to reason about good and bad database design using functional dependencies, (Making judgements)
to apply normalization algorithms to transform a database schema into (various) normal forms, (Applying knowledge and understanding)
to understand concurrency concepts like transactions, schedules and concurrency anomalies, and (Knowledge and understanding)
to reason about different techniques for concurrency control (lock based concurrency control, optimistic concurrency control, multiversion concurrency control). (Making judgements)
Course Content
The course is concerned with base principles and important aspects of relational databases. The course objective is to obtain a good knowledge and understanding of relational database systems. This includes the ability to develop good database models, and to query and update databases using SQL.

Additional Information Teaching Methods
Lectures, exercise/practicum classes and individual homework.
Method of Assessment
The homework is mandatory for qualifying for the exam (70% of the homework points to qualify for the exam). In the case that at least 90% of the homework points are obtained, 0.5 bonus points are awarded for the final grade. At the end of the course, there is a final exam.
The overall grade is the grade of the final exam plus the possibly 0.5 bonus point obtained for the homework. (The bonus is only added for students who pass the exam with a grade of at least 5.5.)

There is no resit opportunity for the homework.

Literature
Database Systems, The Complete Book, by: Hector Garcia-Molina & Jeffrey D. Ullman & Jennifer Widom. 2nd edition.

Additional Information Target Audience
Bachelor Artificial Intelligence ( )
Bachelor Business Analytics ( )
Bachelor Computer Science ( )

Recommended background knowledge
Basic programming skills help.

ACADEMIC WRITING FOR CS

Course Objective

The aim of this course is to improve your academic writing skills. More specifically, you will:

gain insight into the specific characteristics of formal written English (knowledge and understanding);
learn the conventions of using sources through quoting, referring and paraphrasing; (Knowledge and understanding) (Communication) (Lifelong learning skills)
develop your academic skills with regard to text structure, linguistic accuracy, and clarity of expression. (Knowledge and understanding) (Communication) (Lifelong learning skills)
In addition, you will be able to apply this knowledge by writing a text about an academic topic, which uses academic sources appropriately; has no major grammatical errors; uses a clear text structure; and has a style which is suitable for academic texts. (Applying knowledge and understanding) (Communication) (Making judgements) (Lifelong learning skills)

Course Content

In the course, we will focus on generic skills that are relevant for academic writing: creating an appropriate formal style and using standard academic phrases and creating a clear structure for your text. You will write a text about a topic from your discipline, based on academic sources, using a referencing style that is standard in your discipline. We will also pay attention to general principles of good writing, such as guiding the reader through the text and creating a good flow in the text. We will address these topics in such a way that you will be able to apply these principles in any text that you write after following the course.

In addition, you will get individual feedback on your own texts, which will allow you to gain insight into your own strengths and weaknesses in terms of linguistic accuracy (grammar, punctuation, spelling and vocabulary).

Additional Information Teaching Methods

Lectures and seminars.

Method of Assessment

Several short assignments (required; pass/fail) and a final paper (required; pass/fail).

Additional Information Target Audience

Bachelor Computer Science ( )

Additional Information

This course has obligatory attendance (80% of all classes).

Explanation Canvas

Course and reading materials will be made available via Canvas.

PROFESSIONAL DEVELOPMENT
Course Objective
Upon completion of this course, students will:

develop an understanding of a number of different career paths that are available to them in the education sector (knowledge and understanding)
develop an understanding of a number of different career paths that are available to them in the industry (knowledge and understanding)
be able to make informed choices when taking the first steps toward the job market (apply knowledge and understanding) (critical thinking) (making judgements)
learn and be able to apply the main features of technical interviewing (knowledge and understanding) (apply knowledge and understanding) (making judgements) (Communication) (learning skills)
be aware of the relevance of ethics, diversity, and inclusion within the professional world of Computer Science (apply knowledge and understanding) (critical thinking) (making judgements)
Course Content
During this course, students will be exposed to several guest lectures from staff members and guests from the industry who will talk about topics relevant to students trying to get a first impression of their options in the labor market. These topics include an overview of possible career paths in education with a degree in Computer Science, including pursuing a Ph.D. and an academic career. Further, there will be guests from the industry, who will talk about internships and different career paths. Technical interviewing will be covered as well. In addition, the course will cover the topics of ethics, diversity, and inclusion in the professional field of Computer Science.

