COURSE DESCRIPTIONS FOR MASTER COURSES AT REYKJAVÍK UNIVERSITY

Table of Contents

T-403-ADGE Operation Research

Credits: 6 ECTS
Year: one
Semester: fall term
Type of course: advanced elective undergraduate course for MSc in Computer Science.
Necessary Prerequisites: T-101-STA1, Calculus 1, T-211-LINA, Linear Algebra, T-302-TOLF, Statistics I
Organization of course: twelve-week course
Teacher: Heiðar Snær Jónasson and Páll Jensson
Language of teaching: Icelandic

Description

Introduction to standard Operations Research methods. Linear programming and sensitivity analysis, integer programming, dynamic programming, queuing theory, scheduling, networks. Using software for modelling and optimisation. The course supervisor is Páll Jensson.
Taught mostly in Icelandic. Non-Icelandic speaking students will receive course material and tutorial sessions in English.

Learning outcomes:

After the completion of this course students will be capable of using basic methods of Operations Research for analysing and solving complex decision problems. More specifically the student will be capable of:

- Using standardized processes to work on complex decision problems
- Applying systematic methods and algorithms for analysing and solving decision problems
- Understand how to use data and quantitative methods for decision making
- Understand the importance and usefulness of linear optimization and its applications
- Applying commercial software to solve optimization models with particular emphasis on MS Excel and MPL
- Solving optimization models with Simplex method
- Understand the use of sensitivity analysis
- Understand integer programming and how it can used in decision making
- Identify traditional transportation and distribution problems and be able to solve problems with the relevant methods
- Understand the special properties of network model
- Formulate and solve network models from practical problems
- Apply methods from decision science to solve simple practical problems
- Present results in a clear and organized manner

Assessment:

Winter grade (two tests will be conducted over the winter and the better one will count for 50% of final grade) 50%
Final exam: 50% (to earn the right to take the final exam the student needs to take part in exercise classes).

Reading Material:

Introduction to Operations Research, Hillier, Fredericks S. & Lieberman, Gerald J. Mc Graw Hill Education, 10th ed., (2014).

T- 414-AFLV Effective Programming and Problem solving

Credits: 6 ECTS
Year: one
Semester: spring
Type of course: advanced elective undergraduate course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-111-PROG Programming, T-110-VERK, Problem Solving, T-301-REIR, Algorithms
Organization of course: three-week course
Teacher: Arnar Bjarni Arnarson and Baldur Óli Barkarson
Language of teaching: Icelandic

Description

Computer scientists often have to deal with challenging tasks requiring both fast algorithmically linear solutions and efficient coding. This is one of the reasons why programming puzzles and oral exams are used so often in job interviews when applying to the strongest companies or graduate schools. The goal of this course is to make students better in solving algorithm tasks and acquire skills in a fun competition environment. The training exercises include challenges from international competitions, such as the ICPC and ToCoder. Other main tasks is to make decisions under strict time limits. Training will also be done in cooperation and dialogue, and by utilizing scarce resources (e.g. one computer for each team with a limited time). The course is intended to be informative, but at the same time fun. The material that will be covered includes much of the material in the algorithm's courses (e.g., data structures, dynamic programming, network search, and share-and-rule), but the emphasis will be different: how we perceive what solution method is applicable, the choice of design decisions when project is brought into the framework of solution method, and how this is implemented in the code. Students will tackle with applying and refining the core methods of transferring demonstration solutions into a programming solution.

Learning outcomes

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge

- Be able to describe algorithms, data structures and projects in a clear manner.

Skills

- Be able to develop the correct implementation of a well-defined algorithm or data structure.
- Be able to compare the difficulty of different tasks.
- Be able to report on the effectiveness of different solution approach for the given task to determine which methods are effective enough for the given conditions.
- Be able to apply various types of algorithms, such as greedy methods, dynamic programming, share-and-rule and Heuristic to solve given tasks.
- Be able to communicate and work in a group setting to solve problems under time pressure.

Competences

- Be able to develop solutions to projects that have not been seen before.

Assessment:

Problem sets	70%
Problem sessions	10%
Final exam	20%

Reading Material:

Lecture notes provided by teacher.

T-417-TOOR Computer Security

Credits: 6 ECTS
Year: one
Semester: fall
Type of course: elective advanced undergraduate course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-215-STY1, Operating systems, T-409-TSAM, Computer networks
Organization of course: three-week course
Teacher: Alex Már Gunnarsson and Níels Ingi Jónasson
Language of teaching: Icelandic

Description:

This course covers the section of information security that covers software and hardware and their use. We will dive into common vulnerabilities in software and web services, how attackers exploit them, and how it is possible to defend systems against such attacks. We will also cover network security, and many other attacks used by hackers today. The goal of this course is for students to gain a deep understanding of the core fundamentals of cyber and information security and understand the mindset of the hacker well enough to prevent attacks.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Can explain the ideas and implementations of common programs used to exploit vulnerabilities in well protected software systems to obtain some privileges
- Can explain the most common vulnerabilities of today, both within software and networking. Can identify vulnerabilities within real world software
- Can write programs to exploit vulnerabilities within vulnerable systems. Can explain in detail methods to prevent common vulnerabilities and their exploitation within software and networking

Assessment:

Not provided in course catalogue.

Reading Material:

Lecture notes provided by teacher.

T-423-ENOP Engineering Optimization

Credits: 6 ECTS
Year: one
Semester: spring term
Type of course: advanced elective undergraduate course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: 1) working knowledge of MAtlab programming 2) calculus (elementary linear algebra, in particular, vector/matrix operations and linear systems; basic knowledge of derivatives, including Taylor expansion).
Organization of course: three-week course
Teacher: Slawomir Koziel
Language of teaching: English

Description:

The course introduces the concept and methods of engineering optimization. Major topics discussed throughout the course are: formulation of unconstrained and constrained optimization problems, objective functions, classification of optimization methods, first- and second-order optimality conditions, gradient-based search methods, derivative-free optimization, stochastic search methods including multi-agent systems and evolutionary algorithms, multi-objective optimization, surrogate-based optimization with focus on space mapping, functional and physical surrogate modeling, design of experiments, model selection and validation, as well as solving real-world engineering optimization problems with interfacing of commercial simulators. The relevant material concerning Matlab programming as well as calculus in the scope necessary for the course will also be given.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Formulate engineering optimization problem, corresponding objective functions and constraints,
- Select appropriate optimization/modeling methodology,
- Implement basic optimization and modeling procedures as well as develop necessary Matlab code,
- Solve problems using existing packages, in particular Matlab and Matlab's Optimization Toolbox,
- Visualize the optimization process and the results.

Assessment:

Grades are based on the evaluation of reports and contribute 100% of the total grade.

Reading Material:

Lecture notes provided by teacher.

T-431-HANE Practical Networks

Credits: 6 ECTS
Year: one
Semester: spring
Type of course: advanced elective undergraduate course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-107-TOLH, Computer Architecture, T-215-STY1 Operationg systems
Organization of course: Three-week course
Teacher: To be updated
Language of teaching: Icelandic

Description:

The importance of networks is much more than most people realize. If everything is okay no one knows of their existence, but in the event of failures and problems in networks this can affect one's work and play that is involved online. Knowledge of how the network works and is structured is missing, even for those who use it the most, like programmers and system administrators. The evolution of technology means that the importance of networks is increasing, we now see communications being moved to the network and internet. The network is thus becoming more part of our security and coordination. The foundation of all communications is networks and is therefore essential to have an understanding and thorough knowledge of the functionality and possibilities. This course seeks to create a solid foundation that will be useful for anyone intending to establish themselves in information technology. The course is part lecture but mostly it is project based, which utilize the knowledge gained from the lecture. The objective is to teach design and implementation of networks, how requirements of performance and accessibility influence implementation of networks. We go over what is necessary to design and implement a network. This is broken into three parts : 1. Wired communication: Network equipment (Routers, switches), X area networks and protocols 2. Wireless communication: UMTS, 802.11, communications, antennas, wireless security 3. Security: L2/L3 Security, communications, VPN, encryption/decryption, firewalls and IPS/IDS. At the end of the course students have created a coherent network which include all previously mentioned parts.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Be able to describe the importance of networks and good installation for their business operations.
- Be able to describe the structure of networks and the equipment that the network consists of.
- Be able to describe what the trend has been in network systems and how they are likely to develop in the future.

Skills:

- Be able to design and set up a simple network, both wired and wireless.
- Be able to define and apply basic safety methods for networks

Competences:

- Be able to identify the needs for performance and security of networks.
- Be able to report common defects and faults in networks and improved them.

Assessment:

Not available in course catalogue.

Reading Material:

Lecture notes provided by teacher.

T- 501-FMAL Programming Languages

Credits: 6 ECTS
Year: one
Semester: spring term
Type of course: advanced undergraduate elective course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-201-GSKI, Data Structures, T-419-STR2, Discrete Mathematics II
Organization of course: three-week course
Teacher: Tarmo Uusstalu
Language of teaching: English

Description:

The evolution of programming languages is an important factor in computer science. The course describes this evolution from the first programming languages to the more recent languages. Different types of programming languages are discussed and their characteristics compared. Programming languages syntax is introduced as well as Backus-Naur Form (BNF). Main characteristics of imperative languages are examined, particularly regarding to scope rules and procedure activations. The main characteristics of object-oriented languages will be covered. The constructs of functional programming languages are examined with emphasis on Lambda calculus. Logic programming is introduced. Students are introduced to the design and syntax of various languages and experiment with several programming projects using some of these languages.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Be able to describe formal methods used for describing programming languages.
- Know the role of the individual components of a compiler.
- Be able to describe the difference between static and dynamic scope rules.
- Be able to describe the run-time stack and the role and implementation of activation records.
- Be able to define control abstraction and data abstraction.
- Know the main characteristics of object-oriented languages.
- Know the main characteristics of functional languages.
- Know the main characteristics of logic languages.
- Know the history and trends of language developments.
- Be able to use and define a context-free grammar for a simple programming language.
- Be able to program a simple compiler.
- Be able to program in a functional programming language.
- Be able to program in a logic language.

Assessment:

Four assignments	70%
Final exam	30%
Total	100%

Reading Material:

Peter Sestoft. *Programming Language Concepts*, 2nd ed. Springer, 2017. (Undergraduate Texts in Computer Science, Michael R. Hansen, Hans Rischel). *Functional Programming Using F#*. Cambridge University Press, 2013. (

T-502-HERM Simulation

Credits: 6 ECTS
Year: one
Semester: fall
Type of course: advanced undergraduate elective course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-101 STA1, Calculus 1, T-302- TOLF, Statistics I
Organization of course: three-week course
Teacher: Sigurður Óli Gestsson
Language of teaching: Icelandic

Description:

The focus of the course is to develop understanding of simulation concepts, and to clarify the advantages and limitations of simulation. We then look at discrete-event simulation using Simul8, a widely used simulation modelling language for a variety of application areas.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- • Understand the discrete-event simulation process.
- • Demonstrate a basic understanding of how simulation software computes its answers.
- • Analyse a real situation, model it, and build a simulation model to test hypotheses.
- • Define methods for validating and verifying a simulation model.
- • Specify ways a computer can generate uniform and non-uniform random numbers.
- • Awareness of problems that can cause bias in simulation models.
- • Select statistical models from simulation input.
- • Use statistical techniques to determine which of two simulated systems is better.

Assessment:

Homework 1	15%
Homework 2	15%
Final project	70%
Total	100%

Reading Material:

Simulation Modelling and analysis by Averill Law 5th ed. Lecture notes provided by teacher.

T-504-ITML Introduction to Machine learning

Credits: 6 ECTS
Year: one
Semester: fall
Type of course: advanced undergraduate elective course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-301-REIR, Algorithms, T-317-CAST, Calculus and Statistics, T-419-STRA2, Discrete Mathematics II
Organization of course: twelve-week course
Teacher: Stephan Schiffel and Krisín Bestla Þórsdóttir
Language of teaching: English

Description:

This course presents an overview of the field of machine learning, which deals with finding patterns and rules in large datasets. Such rules can then be used to predict outcomes of future events, for example with the aim of improving decision making in a wide range of business and manufacturing disciplines. In this course we will study machine learning techniques for classification, clustering, and association analysis as well as other selected techniques. In addition to introducing the underlying theory the methods will be used to solve practical problems.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Know how data mining is carried out.
- Recognize different types of training data and how to deal with common problems that arise, such as incomplete data.
- Be familiar with key algorithms and models used for classification, including decision trees, set of rules, Naïve Bayes, neural networks and support vector machines.
- Know the basic algorithms used with clustering, including K-means.
- Know the basic algorithms used to find relationships in data (e.g., association analysis).
- Be familiar with basic ideas behind evolutionary and reinforcement learning.

Skills:

- Be able to use software tools and programming libraries for data mining to categorize and cluster data.
- Be able to set up problems and apply data mining techniques to solve them.

Competences:

- Be able to determine the mechanical data mining strategies best suited to the solution of various practical problems, and be ready to use data mining tools and libraries to their solution

Assessment:

Homework assignments and in-class quizzes	25%
Two projects	30%
Final exam	45%
Total	100%

Reading Material:

Lecture notes provided by teacher.

T-511-TGRA Computer Graphics

Credits: 6 ECTS
Year: one
Semester: fall
Type of course: advanced undergraduate elective course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-301-REIR, Algorithms
Organization of course: twelve-week course
Teacher: Kári Halldórsson
Language of teaching: English

Description:

Computer graphics is an increasing part of the projects of today´s programmer. The first part of this course covers the use of the OpenGL library, vector tools for graphics, transformations of objects and polygonal meshes. The second part deals in more detail with three-dimensional drawing with emphasis on perspective, depth, light and colour. Finally, several issues regarding the implementation of a renderer are presented, in addition to curve and surface design. During the course students build several programs related to the course material.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Be familiar with the algorithms and calculations used when three-dimensional images are drawn on screen in real time (pipeline graphics), including, model transformations, perspective transformations, lighting, shading, clipping and rasterization.
- Be familiar with methods in OpenGL that implement these algorithms and calculations and how they are used in graphics applications such as computer games (OpenGL pipeline).
- Know how the flow in a graphical real-time application (i.e. computer game) is implemented, with respect to input, movement and drawing.

Skills:

- Be able to use the OpenGL standard to draw a three-dimensional image on a screen.
- Be able to implement a drawing loop which draws a motion picture, frame by frame, in real time.
- Be able to implement a programming loop that receives input and output, moves things, makes decisions and draws each frame with respect to camera angles and objects in a three-dimensional space.

Competences:

- Be able to implement three-dimensional video games and real time animations with the OpenGL standard.

Assessment:

Hand in assignment	10%
Programming assignments	50%
Final exam	40%
Total	100%

Reading Material:

Lecture notes provided by teacher.

T- 519-STOR Theory of Computation

Credits: 6 ECTS
Year: one
Semester: fall term
Type of course: mandatory course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-103, STST, Discrete Mathematics for Engineering, T-301-REIR, Algorithms, T-419-STR2, Discrete Mathematics II
Organization of course: twelve-week course
Teacher: Antonios Achiellos
Language of teaching: English

Description:

The main topic of this course is the theoretical basis of computer science. Various types of finite automata are introduced and connected to the formal definition of a programming language. Turing machines are introduced as a theoretical model for computation. Computability is discussed and the classification of solvable and unsolvable problems. Finally, there is a discussion of complexity classes and the classification of algorithmically hard and easy problems.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- A number of recurring themes, and a set of general principles that have broad application to the field of computer science
- The social, legal, ethical, and cultural issues inherent in the discipline of computing
- That software systems are used in many different domains. This requires both computing skills and domain knowledge.
- Software development fundamentals, including programming, data structures, algorithms and complexity.
- System fundamentals, including architectures and organization, operating systems, networking and communication, parallel and distributed computation and security.
- Mathematics, including discrete structures, statistics, calculus and optimization
- Software engineering principles, including a thorough understanding of software analysis and design, evaluation and testing and software quality and correctness.
- Software engineering processes, including management of complex software development projects.
- Application fundamentals, including information management and intelligent applications.
- Multiple programming language, paradigms, and technologies

Skills:

- Know how to apply the knowledge they have gained to solve real problems
- Realise that there are multiple solutions to a given problem and these solutions will have a real impact on people´s lives
- Communicate their solution to others, including why and how a solution solves the problem and what assumptions were made
- Successfully apply the knowledge they have gained through project experience.
- Encompass an appreciation for the structure of computer systems and the processs involved in their constructions and analysis
- Understand individual and collective responsibilities and individual limitations as well as the limitations of technical tools.

- Understand the range of opportunities and limitations of computing.

Competences:

- Understand the multiple levels of detail and abstraction
- Recognise the context in which a computer system may function, including its interactions with people and the physical world.
- Able to communicate with, and learn from experts from different domains throughout their careers.
- Possess a solid foundation that allows and encourages them to maintain relevant skills a sthe field evolves.
- To be able to manage their own career development, including managing time, priorities, and progress
- Have developed interpersonal communication skills as part of their project experience
- Work effectively both individually and as a member of teams
- Make effective presentations to a wide range of audience about technical problems and their solutions
- Encompass an appreciation of the interplay between theory and practice.

Assessment:

Home assignments	30%
Quizzes	10%
Midterm exam	20%
Final exam	40%
Total	100%

Reading material:

Michael Spiser: Introduction to the theory of computation, 3rd edition. CENGAGE Learning.

T- 535-MECH Mechatronics II

Credits: 6 ECTS
Year: one
Semester: spring
Type of course: advanced undergraduate elective course for MSc in Computer Science, and Msc in Software Engineering.
Necessary Prerequisites: T-411-MECH, Mechatronics I
Organization of course: twelve-week course
Teacher: Baldur Þorgilsson
Language of teaching: English

Description:

Mechatronics-2 extends Mechatronics-1 by going into more details. While Mechatronics 1 is broader and more about getting results fast (what is possible), Mechatronics 2 is more about accuracy and how to match a design to a task with economy, accuracy and robustness in mind (what is the limit).
The course includes sensors, signal conditioning, interfacing, analogy-digital conversion, digital input/outputs, timers, low level embedded firmware programming, actuators, UARTs and serial communication. It is expected that the student is familiar with digital electronics, analog electronics and the programming language C.
Along with the lectures, each student has his/her own private project based on the fundamental elements of mechatronics: sense-think-act. For this project the student holds a lab notebook. At the end of the course the student delivers a report about the project.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Know what an embedded system is
- Know in details how a microcontroller works
- Knows of several options of sensors and actuators

Skills:

- Be able to program an embedded system
- How to interface various sensors and actuators to a microcontroller

Competences:

- Choose a microcontroller of a specific mechatronics task
- Choose sensors for a given mechatronic problem
- Optimize code for a given hardware platform
- Can complete a defined personal project in a systematic and predictable way

Assessment:

Homework	20%
Midterm evaluation, draft of report	15%
Final project presentation	15%
Two-minute video	10%
Lab notebook	10%
Report	30%
Total	100%

Reading Material:

Introduction to Mechatronic Design, International edition J. Edward Carryer, R Mathew Ohline, Thomas W. Kerry, Pearson 2011, ISBN 978-0-13-609521-7

T-603-THYD Compilers

Credits: 6 ECTS
Year: one
Semester: fall
Type of course: advanced elective undergraduate course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T- 501-FMAL Programming Languages
Organization of course: twelve-week course
Teacher: Yngvi Björnsson
Language of teaching: English

Description:

Compilers are the most important part of a programming development environment. The course defines the function and objective of a compiler. Lexical analysis of programs is discussed in detail, regular expression and finite automata defined and the use of Lex introduced. Top-down and bottom-up approaches in parsing are discussed precisely and the use of Yacc introduced. Implementation of error handling is illustrated, particularly semantic analysis. Finally, code generation is covered. Construction of a compiler will be a large component of the course.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Understand the structure and design of compilers.
- Understand the role and function of lexical analysers, parsers and code generators.
- Have the theoretical foundation necessary for compiler construction.

Skills:

- Be able to use regular expressions and finite machines to perform lexical analysis.
- Be able to use fragmented grammar and both top-down and bottom-up parsing methods.
- Be able to use software that makes lexical analysers and papers.

Competences:

- Be able to design and build a simple compiler.

Assessment:

Lexical analysis	15%
Syntax Analysis	15%
Semantic Analysis and Intermedia Code Generation	13%
Labs	10%
Project	45%
Total	100%

Reading Material:

Lecture notes provided by teacher.

T-622-ARTI Artificial Intelligence

Credits: 6 ECTS
Year: one
Semester: spring
Type of course: advanced elective undergraduate course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-301-REIR, Algortihms.
Organization of course: twelve-week course
Teacher: Adín Ramírez Rivera
Language of teaching: English

Description:

Artificial intelligence (AI) is devoted to the computational study of intelligent behavior, including areas such as problem solving, knowledge representation, reasoning, planning and scheduling, machine learning, perception and communication. This course gives an overview of the aforementioned AI subfields from a computer science perspective and introduces fundamental solution techniques for addressing them. On the completion of the course the students should have a good overview of the field of artificial intelligence (AI) and a thorough understanding of the fundamental solution methods used to attack a wide variety of AI-related problems. In addition, the student should have gained experience building a small special-purpose AI system.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Be able to name methods for modelling and reasoning with imperfect information, such as Bayesian networks.
- Be able to describe problems and possible solutions for acting in continuous, partially observable and dynamic environments.
- Be able to describe different types of machine learning methods.
- Be able to classify autonomous agents and environments that agents operate in.
- Be able to compare and implement different search methods and optimizations for problem solving in single-agent and adversarial environments.
- Be able to use logic for knowledge representation and problem solving.
- Be able analyze a problem, select a well-suited AI method and create an agent to solve that problem.

Assessment:

Homework assignment, labs, and quizzes	20%
Projects	20%
Final exam	40%
Total	100%

Reading Material:

Artificial Intelligence: A modern Approach by Russel and Norvig.

T-624-CGDD Computer Game Design & Development

Credits: 6 ECTS
Year: one
Semester: fall
Type of course: advanced elective undergraduate course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-301-REIR, Algorithms.
Organization of course: twelve-week course
Teacher: Steingerður Lóa Gunnarsdóttir.
Language of teaching: Icelandic

Description:

This course covers the theory and practice of designing and developing computer games, from generating initial concepts to creating a fully playable game. Computer games are interactive environments that serve a specific goal: some enable player fun, some convey rich emotions, and some change the way that people think about the world. The emphasis of this course will be on team-based collaboration, with each team working to design and develop a game from the start to finish. In support of this process, each team will progress through a structured sequence of challenges during lab time, as guided by the concepts that are discussed and practiced during class.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Be able to describe the formal elements of games and the relationships between them.
- Be able to explain some common game AI techniques.
- Be able to describe common forms and structures of narrative in games.
- Be able to discuss insights gained from games industry practitioners.
- Be able to describe some current directions in computer game research.