Additional Information Teaching Methods
Guest lectures.

Method of Assessment
Individual (pass/fail) assignments.

Literature
The course materials will be posted on Canvas.

Additional Information Target Audience
This is a constrained-choice course for the students enrolled in   of the Bachelor Computer Science.

INFORMATION MANAGEMENT FOR CS
Course Objective
Through this course, students will study the topic of information management in breadth and in depth. After taking this course, the student will be able to:

Explain the strategic relevance of information systems for modern organizations; (Knowledge and understanding) (Applying knowledge and understanding)
Analyze the level of competition within an industry and determine is implications for an information systems strategy; (Applying knowledge and understanding)
Understand the impact of the advent of the internet on the management of information within and between organizations; (Knowledge and understanding)
Identify the various phases in the development of an information system; (Applying knowledge and understanding)
Model simple and moderately complex business processes with a formal modeling technique (Petri nets); (Making judgements) (Applying knowledge and understanding)
Analyze process models with respect to various behavioral properties. (Lifelong learning skills)
Course Content
No organization can do without information systems. For some organizations, such systems are even of strategic relevance, as they offer a clear competitive advantage. Think, for example, of how Amazon has become such a dominant retailer or how an organization like Uber has conquered the taxi market.

This course explains the relevance and use of information systems in modern organizations. We will briefly sketch how the role of information systems has developed over the years to reach its current ubiquitous level. Special attention is devoted to the rise of the internet and its impact on traditional organizations, as well as the emergence of new types of (cloud-based) organizations.

Reasoning from the organizational importance of information systems, we will look into the way information systems are developed such that organizations can achieve their objectives. We will pay considerable attention to an important phase in information system development, namely how we analyze and model business processes. For this purpose, we will rely on the use of classical Petri nets.

This course will approach the topic of information management in breadth and in depth. Breadth is achieved by giving an overview of all relevant topics in the area of information management; depth is attained by introducing students to a powerful, formal modeling technique that they will learn to master in the context of organizational analysis.

Additional Information Teaching Methods
This course consists of lectures and practicals. Attendance is not mandatory but highly encouraged.
Method of Assessment
Intermediate exam (individual, 30%) and final project (group-based, 70%).

Resit option: resit for intermediate exam, re-submission of the final project.

Literature
"Business Information Management: Improving Performance using Information Systems", by Dave Chaffey and Steve Wood. ISBN: 9780273686552.
"Modeling Business Processes", Wil van der Aalst and Christian Stahl. ISBN: 9780262015387 (print), 9780262296465 (eBook).
Additional Information Target Audience
Bachelor Computer Science ( )

LOGIC AND MODELLING
Course Objective
Students will learn to:

Describe fundamental principles of user experience and human factors (Knowledge and understanding)
Describe fundamental input and output techniques for human-computer interaction (Knowledge and understanding)
Gather requirements for designing interactive systems (Applying knowledge and understanding)
Apply prototyping methods for interactive systems (Applying knowledge and understanding)
Evaluate interactive systems (Making judgements)
Validate empirical evaluations of interactive systems (Making judgements, Applying knowledge and understanding)
Communicate design solutions and evaluations on an academic level (Communication)
Work in teams to achieve a common goal (Communication) (Lifelong learning skills)
Course Content
The lectures in this course will discuss and present examples of concepts and methods in the field of human-computer interaction. It will cover in detail all steps in the human-centered design lifecycle; We will discuss methods for requirement analysis, designing, and building prototypes of interactive systems, and evaluating them. In practical sessions, students will practice the use of relevant methods within the context. Some of the topics covered in the course are: User Needs Analysis, Conceptual Design, Mockups, and Prototypes, Usability, Evaluation of Prototypes and technology for interactive systems.

Additional Information Teaching Methods
Lectures, practical sessions.
Method of Assessment
The final grade is determined by the partial grades of the following deliverables:

course assignments (group and possibly individual);
exam (individual).
The final grade will be calculated based on the assignment grades and the exam grade (FINAL GRADE = 0.6 ASSIGNMENTS + 0.4 EXAM). To pass the course, students need to complete all assignments and pass both parts (Grade >= 5.5).

Literature
Recommended Literature:

MacKenzie, I. S. (2012). Human-computer interaction. Elsevier Science.

ISBN: 9780124058651. Available for free online from the VU Library for registered VU students (https://vu.on.worldcat.org/oclc/875263048)

Additional Information Target Audience

Bachelor Computer Science ( )
Explanation Canvas
All details of the course will be published on the Canvas page.
Recommended background knowledge
Basic programming skills, basics statistics, data collection.

HUMAN-COMPUTER INTERACTION
Course Objective
This course is an introduction to security and safety engineering for Bachelor Students in Computer Science to

build awareness of security and safety issues in software systems,
apply security and safety threat analysis and mitigation techniques at conceptual level to practical case studies, and
inspire students to further their education in computer security by exposing them to industrial practices.
The course is organised along the principles of scientific peer reviews and expects motivated students to reach the highest educational objective in Bloom's cognitive taxonomy (Judgement).

Course Content
After completing this course, the student will be able to:

Apply the conceptual elements of security and safety engineering for security and safety risk analysis (assets, threats, risks and controls) on a concrete problem [Applying knowledge and understanding, Communication]
Identify and review pros and cons of qualitative techniques on the concrete problem instance by reviewing the work of their peers. [Making judgements, Communication, Lifelong learning skills]
Apply the general methodology so learned to the industrial software vulnerability assessment techniques on a concrete problem. [Applying knowledge and understanding, Lifelong learning skills]
Additional Information Teaching Methods
The course is organised into weekly lectures coupled with practical assignments where the students will apply gained theoretical knowledge on one or more case studies of a software system.

Lectures and discussion/Q&A sessions. These activities will cover risk analysis, authentication and identification, assets, threats and security controls, software vulnerability and IT system assessment and introduce the concepts the students will be asked to apply.
All questions will be answered in the Q&A sessions
Security Risk Analysis report(s). Each student will write a risk analysis report that is submitted in incremental stages with weekly deadlines including assets, threat analysis and security controls.
Each report includes a new part corresponding to the newly introduced material and a rebuttal of the received reviews on the previous part as in scientific articles
Students who do not submit by the deadline will not be able to review the reports of their peers.
Individual peer reviews. Each student will review the assignments submitted by their peers according a detailed grading scheme provided and exemplified by the lecturers
Each student will have to review five assignments for each type of assignment.
The feedback of students of past editions of the course is that the average time for peer evaluation is estimated at one hour per (good) review.

Class presentation and public review. To provide intermediate feedback and to learn to identify and review a variety of applications of the technique students will be asked to discuss their assignments in front of the class.
Attending the discussion classes is mandatory, also for students that have already discussed in previous sessions.
Method of Assessment
Given the tight timeline, at the beginning of the course students will have to submit a short (1/3 page) essay on their study strategy and grade expectation. Those that do not submit by the deadline will get a NS "No Show" as the final grade.

The overall grade is determinbed as follows:

60% for the reportseach of which is divided as
65% on the evaluation of the report as graded by the peers
35% for the participation to the peer review process and the quality of the reviews as re-evaluated by the lecturers and teaching assistants
40% for the final exam for the score of vulnerability assessments
Pass or fail

Quality of the reviews. Additional points beyond the participation will be deducted to students who write poor (e.g. generic such as "it is ok, full points") or unfair reviews.
Presentation of the reports in class. If a student is asked to present his/her report in class and is absent or not able to do present it, the assignments will be considered void.
Students must reach 5.5 on both the report and the final exam.

Plagiarism and Fraud Check

If the number of students is so large that discussions in front of the class cannot be organised for all students to present at least once, a digital exam will be organised to assess the knowledge of the reviewed material. Students who have not presented must score at least 6/10in the confirmation exam for the assignments score to be considered valid.

Resit

There is a resit for the final exam on vulnerability assessment.

It is not possible to resit the assignments and the peer reviews. Students can only resit failed reviews due to additionally deducted points by writing a report analyzing the reviews of all other students for each failed report.

Entry Requirements
While there is no formal entry requirements, students who do not have the background knowledge provided by the courses on Computer Networks, Operating Systems, Web/Application Software development will encounter some difficulties in the course.