Skills:

- Be able to employ focused strategies to generate ideas for computer games.
- Be able to apply some practical paradigms for game design & development.
- Be able to communicate game ideas clearly and concisely.

Competences:

- Be able to navigate intellectual property concerns in game development.
- Be able to design and conduct a play-test to evaluate a game.
- Be able to design and develop a game demo in a limited amount of time.

Assessment:

group work methods, progress and final demo.

Reading Material:

Lecture notes provided by teacher.

T-631-SOE2 Software Engineering II-testing

Credits: 6 ECTS
Year: one
Semester: spring
Type of course: advanced elective undergraduate course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-303-HUGB, Software Engineering.
Organization of course: twelve-week course
Teacher: Grischa Liebel
Language of teaching: English.

Description:

Various studies show that over than 50% of efforts and costs of software development are devoted to activities related to testing. This includes test design, execution, and evaluation. This course is an introductory course in software testing. In which, students will learn quantitative, technical, and practical methods and techniques that software engineers use to test their software throughout the software lifecycle.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Understand what software testing is and why we need it.
- Understand the concepts and theory related to software testing.
- Learn the different types of formal coverage criteria.
- Differentiate between different techniques that can be used for software testing and when to apply each of them.
- Understand how software developers can integrate a testing framework into code development in order to incrementally develop and test code.

Skills:

- Identify the test requirements.
- Define a model of the software, then find ways to cover it.
- Derive the test plan and evaluate the test suite coverage.
- Learn to use automated testing tools in order to measure code coverage.

Competences:

- Design tests based on structures: graph, logic, and input space.
- Define coverage criterion, define the test requirements for each coverage criterion, and derive the test cases that satisfy a coverage criterion.
- Apply the coverage criteria and software testing techniques to uncover defects in a large software system.
- Use open-source testing tools such (e.g., JUnit and NUnit) to test a software system.

Assessment:

Assignments	20 %
Project	20 %
Labs	10 %
Final exam	50 %

Reading Material:

No textbook required. Lecture slides or notes will be provided.

T-634-AGDD Advanced Game Design & Development

Credits: 6 ECTS
Year: one
Semester: spring
Type of course: advanced undergraduate elective course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-624- CGDD, Computer Game Design & Development
Organization of course: twelve-week course
Teacher: Sigursteinn J Gunnarsson
Language of teaching: English

Description:

This course expands RU's prior offerings in game design & development with more advanced topics in game design as well as delving into useful aspects of interaction and experience design. Through lectures, lab exercises, and project work, students will learn and gain experience with a variety of game design topics. Working together in teams, students will design, develop, and critically analyze several smaller games, each focused on applying the concepts that are discussed in class. Each of these exercises will differ in terms of either the team's composition, the game's scope, or the constraints that the instructors provide to guide the creation process.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Discuss game design, interaction design, player experience.
- Explain different methods for game design.
- Understand the roles and responsibilities required in a game's production.

Skills:

- Critically analyze given game designs and interaction designs.
- Conduct design sessions involving players.
- Develop focused game prototypes.

Competences:

- Assess team health and their effect on it.
- Design game mechanics to achieve an intended experience.
- Analyze and evaluate game prototypes.
- Develop a game informed by past prototypes and research.

Assessment:

- Project, analysis reports and participation

Reading Material:

slides from lectures.

T-637-GEDE Game Engine Architecture

Credits: 6 ECTS
Year: one
Semester: spring
Type of course: advanced undergraduate elective course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: T-211-LINA Linear Algebra, T-301-REIR, Algorithms, T-511-TGRA, Computer Graphics.
Organization of course: twelve-week course
Teacher: Hannes Högni Vilhjámsson.
Language of teaching: English

Description:

The course covers the theory and practice of game engine software development, bringing together topics that range from large-scale software architectures and modern game programming paradigms to the design and implementation of subsystems for memory management, interface devices, resource management, rendering, collision, physics and animation. Through practical lab exercises and group projects, the students will get technical hands-on experience in C++ game development, including the use and development of supporting tool pipelines.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Be able to explain game engines and their role in game development.
- Be able to sketch the typical components of a run-time game architecture.
- Be able to explain programming paradigms and data structures that are commonly used in game development
- Be able to understand what goes on in the rendering pipeline.
- Be able to explain engine sub-systems that deal with start-up/shut-down, memory management, engine configuration, file system, game resources, game loop, rendering loop and interface devices
- Be able to explain game engines and their role in game development.
- Be able to use and extend a C++ graphics engine to develop tech demos.
- Be able to use industry standard C++ development and version control tools.
- Be able to apply 3D math, covering points, vectors and matrices, for solving game world problems. Be able to import resources from Digital Content Creation tools.
- Be able to read input from game interface devices.
- Be able to program a basic vertex and fragment shader. Be able to use a particle system to create visual effects.
- Be able to use a physics library for realistic object behavior.
- Be able to analyze and compare existing game engines with respect to game development goals and system requirements.
- Be able to research, design, implement and present a tech demo of a low-level engine feature.
- Be able to design new game engines or engine sub-systems, based on established practices and an insight into various architectural decisions (pros and cons).

Assessment:

Participation	5%
Labs	8%
Problem sets	12%
Engine Presentation	10%

Final Project	35%
Final Written Exam	30%
Total	100%

Reading Material:

Game Engine Architecture by Jason Gregory, CRC Press third ed. (2018).

T-701-REM4 Research Methodology

Credits: 8 ECTS
Year: One
Semester: spring term
Type of course: mandatory in MSc in Computer Science, MSc in Software Engineering, MSc in Artificial Intelligence and Language Technology and MSc in Data and Applied Data Science.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: Stefán Ólafsson

Description:

The main aim of this course is to introduce the student with the principles of conducting scientific research and gain experience in the writing of scientific text to prepare the student for writing their MSc thesis and research papers.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Relate Scientific problems to theoretical principles in Computer Science. This includes knowledge of the following topics: various types of finite automata, the formal definitions of programming languages and their connection with automata, Turing machines and computability theory, and algorithmic complexity classes.
- Describe research methodology, including basic history of science, the fundamentals of scientific writing. Give a scientific talk, evaluate a scientific paper, and discuss research ethics.
- Apply statistical principles, and software tools embodying those
- Discuss the underlying hardware and software infrastructure upon which applications are constructed. These concepts include computational paradigms, parallelism, cross-layer communications, state and state transition, resource allocation and scheduling, etc.
- Discuss advanced principles and techniques from elective areas. Areas of specialization include artificial intelligence (e.g., agent technology, computer games, robotics and virtual environments), concurrency theory (with emphasis on modelling and verification of reactive systems, process algebra, and structural operational semantics), databases (with focus on efficient indexing of multimedia databases), and language technology (e.g. tagging of Icelandic and software support for the analysis of Icelandic text).
- Give examples of established and potential applications of techniques developed within the chosen area of specialization.

Skills:

- Apply methods and tools to design, implement, test, document, and maintain computer-based systems and processes
- Apply research methods, techniques, and problem-solving approaches from the field of research in which they specializing.
- Communicate their solution to others, including why and how a solution solves the problem and what assumptions were made.
- Access, retrieve and evaluate relevant professional information
- Apply methods and tools, create information models for analysing complex real-world systems and processes, and devise efficient computer-based solutions for these
- Invent new software, methods, or tools.

Competence:

- Work in a collaborative manner with others on a team, demonstrating proficiency in project management and business practices, such as risk and change management.

- Independently propose a small-scale research project, plan its execution, undertake its development, evaluate its outcome and report on its results in a professional manner.
- Communicate effectively and professionally both in writing and by means of presentations to both specialist and a general audience.
- Possess a solid foundation that allows and encourages the to maintain relevant skills as the field evolves.
- Interpret and present theoretical issues and empirical findings.

Assessment:

Student oral and written introduction	5%
Peer-reviewed	5%
Summary or material/guest talks	15%
Writing of a research paper	50%
Final version of paper	15%
Poster design and presentation	10%
Total	100%

Reading material:

Lecture notes provided by teacher, research papers etc.

T-707- MOVE Modelling and Verification

Credits: 8 ECTS
Year: one
Semester: spring term
Type of course: mandatory course in MSc in Software Engineering. Elective course for other Master programmes at DCS.
Necessary Prerequisites: T-301-REIR, Algorithms
Organization of course: twelve-week course
Teacher: Anna Ingólfsdóttir

Description:

Study of mathematical models for the formal descriptions and analysis of programs. Study of formal languages for the specification of program behaviour. Particular focus on parallel and reactive systems. Verification tools and implementation techniques underlying them.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Relate scientific problems to theoretical principles in Computer Science. This includes knowledge of the following topics: various types of finite automata, the formal definitions of programming languages and their connection with automata, Turing machines and computability theory, and algorithms complexity classes.
- Describe research methodology, including basic history of science, the fundamentals of scientific writing. Give a scientific talk, evaluate a scientific paper, and discuss research ethics.
- Apply statistical principles, and software tools embodying those.
- Discuss the underlying hardware and software infrastructure upon which applications are constructed. These concepts include computational paradigms, parallelism, cross-layer communications, state and state transition, resource allocation and scheduling, etc.
- Discuss advanced principles and techniques from elective areas. Areas of specialization include artificial intelligence (e.g., agent technology, computer games, robotics and virtual environments), concurrency theory (with emphasis on modelling and verification of reactive systems process algebra, and structural operational semantics), databases (with focus on efficient indexing of multimedia databases), and language technology (e.g., tagging of Icelandic and software support for the analysis of Icelandic text).
- Give examples of established and potential applications of techniques developed within the chosen area of specialization.

Skills:

- Apply methods and tools to design, implement, test, document, and maintain computer-based systems and processes
- Apply research methods, techniques, and problem-solving approaches from the field of research in which they are specializing.
- Communicate their solution to others, including why and how a solution solves the problem and what assumptions were made.
- Access, retrieve and evaluate relevant professional information.
- Apply methods and tools, create information models for analysing complex real-world systems and processes, and devise efficient computer-based solutions for these.
- Invent new software, methods, or tools.

Competence:

- Work in a collaborative manner with others on a team, demonstrating proficiency in project management and business practises, such as risk and change management.
- Independently propose a small-scale research project, plan its execution, undertake its development, evaluate its outcome and report on its results in a professional manner.
- Communicate effectively and professionally both in writing and by means of presentations to both specialist and a general audience.
- Possess a solid foundation that allows and encourages them to maintain relevant skills as the field evolves
- Interpret and present theoretical issues and empirical findings.

Assessment:

Assignments	60%
Final Exam	40%
Total	100%

Reading material:

Modelling, Specification and Verification by L. Aceto, A. Ingólfsdóttir, Kim G. Larsen and J. Srba, Cambridge University Press, 2007.

T-717-SPST Speech Synthesis

Credits: 6 ECTS
Year: one
Semester: spring term
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites: none
Organization of course: three-week course
Teacher: Jón Guðnason

Description:

Text to speech (TTS) synthesis converts written language to speech. The aim of this course is to introduce the classical processing steps of TTS systems and to point towards the state-of-the-art developing in the field. Front-end processing for classical TTS converts text to linguistic units which is an abstract representation the speech corresponding to the text. The front-end processing includes text normalisation, part-of-speech-tagging, converting letters to sound units, phrase breaking and prosodic analysis. Feature engineering is then applied to the linguistic units in order to make them suitable for the back-end processing. Deep neural networks or other machine learning mechanism converts linguistic features to acoustic features which are then used to generate the final waveform through a vocoder. This process is designed using machine learning methods which are based on annotated speech recordings. The course will give a detailed overview of this process and the students will go through a tutorial based on Merlin and Icelandic TTS language resources. Assessing the quality of TTS systems is a non-trivial thing as it is most often based on subjective ratings. An overview of the most common methods is given.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- General TTS architectures, including unit selection, parametric synthesis and statistical approaches using deep neural networks and Wavenet.
- Main challenges in text normalization for a target language (e.g. Icelandic)
- State-of-the-art linguistic features
- The role of vocoding in speech synthesis and acoutic features
- Methods for transforming lingusitic features to acoustic features

Skills:

- identifying language resources needed for speech synthesis
- formatting and adapting language resources using software such as Festival, MaryTTS or Ossian
- setting up TTS DNN training using Merlin with appropriate hardware
- setting up and running a vocoder (e.g. WORLD)
- evaluating TTS systems using both objective and subjective approaches

Competence:

- Apply Python and shell scripting to control TTS software
- Use the scientific method in testing TTS implementations
- Using knowledge of machine learning, signal processing and/or linguistics in designing TTS systems.

Assessment:

Topic Quizzes	35%
Computer assignments	35%
Open Project	30%

Total	100%

Reading material:

Lecture notes and material provided by teacher.

T-720-ATAI Advanced topics in Artificial Intelligence.

Credits: 8 ECTS
Year: one
Semester: fall term
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites: Programming experience necessary (LISP, Prolog, Haskel or related is a plus). A prior introductory class in one or more of the following is recommended: Artificial intelligence, simulation techniques, cognitive science.
Organization of course: twelve-week course
Teacher: Kristinn Rúnar Þórisson
Language of teaching: English

Description:

The course focuses on the phenomenon of intelligence and how to create a truly intelligent machine. The course asks the fundamental questions that the founders of the field of artificial intelligence (AI) –Turing, McCarthy, Minsky and others – considered the field's central concern: *What is intelligence?* and *How can we implement intelligence in a machine?* In the past 10-15 years attempts to answer this question have been made under the rubric of *general machine intelligence (GMI),* artificial general intelligence (AGI), *developmental robotics* and *cognitive robotics*. Looking further into the future than allowed by mere linear extrapolations of popular technologies being applied in various industries today, the course centers on the issues of intelligence architecture, system autonomy, realtime attention, anytime planning, model-based knowledge representation, and holistic systems integration, or what the late Allen Newell referred to as *unified theories of cognition*.
Ideas from control theory, constructivist AI, systems theory and cybernetics provide a conceptual foundation. Historical background and relevant topics from constructionist AI ("good old-fashioned AI") as well as the ideas of cyberneticians and early pioneers of systems science provide a contrasting backdrop for our treatment of how to build more autonomous and self-contained intelligent systems than possible with today's methods. Relevance of AGI to autonomous robotics and systems operating in the physical world will be addressed.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Relate scientific problems to theoretical principles in Computer Science. This includes knowledge of the following topics: various types of finite automata, the formal definitions of programming languages and their connection with automata, Turing machines and computability theory, and algorithmic complexity classes.
- Describe research methodology, including basic history of science, the fundamentals of scientific writing. Give a scientific talk, evaluate a scientific paper, and discuss research ethics.
- Apply statistical principles, and software tools embodying those.
- Discuss the underlying hardware and software infrastructure upon which applications are constructed. These concepts include computational paradigms, parallelism, cross-layer communications, state and state transition, resource allocation and scheduling, etc.
- Discuss advanced principles and techniques from elective areas. Areas of specialization include artificial intelligence (e.g. agent technology, computer games, robotics and virtual environments), concurrency theory (with emphasis on modelling and verification of reactive systems, process algebra, and structural operational semantics), databases (with focus on efficient indexing of multimedia databases), and language technology (e.g., tagging of Icelandic and software support for the analysis of Icelandic text).
- Give examples of established and potential applications of techniques developed within the chosen area of specialization.

Skills:

- Apply methods and tools to design, implement, test document, and maintain computer-based systems and processes.
- Apply research methods, techniques, and problem-solving approaches from the field of research in which they are specializing.
- Communicate their solution to others, including why and how a solution solves the problem and what assumption were made.
- Access, retrieve and evaluate relevant professional information.
- Apply methods and tools, create information models for analysing complex real-world systems and processes, and devise efficient computer-based solutions for these.
- Invent new software, methods, or tools.

Competences:

- Work in a collaborative manner with others on a team, demonstrating proficiency in project management and business practices, such as risk and change management.
- Independently proposes a small-scale research project, plan its execution, undertake its development, evaluate its outcome and report on its results in a professional manner.
- Communicate effectively and professionally both in writing and by means of presentation to both specialist and a general audience.
- Possess a solid foundation that allows and encourages them to maintain relevant skills as the field evolves.
- Interpret and present theoretical issues and empirical findings.

Assessment:

Engineering projects	22%
Short essay	12%
Online class discussions	10%
Final project	16%
Final exam	40%
Total	100%

Reading material:

Lecture notes provided by teacher.

T-723- VIEN Virtual Environments

Credits: 8 ECTS
Year: one
Semester: fall
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites: T-511-TGRA, Computer Graphics. This course is taught both at graduate and undergraduate level, where undergraduates require the permission of instructor.
Organization of course: twelve-week course
Teacher: Hannes Högni Vilhjálmsson

Description:

This is a comprehensive course in both the theory and practice of Virtual Environments (VEs). Virtual Environments are simulations that engage the senses of users through real-time 3D graphics, audio and interaction to create an experience of presence within an artificial world. VEs are used in a variety of settings, including training, education, health, online collaboration, scientific visualization and entertainment. Their use is becoming more and more pervasive as hardware gets more capable of simulating reality in real-time (including graphics, physics and intelligent behavior). As part of the theoretical overview, the course will introduce the history of VEs, what kind of problems VEs have proven to be best at addressing, what are their shown limitations, what models of human-computer interaction apply to VEs and how these models are evolving and pushing the state-of-the-art in interactivity. The technical portion of the course will lead students through the construction and population of VEs in a very hands-on manner, covering topics such as world representation, real-time graphics and simulation issues, networked environments, avatars and interactive characters, event scripting and AI control, special real-time visual and aural effects and intuitive user interfaces.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Know what constitutes a virtual environment, why they have been created throughout history and how they are used today.
- Be able to think critically about virtual environments as a user interface and design effective environments.
- Understand how humans construct a mental image of their environment using visual cues and how this can be exploited.
- Know the difference between presence and immersion, and understand how these may be measured.
- Understand the principles of effective action in virtual environments, including concepts such as flow, implicit constraints, explicit constraints and contextual action.
- Be familiar with the roles of characters in virtual environments and the common ways to make them autonomous and to animate them.
- Know what an avatar is and understand the issues that relate to level of control.
- Be familiar with the several techniques for constructing visual realism in virtual environments.
- Be able to create an interactive virtual environment in a scripting language and use a scene representation, models, terrain, lights, texturing, physics, animation, heads-up-display and shaders.

Assignments:

Discussion, Labs	10%
Programming assignments	20%
Final project proposal	5%
Final programming project	30%
Final project report	5%
Final written exam	30%

Reading material:

Lectures with live demonstrations and hands-on lab sessions. Lecture notes and papers provided by teacher.

T-725-MALV Natural Language Processing

Credits: 8 ECTS
Year: one
Semester: fall
Type of course: advanced elective undergraduate course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: Hannes Högni Vilhjálmsson, Helga Svala Sigurðardóttir, Hrafn Loftsson and Stefán Ólafsson.
Language of teaching: English

Description:

The goal of language technology (LT) is to develop systems which allow people to communicate with computers using natural languages. LT is an interdisciplinary field, requiring knowledge from subjects like linguistics, statistics, psychology, engineering and computer science. This course discusses fundamentals of natural language processing (NLP), which is one of the subfields of LT, and introduces research in the field, in part with regard to the Icelandic language. Students acquire understanding of the various stages of NLP, e.g. morphological analysis, part-of-speech tagging, syntactic analysis, semantic analysis, discourse and dialogue. In the course, students work on programming projects related to the aforementioned stages.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Know the main methods of processing required for computers to analyse and understand texts in a human language.
- Understand the strengths and weaknesses of current Natural Language Processing (NLP) technology.
- Know the main models and algorithms used in NLP, such as in morphological analysis, part-of-speech tagging, parsing, semantic analysis, and discourse and dialogue analysis.
- Know at least one programming language suitable for text processing. Be able to write simple NLP applications and present their work both orally and in writing.
- Be able to evaluate the performance/accuracy of NLP systems. Be aware of current research in NLP.

Assessment:

Quizzes	5%
Labs	15%
Individual projects/assignment	20%
Final project	30%
Final exam	30%
Total	100%

Reading Material:

"Speech and Language Processing", by Jurafsky & Martin.
"Natural Language Processing with Python", by Bird, Klein & Loper.

T-731-MLTH The structure of Icelandic and language technology (UI)

Credits: 10 ECTS
Year: one
Semester: fall
Type of course: elective master course for MSc in Artificial Intelligence and Language Technology. Engineering.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: contact programme administrator for information.
Language of teaching: Icelandic

Description:

course taught at University of Iceland.

Learning outcomes:

see in Ugla the student handbook of UI.
After completion of the course the student will hold a knowledge, skills and competence of:
See in Ugla the student handbook of UI.

Assessment:

Information available in Ugla, student handbook of UI.

Reading Material:

Information available in Ugla, student handbook of UI.

T-732- FSAA Financial simulation and analysis-systems

Credits: 8 ECTS
Year: one
Semester: spring term
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: Jacky Mallet

Description:

This is a practical course based on using computer simulation to explore and understand the behaviour of the modern financial system. Students will gain an overview of how agent based simulation and modelling is performed, and using the Threadneedle Financial Simulation System (Java), will create agent based simulations that allow software agents to interact with each other using lending, market based trading, foreign exchange, etc.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:
Students will construct their own simple economies, perform scientific experiments on them, and examine why they have crashed the banking system (again). By the end of the course, students should be able to understand and explain:

- Design and programming approaches for simulating network-based flow systems
- Differences between simulation and modelling
- Be able to build simple economic simulations and analyse their behaviour
- Financial systems such as market trading, fractional reserve banking, cryptocurrencies
- Long term behaviour of the financial system and its sensitivity to previous conditions
- Assess the interaction of debt, interest rates and monetary expansion as inputs to financial decision making
- Understand the root causes of credit crises, and in particular the crisis of 2020
- Apply understanding of financial system to personal and company financial decisions

Assignments:

Project proposal	30%
Final project	50%
Class participation	20%
Total	100%

Reading material:

Lecture notes provided by teacher

T-732-ISIT Introduction to embedded systems and the internet of things

Credits: 6 ECTS
Year: one
Semester: fall
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites: T-445-STYR, Operating systems, T-301 Algorithms, T-411 MECH, Mechatronics I
Organization of course: three-week course
Teacher: Marcel Kyas
Language of teaching: English

Description:

The Internet of Things (IoT) is a network of devices in our network that communicate and collaborate. IoT is changing our world. In this course you will learn the importance of IoT in society, the current components of typical IoT devices and trends for the future. IoT design considerations, constraints and interfacing between the physical world and your device will also be covered. You will also learn how to make design trade-offs between hardware and software. You will learn about the key components of networking to ensure that you understand how to connect the device to the Internet. Besides the functional design considerations of embedded devices, you will learn about the economic trade-offs. You will apply methods for ensuring that the built system meets high quality standards, be reliable, be resilient (handle errors as gracefully as possible), and secure. You will design a microcontroller-based embedded system. The focus of your project will be to design the system so that it can be built on a low-cost budget for a real-world application. The system should connect to the internet to supply data or to be controlled from the internet. To complete this project, you´ll need to use all the skills you´ve learned in the course (programming microcontrollers, system design, interfacing, etc.).