Literature
Gibson. Managing Risk in Information Systems. Jones & Bartlett
This book offers a general structure of the security assessment process in industry and can be followed for the high level process of threat and security
Additional Lecture Material
Slides and Additional Material will be provided in Canvas
Common Vulnerability Scoring System Standard v3.1 and v4.0
National institute of Standards and Technology - 800-* family
SESAR OpenSky Security Risk Assessment Method SecRAM simplified version used by Eurocontrol
Anderson. Security Engineering. Wiley.
The previous version of the book is available on line https://www.cl.cam.ac.uk/~rja14/book.html
Shostack, A. (2014). Threat modelling: Designing for security. Wiley
Selected chapters will be made available on canvas for educational use.
Additional Information Target Audience
Computer Science Bachelor ( )
Additional Information
Please see the information on Canvas.

Custom Course Registration
Given the tight timeline, at the very beginning of the course students will have to submit a short (1/3 page) essay on their study strategy and grade expectations. Those that do not submit by the deadline will get a NS "No Show" as the final grade.

Explanation Canvas
The submission of the report and the peer review of the reports will be done in Canvas/FeedbackFruit.

Because of the way the system works by automatically assigning reviewers, it is not possible to grant any deadline extension for the submission of the reports and the submission of the reviews.

Students should therefore carefully plan their activities and submit also a draft version of their reports well in advance of the deadline (multiple submissions are possible) rather than waiting for the last moment.

Recommended background knowledge
While there is no formal entry requirements, students who do not have the background knowledge provided by the courses on Computer Networks, Operating Systems, Web/Application Software development will encounter some difficulties in the course.

SECURITY AND SAFETY ENGINEERING
Course Objective
This course is an introduction to security and safety engineering for Bachelor Students in Computer Science to

build awareness of security and safety issues in software systems,
apply security and safety threat analysis and mitigation techniques at conceptual level to practical case studies, and
inspire students to further their education in computer security by exposing them to industrial practices.
The course is organised along the principles of scientific peer reviews and expects motivated students to reach the highest educational objective in Bloom's cognitive taxonomy (Judgement).

Course Content
After completing this course, the student will be able to:

Apply the conceptual elements of security and safety engineering for security and safety risk analysis (assets, threats, risks and controls) on a concrete problem [Applying knowledge and understanding, Communication]
Identify and review pros and cons of qualitative techniques on the concrete problem instance by reviewing the work of their peers. [Making judgements, Communication, Lifelong learning skills]
Apply the general methodology so learned to the industrial software vulnerability assessment techniques on a concrete problem. [Applying knowledge and understanding, Lifelong learning skills]
Additional Information Teaching Methods
The course is organised into weekly lectures coupled with practical assignments where the students will apply gained theoretical knowledge on one or more case studies of a software system.

Lectures and discussion/Q&A sessions. These activities will cover risk analysis, authentication and identification, assets, threats and security controls, software vulnerability and IT system assessment and introduce the concepts the students will be asked to apply.
All questions will be answered in the Q&A sessions
Security Risk Analysis report(s). Each student will write a risk analysis report that is submitted in incremental stages with weekly deadlines including assets, threat analysis and security controls.
Each report includes a new part corresponding to the newly introduced material and a rebuttal of the received reviews on the previous part as in scientific articles
Students who do not submit by the deadline will not be able to review the reports of their peers.
Individual peer reviews. Each student will review the assignments submitted by their peers according a detailed grading scheme provided and exemplified by the lecturers
Each student will have to review five assignments for each type of assignment.
The feedback of students of past editions of the course is that the average time for peer evaluation is estimated at one hour per (good) review.

Class presentation and public review. To provide intermediate feedback and to learn to identify and review a variety of applications of the technique students will be asked to discuss their assignments in front of the class.
Attending the discussion classes is mandatory, also for students that have already discussed in previous sessions.
Method of Assessment
Given the tight timeline, at the beginning of the course students will have to submit a short (1/3 page) essay on their study strategy and grade expectation. Those that do not submit by the deadline will get a NS "No Show" as the final grade.

The overall grade is determinbed as follows:

60% for the reportseach of which is divided as
65% on the evaluation of the report as graded by the peers
35% for the participation to the peer review process and the quality of the reviews as re-evaluated by the lecturers and teaching assistants
40% for the final exam for the score of vulnerability assessments
Pass or fail

Quality of the reviews. Additional points beyond the participation will be deducted to students who write poor (e.g. generic such as "it is ok, full points") or unfair reviews.
Presentation of the reports in class. If a student is asked to present his/her report in class and is absent or not able to do present it, the assignments will be considered void.
Students must reach 5.5 on both the report and the final exam.

Plagiarism and Fraud Check

If the number of students is so large that discussions in front of the class cannot be organised for all students to present at least once, a digital exam will be organised to assess the knowledge of the reviewed material. Students who have not presented must score at least 6/10in the confirmation exam for the assignments score to be considered valid.

Resit

There is a resit for the final exam on vulnerability assessment.

It is not possible to resit the assignments and the peer reviews. Students can only resit failed reviews due to additionally deducted points by writing a report analyzing the reviews of all other students for each failed report.

Entry Requirements
While there is no formal entry requirements, students who do not have the background knowledge provided by the courses on Computer Networks, Operating Systems, Web/Application Software development will encounter some difficulties in the course.

Literature
Gibson. Managing Risk in Information Systems. Jones & Bartlett
This book offers a general structure of the security assessment process in industry and can be followed for the high level process of threat and security
Additional Lecture Material
Slides and Additional Material will be provided in Canvas
Common Vulnerability Scoring System Standard v3.1 and v4.0
National institute of Standards and Technology - 800-* family
SESAR OpenSky Security Risk Assessment Method SecRAM simplified version used by Eurocontrol
Anderson. Security Engineering. Wiley.
The previous version of the book is available on line https://www.cl.cam.ac.uk/~rja14/book.html
Shostack, A. (2014). Threat modelling: Designing for security. Wiley
Selected chapters will be made available on canvas for educational use.
Additional Information Target Audience
Computer Science Bachelor ( )
Additional Information
Please see the information on Canvas.

Custom Course Registration
Given the tight timeline, at the very beginning of the course students will have to submit a short (1/3 page) essay on their study strategy and grade expectations. Those that do not submit by the deadline will get a NS "No Show" as the final grade.

Explanation Canvas
The submission of the report and the peer review of the reports will be done in Canvas/FeedbackFruit.

Because of the way the system works by automatically assigning reviewers, it is not possible to grant any deadline extension for the submission of the reports and the submission of the reviews.

Students should therefore carefully plan their activities and submit also a draft version of their reports well in advance of the deadline (multiple submissions are possible) rather than waiting for the last moment.

Recommended background knowledge
While there is no formal entry requirements, students who do not have the background knowledge provided by the courses on Computer Networks, Operating Systems, Web/Application Software development will encounter some difficulties in the course.

SOFTWARE ENGINEERING PROCESSES

Course Objective

The main goal of the course is to provide students with theory and practice on concepts, processes, and tools related to software development and maintenance. These include traditional software quality factors, software testing techniques, and modern software engineering processes (i.e., collaborative development and DevOps).

Upon completion of the course, students will be able to:

Understand and explain software quality factors (Knowledge and understanding);
Understand, explain, and apply basic software testing techniques (Knowledge and understanding) (Applying knowledge and understanding);
Understand and explain the fundamentals of DevOps and how DevOps teams can build and deliver software with the help of existing tools (Knowledge and understanding);
Implement a Continuous Integration/Deployment pipeline (Applying knowledge and understanding).

Course Content

Developing real software systems is complex. They have various dimensions (e.g., they are usually large), they involve several people (having different backgrounds) collaborating and synchronizing with each other, they need various resources of different types, and their development often starts when the customer requirements are still unclear. The main steps of a software engineering process are analysis of requirements, software design, implementation, testing, and maintenance. While other courses cover early activities in a software engineering process (Requirements Engineering and Software Design), this course covers the continuous activities performed by software engineers to develop, maintain, and operate software systems.

The topics covered by the course are:

Introduction to software quality
Introduction to software testing
- DevOps

Continuous Integration (CI) and Continuous Deployment (CD)
Collaborative development (e.g., version control systems, code review)
Open-source software development

Additional Information Teaching Methods

The course contains lectures about the listed topics above. In addition, there will be at least one guest lecturer from the industry who will bring their case to show how software development and maintenance happens in practice.

Method of Assessment

The grade will be based on assignment(s) (group-based, 80%) and a digital exam (multiple-choice questions, 20%).