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Explain the concepts and components of embedded systems and IoT devices
- Produce schedules for real-time scheduling of an application
- Illustrate and implement network protocols like 6LoWPAN
- Demonstrate the functionality, reliability and security of an embedded system and an IoT device
- Implement embedded applications in C/C++ or Ada with or without support of an operating system
- Design an IoT application systematically

Assessment:

Attendance	1%
Learning Portfolio	24%
Design log book	15%
Project Demonstration	15%
Project Presentation	15%
Project Report	30%
Total	100%

Reading material:

Lecture notes. Cirani et al. (2018): Internet of things: Architecture, Protocols and Standards. McEwan and Cassimaly (2014): Designing the Internet of things, Marwedel (2018): Embedded system design, Knapp, Zeratsky and Kowitz (2016): Sprint: How to solve a problem and test new ideas in just five days. Additional papers recommended by teacher.

T- 740- SPMM Software Project Management

Credits: 8 ECTS
Year: one
Semester: fall
Type of course: mandatory course for MSc in Computer Science, MSc in Software Engineering and MSc in Applied Data and Data Science.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: Birna Íris Jónsdóttir
Language of teaching: English

Description:

The Software Project Management course covers a wide range of methods, activities, and tools to assure timely delivery of the software systems that meet specified requirements within project resources in a structured and organized way. It also covers the basics of Project Management and the importance of team work. The course introduces some of the methods and metrics used in software project estimation and risk management, in addition, setting up a project proposal, working with project portfolio and resource management. The course also covers the software quality management and explains the role of standards, policies, and procedures to ensure the software quality.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Relate scientific problems to theoretical principles in Computer Science. This includes knowledge of the following topics: various types of finite automata, the formal definitions of programming languages and their connection with automata, Turing machines and computability theory, and algorithmic complexity classes.
- Describe research methodology, including basic history of science, the fundamentals of scientific talk, evaluate a scientific paper, and discuss research ethics.
- Apply statistical principles, and software tools embodying those.
- Discuss the underlying hardware and software infrastructure upon which applications are constructed. These concepts include computational paradigms, parallelism, cross-layer communication, state and state transition, resource allocation and scheduling, etc.
- Discuss advanced principles and techniques from elective areas. Areas of specialization include artificial intelligence (e.g. agent technology, computer games, robotics and virtual environments), concurrency theory (with emphasis on modelling and verification of reactive systems, process algebra, and structural operational semantics), databases (with focus on efficient indexing of multimedia databases), and language technology (e.g., tagging of Icelandic and software support for the analysis of Icelandic text).
- Give examples of established and potential applications of techniques developed within the chosen area of specialization.

Skills:

- Apply methods and tools to design, implement, test, document, and maintain computer-based systems and processes
- Apply research methods, techniques, and problem-solving approaches from the field of research in which they are specializing.
- Communicate their solution to others, including why and how a solution solves the problem and what assumptions were made.
- Access, retrieve and evaluate relevant professional information

- Apply methods and tools, create information models for analysing complex real-world systems and processes, and devise efficient computer-based solutions for these.
- Invent new software, methods, or tools

Competences:

- Work in a collaborative manner with others on a team, demonstrating proficiency in project management and business practises, such as risk and change management.
- Independently propose a small-scale research project, plan its execution, undertake its development, evaluate its outcome and report on its results in a professional manner.
- Communicate effectively and professionally both in writing and by means of presentations to both specialist and a general audience.
- Possess a solid foundation that allows and encourages them to maintain relevant skills as the field evolves.
- Interpret and present theoretical issues and empirical findings.

Assessment:

Various project throughout the semester. No final exam.

Team project	10%
Risk Analysis	5%
The Project proposal	10%
Prioritization	5%
The Process and Architecture	15%
The Test Cases and Prioritized Backlog	10%
The Final Report with Prototype or System	35%
The Presentation	10%
Total	100%

Reading material:

No single book. References to books and articles as well as information online e.g.:

- The Phoenix Project, Gene Kim, Kevin Behr, George Spafford
- Effective Project Management, Robert K. Wysocki
- Project Portfolio Management, Harvey A. Levine
- World Class IT, Peter A. High
- PMI, Project Management Institute, https://pmi.org

T-742-CSDA Computer Security- Defence against the Dark Arts

Credits: 8 ECTS
Year: one
Semester: spring
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites: none
Organization of course: twelve -week course
Teacher: Jacqueline Clare Mallett
Language of teaching: English

Description:

In this course we will examine the challenges of computer and network security in the current era of state financed cyber warfare, with an emphasis on developing methods for active and passive defence. Lectures will cover common programming flaws, penetration testing, introductory cryptography, security architectures, and information warfare. A series of laboratories will provide practical experience in defending and attacking computer-based systems.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Understand and identify the source of common security flaws in computer and web applications on both Windows and Linux architectures
- Understand the theory of secure network architectures and topologies.
- Understand theoretical contingency planning methodologies and procedures for business, disaster recovery, and incident response.
- Be familiar with historical issues in computer security and the challenge of legacy systems.
- Be familiar with sources of current information, BlackHat Conferences, Bug Bounty programs, Mailing Lists, etc.
- Use vulnerability detection tools such as OpenVas, Snort, Kali, etc
- Be able to identify and protect against botnets, viruses, ransom ware, remote access attacks, etc
- Be able to conduct penetration tests, and identify insecure practices and procedures.
- Be familiar with emerging European Security Standards.

Assessment:

Final Project	60%
Assignments	20%
Quizzes	10%
Class Participation	10%
Total	100%

Reading material:

Readings and Lectures
Art of War by Sun Tzu https://suntzusaid.com
Violent Python T.J. O´Connor
Other material provided by teacher

T-743-PLSA Semantics with application

Credits: 6 ECTS
Year: one
Semester: spring term
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites: none
Organization of course: three-week course
Teacher: Tarmo Usstalu
Language of teaching: English

Description:

Programming language semantics is about formal description of meanings of programs so there cannot be any misinterpretation.
This is an introductory course on programming language semantics, with applications to compiler correctness and correctness of source-level program analyses and transformations.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Account for the main semantic styles used for capturing the meaning of programs in a formal way, both in theory and on examples.
- Relate different semantic styles, and compare their strengths and weaknesses.
- Use these semantic styles for program analysis, optimisation and verification, both in theory and as a basis for software tools.
- Extend a programming language with new language features, and extend its semantics accordingly.

Assessment:

Assessment in this course is based on three-week take-home assignment

Reading material:

Hanne Riis Nielson, Flemming Nielson. *Semantics with Applications: An Appetizer.* Springer, 2007.

T-747- RELE Reinforcement Learning

Credits: 6 ECTS
Year: one
Semester: spring term
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites:
Organization of course: three -week course
Teacher: Stephan Schiffel
Language of teaching: English

Description:

In the course, the students will learn about the main algorithms of Reinforcement learning, a computational approach to learning from interaction. We will cover different approaches such as Monte Carlo Methods, Temporal-Difference Learning, Policy Gradient Methods, and Deep Reinforcement Learning both theoretically and applied to practical examples.

Learning outcomes:

On completion of the course students should be bale to:

Knowledge:

- Explain the concepts Reinforcement Learning and the related terminology, such as policy, value function, markov decision process, optimality, exploration, exploitation, etc.
- Compare the main algorithms (Dynamic Programming, Monte Carlo, Temporal Difference Learning) wrt. their advantages and disadvantages on a specific problem domain

Skills:

- Implement the main algorithms and apply them to solve specific problems
- Use function approximation techniques to deal with large-scale problems

Competences:

- Model a problem as a markov decision problem and implement the model
- Make a well-informed decision on which methods to use for solving a specific problem

Assessment:

Homework assignments/lab assignments	20%
Presentation on a research paper or application of reinforcement learning	10%
Final Project	40%
Exam	30%

Reading material:

Reinforcement learning: An Introduction by Sutton & Barto

T-749-INDS Independent study

Credits: 2- 16 ECTS
Year: all years
Semester: spring/fall
Type of course: elective master course for all Master programmes at DCS.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: supervisor of student.
Language of teaching: English or Icelandic.

Description:

Independent study project must be well anchored in research within computer science. Each project is defined separately, at the beginning of the term and the student and the supervisor are defined before the project starts. Furthermore, a project proposal, along with the names of the student and the supervisor, for an independent project is sent to the research and graduate council for approval. Once the individual project is accepted, the project can be initiated.
The individual project should differ from the final project of the student, to ensure diversity in topic exposure through the entire program. A grade should be assigned immediately following the defence of the individual project. The grade will not be changed even if changes are made before final delivery. In case of a failing grade, the defence of the project can be repeated once. The final version should be delivered to the department within two weeks of the (first) defence of the individual project.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- The student has gained skills in defining, working on and professionally reporting on an individual study project.
- The student has gained specific knowledge, within a defined topic within the broad scope of computer science.
- The student has gained knowledge in writing through reporting on their findings during the individual study project.
- The student has gained skills in reflecting upon the advantages, and disadvantages in using technology for tackling societal problems questions after presenting their specific knowledge.

Reading Material:

Defined for each project separately.

Assessment:

The project is graded on the scale 1-10 by the supervisor. A passing grade for an individual project is 6.0.

T-754-SPLP Spoken Language Processing

Credits: 8 ECTS
Year: one
Semester: spring term
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: Jón Guðnason
Language of teaching: English

Description:

The course offers an introduction to the nature of spoken language, the main technological components used to deal with spoken language and the main application areas of spoken language processing. The course will introduce concepts such as automatic speech recognition, text-to-speech synthesis, natural language understanding and generation and spoken dialogue management. Other fields of spoken language processing such as speaker recognition and affective speech processing will be introduced as well. The course is centred on a large practical project where the students build their own spoken dialogue system using all the components that are introduced in the course.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Speech production
- Speech signal processing
- Prosody
- Phones and phonemes
- Lexicon and words
- Conversation
- Main components of automatic speech recognition (ASR)
- Main components of text to speech synthesis (TTS)
- Natural language understanding (NLU)
- Natural language generation (NLG)
- Dialogue management
- Affective speech processing
- Speaker recognition

Skills:

- Identifying good and bad acoustic environments for speech processing
- Setting up speech processing font-end for ASR
- Using ASR in a real environment and applying NLU
- Setting up language processing front-end for TTS using NLG
- Setting up TTS backend system
- Building a dialogue manager for specific task

Competences:

- Recognise strengths and weaknesses of specific spoken dialogue system components

- Using the scientific method in assessing spoken dialogue systems

Assessment:

Spoken Dialogue systems	30%
Speech Analysis	35%
Speech Synthesis and Final Demo	15%
Exam and quizzes	20%
Total	100%

Reading material:

Online book “Speech and Language Processing” by Jurafsky and Martin

T-764-DATA Big Data Management

Credits: 8 ECTS
Year: one
Semester: spring term
Type of course: mandatory course for MS in Software Engineering. Elective course in other MSc programmes.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: Gylfi Þór Guðmundsson
Language of teaching: English

Description:

Big data has become an umbrella term that refers to very large, and typically unstructured, data collections, as well as the algorithms and tools required to manage and utilise them. In this course we will look at the big data problem from a database perspective. We will read and discuss some key papers from the literature, in particular focusing on the large and diverse set of NoSQL database management systems, and obtain hands-on experience of working with big data technology, such as the Hadoop system

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Describe and debate the need and motivation for big data management.
- Discuss the key challenges associated with capturing, storing, curating, and querying large data sets.
- Describe big data analysis techniques and potential pitfalls. Explain the CAP theorem and its implications.
- Enumerate and discuss key partition management techniques.
- Describe and discuss differences between major data and query models and data management systems available for handling large datasets.
- Discuss ethical and legal concerns associated with big data collections.
- Apply a methodology for big data management projects.
- Use an automatically distributed computing framework to perform basic data analysis.
- Analyze the pros and cons of using different big data management systems and methodologies based on data and application characteristics.
- Analyze the pros and cons of using different automatically distributed computing frameworks based on data and application characteristics

Assessment:

Student participation and contribution

- Two short presentations of papers 20%
- Contributions to discussions in class on Piazza, and two short write-ups (start and end) 10%
- Assigned group projects
- Distributed hashing simulation with Scala 10%
- Data analysis with Spark 10%
- Data analysis with Spark 10%
- Individual project of the student´s choice 30%
- Oral final exam 10%

Reading material:

Principles of Big Data: preparing, sharing, and analysing complex information by Jules J. Berman, published by Morgan Kaufmann (imprint of Elsevier) in 2013.