Resit option: re-submission of a simplified version of the assignment(s) and re-take of the digital exam.

To pass the course, it's mandatory to have a passing grade for each assignment and a passing grade for the digital exam.

Literature
The course material will be handed out by the lecturers and distributed online (Canvas). It is composed of a selection of book chapters, relevant scientific articles, and documentation of the used software/tools.

Additional Information Target Audience
Bachelor Computer Science ( ).

Explanation Canvas
Further information for this course will be made available online (Canvas). All students must be enrolled in the course on Canvas. Group enrollment takes lace in Canvas.

Recommended background knowledge
Requirements Engineering, Software Design.

BACHELOR PROJECT COMPUTER SCIENCE
Course Objective
Upon successful completion of the Bachelor Project Computer Science, students will be able to:

Describe an open problem in the domain of computer science;
Search and identify scientific literature relevant to the chosen problem;
Design a scientific and systematic approach to solving the chosen problem;
Execute a project by following the approach, which may involve designing and conducting experiments, developing tools or techniques, collecting and analyzing data;
Clearly and concisely present the contributions of the project as well as demonstrate the validity, effectiveness, and limitations of the chosen approach;
Report the findings in writing in the form of a coherent story to a general CS audience;
Present their project (i.e., the motivation for the study, the methodology, the findings, and conclusions) to a general CS audience.
Each learning outcome of the bachelor Project Computer Science covers all the Dublin Descriptors (Knowledge and Understanding, Applying knowledge and understanding, Making judgements, Communication, Lifelong learning skills).

Course Content
The CS Bachelor Project marks the culmination of the Bachelor's programme in Computer Science. During this project, you will tackle a clear and focused question within your area of interest in computer science. The skills and knowledge you have gained throughout your studies and the experience from this project will equip you for further graduate studies or a career in the industry.

The CS Bachelor project can be either purely theoretical or applied (possibly involving practical implementation). Regardless of its nature, the project must adhere to VU’s scientific standards. The scientific work you perform, along with related tasks such as identifying and reviewing relevant prior research, evaluating findings, and presenting your work in the thesis, are all critically important.

Additional Information Teaching Methods
The CS Bachelor Project officially begins in period 5 and spans periods 5 and 6. Prior to the start, students must find a project and a supervisor. A list of available project topics is provided via Canvas in periods 2 or 3. Students should contact the supervisors who proposed the topics and, after an orientation meeting, confirm their collaboration. Students needing assistance in finding a suitable supervisor can reach out to the course coordinator. Students may also choose to do their project in semester 1. To find a project and a supervisor, and plan the following steps, they need to contact the course coordinator.

Students define the exact nature and scope of their CS Bachelor Project in collaboration with their supervisor. They can choose a topic from the list available via Canvas, or propose a topic that aligns with the research interests of a potential supervisor. Another option is to find an internship where they can carry out relevant research. Notice that it is necessary to seek the approval of a VU staff member who can act as the examiner before starting any internship. The course coordinator can advise students on this matter.

Projects may be carried out individually or in a group, where each student has a specific role. Supervision meetings may also be carried out in a group fashion.

Method of Assessment
The project work results in a written report (thesis) and an oral presentation towards the end of period 6, which are assessed individually by two examiners.

The final grade is awarded by the first and second examiners as follows: 50% for the technical contributions, 40% for the written report (i.e., thesis), and 10% for the oral presentation. A detailed assessment rubric is provided via Canvas.

The deadlines for submitting the final version of the thesis and doing the presentation are planned by the students in agreement with the supervisor. For students who wish to graduate in August, the supervisor must submit their graded thesis and final assessment form by July 31.

For students who wish to graduate in February, the supervisor must submit their graded thesis and final assessment form by January 31. More information is provided on Canvas.

Entry Requirements
Before starting the Bachelor Project Computer Science (XB_40001), students must have completed at least 120 EC of the Computer Science Bachelor's programme.

Additional Information Target Audience
Bachelor Computer Science ( )

Explanation Canvas
Detailed information about the Bachelor Project is offered on its Canvas page. The Canvas page is published in period 1.