T-814-INNO Creating a Complete Business Plan for a Technical Idea- Entrepreneurship and the Innovation Process

Credits: 8 ECTS
Year: one
Semester: spring
Type of course: advanced undergraduate elective course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: Páll Kristján Pálsson
Language of teaching: English

Description:

The course will give an overview of the running and managing business entities, including planning, cost analysis, human resource management and the role of managers and directors. The importance of continueous innovation is emphasised and related to the corporate lifecycles. As a practical project the students will develop a full business plan for a start-up of a technical idea.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Possess a clear understanding of the methodology and theoretical understanding of the managerial aspect used in defining and writing complete business plans.
- Understand innovation through the search for promising and inspiring ideas, idea evaluation and selection.
- Understand the basics of innovation through technical developmental processes and life-cycle of both products and businesses.
- Understand marketing through market analysis and create a marketing and sales plans that define customers and market demands.
- Understand the technical challenges in innovation and define developmental processes for solutions and plan actions accordingly.
- Understand the financial and funding aspect of innovation: Plan for capital and financing, revenue and cost estimates, cash flow plan and balance sheets. Also cost estimations, revenue, value assessment and sensitivity analysis.
- Understand innovation through the human aspect of management such as the need for direction, strategy, organisation chart, and human resource management.
- Define business opportunities and write a business plan for technical coplicated projects and interpret business plans.

Also, students should at the completion of the course know how to define business opportunities and make a text- and calculation models in order to evaluate the business opportunity according to demand, solution, profit and financing interest. To know how to avoid making mistakes when searching and evaluating business opportunities.

Skills:

- Students should be able to adapt the most important methods in optimizing business opportunities by analysing current situation and suggest methods that are likely to lead to optimal results in business planning and business plans. Also students shall be able to describe how to realize their proposals.

Competences:

- To possess the knowledge to present and interpret the outcome of a business plan and be able to establish and/or operate minor companies

Assessment:

projects, final exam.

Reading Material:

Lecture notes from teacher. Book: Handbók athafnamannsins, gerð rekstrar og viðskiptaáætlana.

T-814-PROD Managing Research and Development-Methods and Models

Credits: 8 ECTS
Year: one
Semester: spring
Type of course: elective advanced undergraduate course for MSc in Computer Science, and MSc in Software Engineering.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: Páll Krisján Pálsson
Language of teaching: English

Description:

We cover the engineering approach to innovation and entrepreneurship in lectures and a practical program in which students work in an active company.Due to increasing freedom in trade and internationalization the competition between companies is boosting. At the same time consumers demand new solutions, and the technology develops, resulting in older solutions becoming obsolete. Such conditions require constant innovation in companies' management and an understanding of the nature of innovation and entrepreneurship. Innovation is not only necessary in technological companies, but in all companies that intend to live and prosper.The course will cover innovation and the ability companies have for innovation in light of market, science, engineering, planning and financial presumptions. We deal with the terms innovation and entrepreneurship and their significance for modern management and put in context with success. We will also cover the value of knowledge, intellectual property rights and patent rights. Then we cover the internationalization and its impact on the innovation process.Special emphasis will be put on systematic development of the processes connected to innovation and worked on a project in a real company in this field.The aim is that the students aquire an understanding of the cause of success and mistake in innovation within a company and how companies can increase their ability for innovation and the importance of innivation and initative thinking for the existence of companies.Students will, at the end of the course, have aquired a steadfast knowledge of the method applied within product-developement and innovation in companies and be able to apply them on their own in the furture.

Learning outcomes:

Knowledge:

- Understand the presumption for success and the reasons for mistakes in innovation within companies.
- Understand how companies can develop, maintain and increase their skill for innovation and the value of innovation and initiative thinking for the existence of companies.
- Knowing company's methodology for developing products and innovation and pioneer thinking and the development of new products (goods and service).

Skills:

- Be familiar with company's methodology for developing products and innovation and being able to use it.
- Possess good knowledge of the main items of the innovator science and adaptation and integration of the knowledge of individual employees in order to create strong teams.
- Be able to evaluate the reasons for success and evade mistakes in innovation within companies.

Competences:

- Can by themselves take on a systematical construction and the processes connected to innovation in companies and possess the understanding, skill and knowledge to manage the development and running of such systems within companies.
- Be able to introduce and interpret the conclusions and proposals on the above-mentioned fields and be able to express themselves on those issues.

Assessment:

Reports	72%
Verbal exam	28%
Total	100%

Reading Material:

Various material which will be delivered by the teacher and used in the tuition, also links to websites and various artcles connected to the study-material.
Integrated Product Development, (IPD). Authors: Andreasen, M. Myrp & Lars Hein.
Handbók athafnamannsins; (HASSS) Stefna, stjórnun og starfsmenn, and Handbók athafnamannsins; (HARV) Gerð rekstrar og viðskiptaáætlana: Author: Páll Kr. Pálsson. Publisher Skyggni ehf.
Engineering Management; (EM) W. Dale Compton Chapters: 11, 14, 15 and 16.

T-879-MSRS MSc Research

Credits: 30 ECTS
Year: two
Semester: fall
Type of course: master course for all MSc programmes.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: supervisor of the student
Language of teaching: English

Description:

Discuss advanced principles and techniques from elective areas. Areas of specialization include artificial intelligence (e.g., agent technology, computer games, robotics and virtual environments), concurrency theory (with emphasis on modelling and verification of reactive systems, process algebra, and structural operational semantics), databases (with focus on efficient indexing of multimedia databases), and language technology (e.g., tagging of Icelandic and software support for the analysis of Icelandic text). Give examples of established and potential applications of techniques developed within the chosen area of specialization. Apply research methods, techniques, and problem-solving approaches from the field of research in which they are specializing Invent new software, methods, or tools. Independently propose a small-scale research project, plan its execution, undertake its development, evaluate its outcome and report on its results in a professional manner Communicate effectively and professionally both in writing and by means of presentations to both specialist and a general audience.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Understand the presumption for success and the reasons for mistakes in innovation within companies.
- Understand how companies can develop, maintain and increase their skill for innovation and the value of innovation and initiative thinking for the existence of companies.
- Knowing company's methodology for developing products and innovation and pioneer thinking and the development of new products (goods and service).

Skills:

- Be familiar with company's methodology for developing products and innovation and being able to use it.
- Possess good knowledge of the main items of the innovator science and adaptation and integration of the knowledge of individual employees in order to create strong teams.
- Be able to evaluate the reasons for success and evade mistakes in innovation within companies.

Competences:

- Can by themselves take on a systematical construction and the processes connected to innovation in companies and possess the understanding, skill and knowledge to manage the development and running of such systems within companies.
- Be able to introduce and interpret the conclusions and proposals on the above-mentioned fields and be able to express themselves on those issues.

Assessment:

Reports	72%
Verbal exam	28%
Total	100%

Reading Material:

Various material which will be delivered by the teacher and used in the tuition, also links to websites and various artcles connected to the study-material.

- Integrated Product Development, (IPD). Authors: Andreasen, M. Myrp & Lars Hein.
- Handbók athafnamannsins; (HASSS) Stefna, stjórnun og starfsmenn, and Handbók athafnamannsins; (HARV) Gerð rekstrar og viðskiptaáætlana: Author: Páll Kr. Pálsson. Publisher Skyggni ehf.
- Engineering Management; (EM) W. Dale Compton Chapters: 11, 14, 15 and 16.

T-780-MITS MITx Data Analysis: Statistical Modelling and Computation in Applications

Credits: 6 ECTS.
Year: one.
Semester: fall term.
Type of course: elective master course for MSc in Data and Applied Data Science.
Necessary Prerequisites: none.
Organization of course: twelve-week course.
Teacher: María Óskarsdóttir.
Language of teaching: English.

Description:

further information can be found at: https://micromasters.mit.edu/ds/

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:
See information at https://micromasters.mit.edu/ds/

Assessment:

projects and final exam.

Reading material:

further information can be found at: https://micromasters.mit.edu/ds/

T-781- MITP MITx Probability-The Science of Uncertainty and Data

Credits: 6 ECTS
Year: one
Semester: spring and fall term
Type of course: elective master course for MSc in Data and Applied Data Science.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: María Óskarsdóttir
Language of teaching: English

Description:

The world is full of uncertainty: accidents, storms, unruly financial markets, noisy communications. The world is also full of data. Probabilistic modeling and the related field of statistical inference are the keys to analyzing data and making scientifically sound predictions. Probabilistic models use the language of mathematics. But instead of relying on the traditional "theorem-proof" format, we develop the material in an intuitive -- but still rigorous and mathematically-precise -- manner. Furthermore, while the applications are multiple and evident, we emphasize the basic concepts and methodologies that are universally applicable. The course covers all of the basic probability concepts, including: multiple discrete or continuous random variables, expectations, and conditional distributions laws of large numbers the main tools of Bayesian inference methods an introduction to random processes (Poisson processes and Markov chains) The contents of this courseare heavily based upon the corresponding MIT class -- Introduction to Probability -- a course that has been offered and continuously refined over more than 50 years. It is a challenging class but will enable you to apply the tools of probability theory to real-world applications or to your research. This course is part of theMITx MicroMasters Program in Statistics and Data Science. Master the skills needed to be an informed and effective practitioner of data science. You will complete this course and three others from MITx, at a similar pace and level of rigor as an on-campus course at MIT, and then take a virtually-proctored exam to earn your MicroMasters, an academic credential that will demonstrate your proficiency in data science or accelerate your path towards an MIT PhD or a Master´s at other universities. To learn more about this program, please visit https://micromasters.mit.edu/ds/.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:
See information at https://micromasters.mit.edu/ds/

Assessment:

see information at https://micromasters.mit.edu/ds/

Reading material:

see information at https://micromasters.mit.edu/ds/

T-782-MITD MITx Data Analysis in Social Science-Assessing your Knowledge

Credits: 6 ECTS
Year: One
Semester: spring and fall term
Type of course: mandatory master course for MSc in Data and Applied Data Science.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: María Óskarsdóttir
Language of teaching: English

Description:

This course consists of an assessment that tests your knowledge on the course content from 14.310x - Data Analysis for Social Scientists, a statistics and data analysis course that will introduce you to the essential notions of probability and statistics. It will cover techniques in modern data analysis: estimation, regression and econometrics, prediction, experimental design, randomized control trials (and A/B testing), machine learning, and data visualization. It will illustrate these concepts with applications drawn from real world examples and frontier research. Finally, it will provide instruction for how to use the statistical package R and opportunities for students to perform self-directed empirical analyses. This assessment course should only be taken by learners who have completed and passed 14.310x - Data Analysis for Social Scientists and intend to pursue the MicroMasters credential in Statistics and Data Science. To get credit in this MicroMasters program: Enroll in both this assessment course and the content course 14.310x - Data Analysis for Social Scientists (Note: There is no additional fee to enroll in the content course), Complete the content course 14.310x with a passing grade, come back to this course and take the exam to earn your verified certificate that will count toward the MicroMasters credential in Statistics and Data Science. The timed exam will open between September 3, 2020 - September 14, 2020, but enrollment is open until August 17, 2020. This assessment course, along with the content course 14.310x - Data Analysis for Social Scientists, is part of the MITx MicroMasters Program in Statistics and Data Science. Master the skills needed to be an informed and effective practitioner of data science. You will complete this course and three others from MITx, at a similar pace and level of rigor as an on-campus course at MIT, and then take a virtually-proctored exam to earn your MicroMasters, an academic credential that will demonstrate your proficiency in data science or accelerate your path towards an MIT PhD or a Master´s at other universities. To learn more about this program, please visit https://micromasters.mit.edu/ds/.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Intuition behind probability and statistical analysis.
- How to summarize and describe data.
- A basic understanding of various methods of evaluating social programs.
- How to present results in a compelling and truthful way.
- Skills and tools for using R for data analysis

Assessment:

please look at: https://micromasters.mit.edu/ds/.

Reading material:

please look at: https://micromasters.mit.edu/ds/.

T-783-MITF MITx Fundamentals of Statistics

Credits: 6 ECTS
Year: one
Semester: summer and spring term
Type of course: mandatory master course for MSc in Data and Applied Data Science.
Necessary Prerequisites: none
Organization of course: three-week course
Teacher: María Óskarsdóttir
Language of teaching: English

Description:

Statistics is the science of turning data into insights and ultimately decisions. Behind recent advances in machine learning, data science and artificial intelligence are fundamental statistical principles. The purpose of this class is to develop and understand these core ideas on firm mathematical grounds starting from the construction of estimators and tests, as well as an analysis of their asymptotic performance. After developing basic tools to handle parametric models, we will explore how to answer more advanced questions, such as the following: How suitable is a given model for a particular dataset? How to select variables in linear regression? How to model nonlinear phenomena? How to visualize high-dimensional data? Taking this class will allow you to expand your statistical knowledge to not only include a list of methods, but also the mathematical principles that link them together, equipping you with the tools you need to develop new ones. This course is part of theMITx MicroMasters Program in Statistics and Data Science. Master the skills needed to be an informed and effective practitioner of data science. You will complete this course and three others from MITx, at a similar pace and level of rigor as an on-campus course at MIT, and then take a virtually-proctored exam to earn your MicroMasters, an academic credential that will demonstrate your proficiency in data science or accelerate your path towards an MIT PhD or a Master´s at other universities. To learn more about this program, please visit https://micromasters.mit.edu/ds/.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:
See information at https://micromasters.mit.edu/ds/

Assessment:

see information at https://micromasters.mit.edu/ds/

Reading material:

further information can be found at: https://micromasters.mit.edu/ds/

T-784-MITM MITx Machine learning with Python

Credits: 6 ECTS
Year: one
Semester: fall term
Type of course: mandatory master course for MSc in Data and Applied Data Science.
Necessary Prerequisites: none
Organization of course: three-week course
Teacher: María Óskarsdóttir
Language of teaching: English

Description:

Machine learning methods are commonly used across engineering and sciences, from computer systems to physics. Moreover, commercial sites such as search engines, recommender systems (e.g., Netflix, Amazon), advertisers, and financial institutions employ machine learning algorithms for content recommendation, predicting customer behavior, compliance, or risk. As a discipline, machine learning tries to design and understand computer programs that learn from experience for the purpose of prediction or control. In this course, students will learn about principles and algorithms for turning training data into effective automated predictions. We will cover: Representation, over-fitting, regularization, generalization, VC dimension; Clustering, classification, recommender problems, probabilistic modeling, reinforcement learning; On-line algorithms, support vector machines, and neural networks/deep learning. Students will implement and experiment with the algorithms in several Python projects designed for different practical applications. This course is part of theMITx MicroMasters Program in Statistics and Data Science. Master the skills needed to be an informed and effective practitioner of data science. You will complete this course and three others from MITx, at a similar pace and level of rigor as an on-campus course at MIT, and then take a virtually-proctored exam to earn your MicroMasters, an academic credential that will demonstrate your proficiency in data science or accelerate your path towards an MIT PhD or a Master´s at other universities. To learn more about this program, please visit https://micromasters.mit.edu/ds/. If you have specific questions about this course, please contact us atsds-mm@mit.edu.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Understand principles behind machine learning problems such as classification, regression, clustering, and reinforcement learning
- Implement and analyze models such as linear models, kernel machines, neural networks, and graphical models
- Choose suitable models for different applications Implement and organize machine learning projects, from training, validation, parameter tuning, to feature engineering.

Assessment:

see information at https://micromasters.mit.edu/ds/

Reading material:

further information can be found at: https://micromasters.mit.edu/ds/

T-785-MITx Capstone Exam in Statistics and Data Science

Credits: 6 ECTS
Year: one
Semester: fall term
Type of course: mandatory master course for MSc in Data and Applied Data Science.
Necessary Prerequisites: none
Organization of course: three-week course
Teacher: María Óskarsdóttir
Language of teaching: English

Description:

This capstone exam is the final part of the MITx MicroMasters Program in Statistics and Data Science. Complete the four courses in this program and take this virtually-proctored exam to earn your MicroMasters credential and demonstrate your proficiency in data science or accelerate your path towards an MIT PhD or a Master´s at other universities.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:
See information at https://micromasters.mit.edu/ds/

Assessment:

See information at https://micromasters.mit.edu/ds/

Reading material:

further information can be found at: https://micromasters.mit.edu/ds/

T-786- APDS Applied Data Science

Credits: 6 ECTS
Year: one
Semester: fall-term
Type of course: mandatory master course for MSc in Data Science and MSc in Applied Data Science.
Necessary Prerequisites: none
Organization of course: three-week course
Teacher: María Óskarsdóttir
Language of teaching: English

Description:

This course gives hands-on data science experience, covering concepts, tools, and techniques to build intelligent systems. The course teaches how to develop and deploy machine learning pipelines and to build deep learning architectures while working with large and complex datasets. The course covers supervised and unsupervised learning, deep learning, feature engineering, dimensionality reduction, visualization, and ethics. The evaluation of the course is project-based. The students will work with various real-life datasets while solving actual problems and present their results.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Know the fundamentals of data science. Understand various machine learning algorithms.
- Be familiar with various deep learning architectures.
- Possess a working knowledge of scikit-learn, keras and tensflow in Python.
- Know about the ethical challenges in data science.

Skills:

- Know how to create machine learning pipelines.
- Be able to use the scikit-learn, keras and tensor flow.
- Be able to collect, pre-process and visualize data, engineering features and do dimensionality reduction, tune models and hyperparameters, assess model performance, interpret model outcome and present them to non-expert.

Competences:

- Be able to carry out and end-to end data science project, from data acquisitions to actionable insights.
- Decide what kind of analysis approach is appropriate for a given problem.

Assessment:

Assignments	60%
Final Project	40%
Total	100%

Reading material:

Hands-on machine learning with Scikit-learn,Keras and TensorFlow (2nd edition) by Aurélien Géron (https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/).

T-809-DATA Datamining and Machine Learning

Credits: 8 ECTS
Year: one
Semester: fall term
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: Jón Guðnason and Michal Borsky
Language of teaching: English

Description:

Pattern recognition system, classifier design cycle and learning. Statistical pattern recognition, Bayesian decision theory, maximum likelihood and Bayesian parameter estimation. Linear models for classification. Principal component analysis. Multilayer neural networks. Nonparametric methods: k-nearest neighbours and Parzen kernels. Kernel methods and support vector machines. Unsupervised classification, K-means clustering, Gaussian mixture models and expectation maximization. Combination of classifiers, bagging and boosting.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Pattern recognition system, classifier design cycle and learning
- Statistical pattern recognition, Bayesian decision theory, maximum likelihood and Bayesian parameter estimation.
- Linear models for classification
- Principal components analysis
- Multilayer neural networks
- Nonparametric methods: k-nearest neighbours and Parzen kenrels.
- Kernel methods and support vector machines.
- Unsupervised classification, K-means clustering, Gaussian mixture models and expectation maximization.
- Combination of classifiers, bagging and boosting

Skills:

After the course the students should be able to apply the data mining methods and implement the machine learning algorithms presented in the course using standard programming languages such as Python or Matlab and software packages such as scikit-learn and Weka.

Competences:

After the course the students should be able to design a suitable machine learning algorithm for a real-world problem, evaluate its performance, compare different designs and implementations and interpret the results. The students should also be able to present findings and new results in the subject.

Assessment:

Quizzes	5%
Exams (No final exam)	40%
Homework	30%
Course Project	25%

Reading material:

Christopher M. Bishop; “Pattern Recognition and Machine Learning”
Duda, Hark, Stork; “Pattern Classification”

T-810-OPTI Optimization Methods

Credits: 8 ECTS
Year: one
Semester: fall term
Type of course: elective course for all master programmes at DCS.
Necessary Prerequisites: T-403-ADGE, Operation Research
Organization of course: twelve-week course
Teacher: Hlynur Stefánsson
Language of teaching: English

Description:

This course introduces the principal algorithms for linear, network, discrete, nonlinear, dynamic optimization and optimal control. Emphasis is on methodology and the underlying mathematical structures. Topics include the simplex method, network flow methods, branch and bound and cutting plane methods for discrete optimization, optimality conditions for nonlinear optimization, interior point methods for convex optimization, Newton´s method, heuristic methods, and dynamic programming and optimal control methods

Learning outcomes:

After the completion of this course students will be capable of using basic methods of Operations Research for analysing and solving complex decision problems. More specifically the student will be able to:

- Understand the properties of linear optimization and how it can be used to analyze and solve complex decision problems;
- Use and analyze different forms of linear optimization models;
- Understand and be capable of analyzing the geometry of linear optimization;
- Apply systematic methods and algoirthms for analysing and solving decision problems;
- Understand the importance and usefullness of linear optimization and its applications;
- Apply software to solve optimization models;
- Implement solution methods for linear optimization models and have in-depth understanding of the mechanics of the Simplex methods;
- Practice the use of sensitivity analysis and to derive formulas for sensitivity of model parameters;
- Understand integer programming and how it can used in decision making;
- Use the main solution methods for integer programming;
- Understand the special properties of network models and formulate practical problems as network models;
- Understand the nature of non-linear optimization problems and the challenges involved in solving the problems;
- Be familiar with different classes on non-linear optimization models and some of the available solution methods and algorithms;
- Understand the importance of optimization under uncertainty and be able to develop robust programming, change constraints and stochastic programming models;
- Be familiar with dynamic programming;
- Present results in a clear and organized manner.