AUTOMATA AND COMPLEXITY

Course Objective

The first part of the course, on automata and languages, deals with the. concepts of formal language, grammar, and automaton. Two types of languages are covered: regular and context-free languages. Regular. languages in the form of regular expressions are ubiquitous in computer science. They are used in search queries and text manipulation.

Context-free languages and grammars (e.g. the Backus–Naur form) are the prevailing standards for describing programming languages. We also discuss parsing algorithms to determine whether a given string is in a context-free language. The automata-theoretic counterparts for regular and context-free languages are finite automata and the more powerful pushdown automata.

The students will learn to:

design finite automata and create regular expressions for a given regular language, [Knowledge and understanding] [Applying knowledge and understanding]
apply algorithms to translate between finite automata, right-linear grammars, and regular expressions, [Knowledge and understanding] [Applying knowledge and understanding]
apply algorithms to make automata deterministic and minimal, [Knowledge and understanding] [Applying knowledge and understanding]
design pushdown automata and create context-free grammars for a given context-free language, [Knowledge and understanding] [Applying knowledge and understanding]
apply algorithms to translate between pushdown automata and context-free grammars, [Knowledge and understanding] [Applying knowledge and understanding]
use pumping lemmas to reason about whether a language is regular or context-free, [Applying knowledge and understanding] [Making judgements]
apply algorithms for parsing context-free languages. [Knowledge and understanding] [Applying knowledge and understanding]
In the second part of the course, on computability theory, the central question is: what computations can be performed on a computer? To reason about computability, we introduce the mathematical model of Turing machines and discuss the Church-Turing thesis. We discuss examples of undecidable problems: the halting problem and the Post correspondence problem. It is shown how the undecidability of new problems can be shown by reduction from a known undecidable problem. Finally, we discuss a classification of decidable/computable problems into important complexity classes, notably P, NP, and NP-complete problems, together with the corresponding reduction arguments.

The students will learn to:

reason whether a given problem is decidable (computable) or undecidable (not computable), [Making judgements]
understand the classification of decidable problems in the complexity hierarchy (e.g. P, NP, EXP) of, and [Knowledge and understanding]

reason about the complexity of a problem via the reduction to/from problems with known complexity. [Making judgements] [Communication] [Lifelong learning skills]
Course Content
This course treats automata & formal languages and computability theory.

The student gets acquainted with important notions and algorithms regarding formal languages, automata, grammars, compilers, computability, and complexity.

This course addresses foundational questions in computer science:

What can be computed? What are the limitations to what computers can do?
How much time and memory does solving a problem require?
What is a (programming) language?
How can languages be recognized by computers (automata)?
Which problems can be solved by what kinds of automata?
This course conveys the important idea that certain problems cannot be solved by computers. A computer scientist must be able to reason whether a given problem is computable (decidable) or not. Moreover, a computer scientist should be able to reason what language/complexity class a given problem belongs to, and hence what kind of automata/algorithms are needed to tackle the problem.

Additional Information Teaching Methods
4 hours per week lectures; 4 hours per week exercise classes.
Method of Assessment
The homework is mandatory for qualifying for the exam (70% of the homework points to qualify for the exam). In case at least 90% of the homework points is obtained, 0.5 bonus point is awarded for the final grade. At the end of the course there is a final exam.

The overall grade is the grade of the final exam plus the possibly 0.5 bonus point obtained for the homework. (The bonus is only added for students that pass the exam with a grade of at least 5.5.)

There is no resit opportunity for the homework.

Literature
Peter Linz, An Introduction to Formal Languages and Automata, Jones & Bartlett, 4th or 5th edition

Additional Information Target Audience
BSc Computer Science ( )

MACHINE LEARNING
Course Objective
Upon completion of this course, students will:

be acquainted with the dominant concepts of machine learning methods, including some theoretical background. (Knowledge and understanding)
acquire knowledge of established machine learning techniques such as linear models, neural networks, decision trees and ensemble methods (Knowledge and understanding)
learn some statistical techniques to assess and validate machine learning results. (Apply knowledge and understanding, make judgments)
Course Content
Machine learning is the discipline that studies how to build computer programs that learn how to behave from examples, rather than following explicit instructions. It is a subfield of artificial intelligence that intersects with statistics, cognitive science, information theory, and probability theory.