Assessment:

Homework	5%
Group work	10%
Reports	5%
Exams	75%

Lowest exam	5%
Total	100%

Reading material:

Hillier and Lieberman, Introduction to Operations Research, 10th Edition, Pearson 2014.

T-811-Applied Probability

Credits: 8 ECTS
Year: one
Semester: fall term
Type of course: elective master course for all Master programmes at DCS.
Necessary Prerequisites: T-606-PROB Probability and Stochastic processes
Organization of course: twelve -week course
Teacher: Sverrir Ólafsson and Styrmir Hjalti Haraldsson
Language of teaching: English

Description:

This course will start by recalling some basic concepts in probability theory. Important discrete and continuous probability distributions will be introduced and applied to concrete problems. The concepts of expectations, variances and covariances will be introduced and applied to selected problems. The importance of the theorem of large numbers, central limit theorem and the consequences of these will be introduced. Markov chains will be discussed as well as Poisson and death – birth processes with several applications, including queueing theory. Basic stochastic processes such as Brownian motion and Wiener processes and their important role in the modelling and management of uncertainty will be discussed. Throughout the course examples and applications to various practical problems will be considered.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:
This course will cover some important topics in probability theory with particular emphasis on their application to practical problems. At the end of the course the student will have an appreciation of the important role probability plays in various areas of engineering and be able to apply it to a range of concrete real-world problems. This learning outcome can be broken down into the following sub outcomes:

- Understand the basic concepts of probability distributions and their role in the modelling of uncertain outcomes – both in the discrete and the continuous case • Use expectation, variance and covariance to model various probabilistic phenomena
- Apply conditional probabilities and Bayes's formula to events in the presence of partial information
- Understand jointly distributed random variables and functions of random variables
- Understand the theoretical basis of moment generating functions and their application to the construction of probability distribution functions
- Understand the theoretical basis of the limit theorems, the law of large numbers and important inequalities
- Understand the role of probability in Reliability applications
- Understand Poisson processes, birth and death processes and Markov processes and their roles in the modelling of queues
- Understand different types of queues and their classification
- Be able to estimate the performance of different queueing systems in terms of quantities such as, queue length, expected waiting time or the probability of system blockage
- Understand the role of stochastic processes in financial applications

Assessment:

Class exams	30%
Project work	10%
Final exam	60%

Reading material:

Lecture notes which will be sufficient for successfully competing the course. Recommendation to have the book: Sheldon M. Ross, Introduction to Probability Models, 11th edition, Academic Press, 2014. Additional paper provided by teacher and links and websides.

T-796- DEEP Introduction to Deep learning

Credits: 6 ECTS
Year: one
Semester: fall term
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites:
Organization of course: three-week course
Teacher: Yngvi Björnsson
Language of teaching: English

Description:

This course gives a comprehensive overview of the fundamentals of deep learning. We will cover deep feed-forward networks, regularization, and training optimization techniques for deep learning, convolutional- and recurrent networks, as well as practical methodologies and applications for deep learning. We will furthermore read recent scholarly articles on deep learning. There is also a sizeable hands-on part in the course where students use widely-used DL frameworks and techniques learned to solve interesting problems.

Learning outcomes:

The learning outcomes of the course are for participating students to be able to:

- Demonstrate a solid background in the fundamentals of deep learning (DL);
- Read and comprehend scholarly articles on current research in the field;
- Setup and use widely available DL platforms/frameworks for constructing deep
- networks of various complexity and use them for training and evaluating the networks.

Assessment:

Labs	10%
Reports	10%
Quizzes	10%
Assignments	10%
Presentations	10%
Exam	20%
Project	30%
Total	100%

Reading material:

http://deeplearningbook.org/ Deep Learning, Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016). MIT Press.
Lecture notes. Scientific articles (suggestions):

- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton (2015). Deep learning
- Olaf Ronnenberger, Philipp Fischer, and Thomas Brox (2015). U- Net: Convoluational Networks for Biomedical Image Segmentation.
- Martin Abadi et al. (2016). TensorFlow Martín Abadi et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. Download TensorFlow: A System for Large-Scale Machine Learning.
- Xiang Zhang, Junbo Zhao, and Yann LeCun (2015). Character-level Convolutional Networks for Text Classification Download Character-level Convolutional Networks for Text Classification.
- David Silver et al. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm Download Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.

T-891-MSTD Master Thesis Defence

Credits: 6 ECTS
Year: two
Semester: fall term
Type of course: mandatory course for all Master programmes at DCS.
Necessary Prerequisites: Students need to hand in a draft of a MS thesis that the supervisor deems qualified enough for evaluation by the project committee and then they can be signed up for the project defence (this course).
Organization of course: does not apply.
Teacher: the supervisor of the student.
Language of teaching: English or Icelandic.

Assessment:

The project is graded on the scale 1-10 by a project committee. A passing grade for a thesis is 6.0.

Reading material:

based on the master thesis of the student.

T-899-MSTH Master thesis

Credits: 60 ECTS
Year: two
Semester: every semester (duration is two semester- full time study)
Type of course: mandatory course for all Master programmes at DCS.
Necessary Prerequisites: T- 701-REM4 Research methodology
Organization of course: does not apply.
Teacher: the supervisor of the student.
Language of teaching: English or Icelandic.

Description:

In the research-based track, students complete at least 60 ECTS devoted to an individual research project under the supervision of a faculty member. Project or thesis supervision is only performed by mutual consent of the student and supervisor.
Before graduation, the student then submits a research thesis. The thesis must represent a body of original, individual research work, which in quantity and quality matches or exceeds the expectations of the thesis committee for two semesters of full time research. In cases where the thesis is part of a larger research project, or where other students or researchers have contributed to the topics represented in the thesis, the contribution of the student must be clearly identified in the thesis.
An open defence of the thesis must take place prior to the evaluation of the thesis. After the open defence, the thesis committee holds a closed session with the student.

Learning outcomes:

- After completion of the course the student will hold a knowledge, skills and competence of:
- Independently propose a research project,
  plan its execution, undertake its development, evaluate its outcome and report on its results in a professional manner.
- Interpret and present theoretical issues and empirical findings.
- Discuss advanced principles and techniques from elective areas of computer science.
- Give examples of established and potential applications of techniques developed within the chosen area of s pecialization.
- Apply research methods, techniques, and problem solving approaches from the field of research in which the y are specializing.
- Invent new software, methods, or tools.
- Communicate effectively and professionally both in writing and by means of presentations to both specialist and a general audience.

Assessment:

The thesis is graded following the thesis defence on the scale 1-10 by a project committee. A passing grade for a thesis is 6.0

Reading material:

Defined for each thesis separately.

T-991-TPDE Thesis Project Defence

Credits: 6 ECTS
Year: two
Semester: spring/fall
Type of course: mandatory course for all Master programmes at DCS.
Necessary Prerequisites: Students need to hand in a draft of a MS thesis that the supervisor deems qualified enough for evaluation by the project committee and then they can be signed up for the project defence (this course).
Organization of course: does not apply
Teacher: the supervisor of the student.
Language of teaching: English or Icelandic

Description:

An open presentation of the project must take place prior to graduation.
The committee members should attend the presentation of the students, either physically or remotely, and should hold a closed session as part of the presentation. The supervisor (and co-supervisor) must be in attendance during the presentation. If one committee member is unable to attend the presentation, a list of suggestions may be sent to the student and supervisor ahead of the presentation. Additionally, a list of questions may be sent to the supervisor. If two committee members are unable to attend, either physically or remotely, the presentation must be rescheduled.
A grade should be assigned immediately following the defence. The grade will not be changed even if changes are made before final delivery. In case of a failing grade, the defence can be repeated once. The final version should be delivered to the department within six months of the (first) defence.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- The student has gained skills in presenting the thesis
- The student has gained skills in answering questions after presenting his thesis

Assessment:

The project is graded on the scale 1-10 by a project committee. A passing grade for a thesis is 6.0.

Reading material:

Defined for each thesis separately.

E-402-STFO Mathematical Programming

Credits: 6 ECTS
Year: one
Semester: fall
Type of course: elective course for MSc in Computer Science.
Necessary Prerequisites: T-101-STA1, Calculus I, T-103- STST, Discrete Mathematics for Engineering, T-301-REIR, Algorithms, T-317, CAST, Calculus and Statistics, T-419-STR2, Discrete Mathematics II
Organization of course: twelve-week course
Teacher: Arnar Bjarni Arason
Language of teaching: English

Description:

Mathematics is generally discovered through experiments. Traditional tools for such experiments are pen and paper, and, of course, the mind. A (historically) recent addition to these tools is the computer. We will look at problems from several areas of mathematics and, in particular, how programming can be used as a means to better understand and ultimately solve those problems. This will involve designing and implementing algorithms, experimentation to make conjectures, and deductive/formal mathematics to prove conjectures. For programming we will use python/sage (https://cloud.sagemath.com).

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

Knowledge:

- Know how computers and algorithms are used in research, both in mathematics and computer science.
- Recognize linear programming as a method for solving problems.
- Recognize dynamic programming as a method for solving problems.
- Recognize search methods as a method for solving problems.
- Recognize brute force and other common solution methods for solving problems.
- Know when writing code is better, or worse, than trying to prove a problem by hand.
- Know several objects from discrete mathematics, such as permutations, graphs, games (like the Game of Life) and finite surfaces (tori and the Klein bottle).
- Know several objects from continuous mathematics, such as the critical points of functions of several variables.

Skills:

- Be able to use a computer to test conjectures and run simulations.
- Be able to use dynamic programming to solve problems.
- Be able to use linear programming to solve problems.
- Be able to use search and other common methods to solve problems.
- Be able to choose an appropriate method to deal with different problems.
- Be able to prove certain problems by hand, where running simulations is to time-consuming

Competences:

- Be able to use the Sage computer algebra system to assist them in other courses.
- Be able to recognize which kind of problems can be solved with the solution methods treated in the course.

Assessment:

Homework	80%

Final exam	20%
Total	100%

Reading material:

lecture notes provided by teacher.

V-713-ENTR Entrepreneurial Finance

Credits: 7,5 ECTS
Year: one
Semester: spring
Type of course: elective course for MSc in Computer Science.
Necessary Prerequisites: none
Organization of course: twelve-week course
Teacher: will be announced soon.
Language of teaching: English

Description:

This course provides an overview of the entrepreneurial finance process. It covers a wide range of topics associated with a venture's financial strategy: financial planning, sources of capital, exit strategies, financial contracting, and valuation. It focuses on integrating and extending one's basic knowledge of finance principles with the complexities that entrepreneurship, business strategy, and marketing add to the context of new ventures. The fundamental premise of the course is that financial strategy formulation and implementation is a crucial part of the venture development process. As such, the course is suitable for those who plan to engage in new venture development both as an independent effort and within a corporate or family business context. I expect that students taking this course will understand some of the basic finance such as financial statement preparation and discounted cash flow analysis.

Learning outcomes:

After completion of the course the student will hold a knowledge, skills and competence of:

- Identify different funding options.
- Describe the elements of a new venture's business and financial model.
- Explain the key issues associated with financial planning and fundraising in a new venture context.
- The ability to apply knowledge to different tasks of the entrepreneurial finance process.
- The student should:
- Analyze relevant issues in situations at the intersection of entrepreneurship and finance.
- Categorize the uncertainties / risks associated with the development of new ventures.
- Compare different funding options.
- The ability to apply knowledge and skills in entrepreneurial finance settings.
- The student should:
- Evaluate investment proposals.
- Formulate financial plans.
- Design investment proposals.

Assessment:

projects and exam.

Reading material:

lecture notes provided by teacher.