Recently, machine learning has become increasingly important for the design of search engines, robots, and sensor systems, and for the processing of large scientific data sets. Other applications include handwriting or speech recognition, image classification, medical diagnosis, stock market analysis and bioinformatics.

The course covers a wide variety of machine learning techniques, but puts particular emphasis on gradient descent optimization, backpropagation, neural networks and deep learning. Some discussion on the broader social impact of machine learning technology is included.

Additional Information Teaching Methods
The course consists of pre-recorded videos, interactive lecture/QA sessions (two per week) and optional homework assignments discussed in working groups (one per week). The practical assignment is supported by small exercises to help with the relevant technologies, and informal presentations at project groups (one per week).

There is no mandatory attendance for any lectures or workgroups, except that one member of each group must be present at the weekly project group. A large amount of the material is freely available at https://mlvu.github.io The course is taught in English.

Method of Assessment
The course assessment consists of two parts: an examination and a practical assignment. The examination consists of a standard exam and four online quizzes. The examination and practical assignment both comprise 50% of the final grade.
To pass the course, the examination grade should be at least 5.5, the practical assignment grade should be at least 4.5 and the average should be at least 5.5.

The examination is made individually, and the practical assignment is made in groups. There is a resit for the exam, no resit is possible for the practical assignment.

Literature
There is no textbook. Some reading material will be provided digitally.
Additional Information Target Audience
This course is only open to the following programs. If you are not a student of these programs, registration is not available.

It is currently part of the curriculumn of:
Bachelor Artificial Intelligence
Bachelor Business Analytics
Bachelor Computer Science

Premaster Artificial Intelligence
Premaster Business Analytics

It is available as an elective to students of:
Bachelor Mathematics
Bachelor Medische Natuurwetenschappen

If you are in the MSc Bioinformatics, you need to register forScientific Machine Learning (XM_0138), which shares some aspects with this course.

Custom Course Registration
Workgroup registration will be done through Canvas in the first week of the course. Only a registration for the main course itself is required to take part in all parts of the course.

Recommended background knowledge
We require that students have some prior experience with linear algebra, calculus (limited to differentiation), probability theory and statistics. An overview and explanation of the required preliminaries can be found at https://mlvu.github.io/preliminaries/

A basic understanding will suffice and we will take some time to go over the basics again. Feel free to register if you have no experience with any of these, but expect to put in a little extra effort in the first weeks (using the materials provided).

Programming experience in python is also required.

PHILOSOPHY AND ETHICS
Course Objective
At the end of the course, the students will be:

familiar with basic knowledge of key theories in Moral philosophy, Philosophy of Mind and AI, and Epistemology [knowledge and understanding];
able to apply these theories to moral dilemmas concerning new technologies [applying knowledge and understanding];
trained to deal with moral dilemmas they might encounter in their future career [learning skills].
Course Content
Argument of the course:

New technologies (e.g. the ads we see, nudging us to make healthy and sustainable choices, writing texts, etc.) that influence our actions and ways of thinking are everywhere.
It is better if these technologies meet certain moral and epistemic standards.
Hence, those involved in the design or regulation of these technologies (i.e. you, in the future) should know about the main ideas in ethics, epistemology, and philosophy of mind/AI.
This course surveys and critically examines relevant issues in Philosophy and Ethics, including the philosophy of AI, algorithms and echo chambers, and the background moral and epistemological theories that help us to think critically about new technologies. You will learn about the main ethical theories (namely, utilitarianism, Kantian ethics, and theories of well-being), epistemological theories (evidentialism; epistemic paternalism), and in philosophy of mind/AI (functionalism; mind-brain identity theory; the ethics of AI), learning to utilize these theories and distinctions to resolve moral and social dilemmas in a justified and intelligent way.

Additional Information Teaching Methods
Lectures

Method of Assessment
Final exam (100% grade)

Literature
Selected textbook chapters:

All texts will be available online.

- Chapters from Rachels & Rachels The Elements of Moral Philosophy.
- Chapters from The Oxford Handbook of Ethics of AI.

Selected academic articles or chapters:
- Nguyen, C. Thi (2021). Echo chambers and epistemic bubbles. Episteme.
- Rubel, Alan; Clinton, Castro, and Pham, Adam (2021). “Agency Laundering and Information Technologies” Algorithms and Autonomy: The Ethics of Automated Decision Systems. Cambridge University Press.