18th of October
*Subject to change

MSc in Software Engineering - 120 ECTS
Study plan and Module handbook

Table of Contents

18th of October
*Subject to change

M.Sc. IN SOFTWARE ENGINEERING.

To complete a M.Sc. in Software Engineering, students must complete 120 ECT, of which 60 ECTS are mandatory. Each course is either 8 ECTS or 6 ECTS except for the final project which is 30/60 ECTS. An example of study plan can be seen in the table; however, courses can be arranged differently as long as the rules of prerequisites are followed.

Study plan for mandatory courses

1. semester - fall

Course ID	Course name	ECTS	Length	Workload
T-519 STOR	Theory of computation	6	12 weeks	2L+1E
	Elective course	8	12 weeks	
	Elective course	8	12 weeks	
	Elective course	8	12 weeks	
	Elective three-week course 6 ECTS (skip if student take T-519).		3 weeks	

2. semester - spring

Course ID	Course name	ECTS	Length	Workload
T-701 REM4	Research Methodology	8	12 weeks	
T-707-MOVE	Modelling and Verification	8	12 weeks	
	Elective course	8	12 weeks	
	Elective course	6	3 weeks	

3. semester (Course-based track) - fall

Course ID	Course name	ECTS	Length	Workload
	Elective course	8	12 weeks	
	Elective course	8	12 weeks	
	Elective course	8	12 weeks	
	Elective course	6	3 weeks	

4. semester(Course-based) - spring

Course ID	Course name	ECTS	Length	Workload
T-899-MSTH	Master project	24	15 weeks	
T-991-TPDE	Master project defence	6	15 weeks	

3. semester (Research-based track) - fall

Course ID	Course name	ECTS	Length	Workload
T-879-MSRS	Master research	30	15 weeks	

4. semester(Research-based) - spring

Course ID	Course name	ECTS	Length	Workload
T-899-MSTH	Master thesis	24	15 weeks	
T-891-MSTD	Master thesis defence	6	15 weeks	

Study plan for elective courses

MSc in Software engineering	Credits	Length (weeks)		Term		Workload		
Elective Courses	**(ECTS)**	**12 w**	**3 w**	**F**	**S**	**L**	**E**	**L+E**
T-622-ARTI Artificial Intelligence	6	x			x	2	1	
T-431-HANE Practical Networks	6		x	x				x
T-498-GAGR Data Analysis	6	x			x	2	1	
T-504-ITML Introduction to Machine Learning	6	x		x		2	1	
T-511-TGRA Computer Graphic	6	x		x		2	1	
T-535-CPSY Cyber-Physical Systems	6	x		x		2	1	
T-603-THYD Compilers	6	x		x		2	1	
T-624-CGDD Computer Game Design & Development	6		x	x				x
T-634-AGDD Advanced Game Design & Development	6	x		x		2	1	
T-631-SOE2 Software Engineering II - Testing	6	x			x	2	1	
T-637-GEDE Game Engine Architecture	6	x			x	2	1	
T-423-ENOP Engineering Optimization (DE)	6		x		x			x
T-424-SLEE Sleep (DE)	6		x		x			x
T-502-HERM Simulation (DE)	6		x	x				x
T-717-SPST Speech Synthesis	6		x					x
T-720-ATAI Advanced topics in Artificial Intelligence.	8	x		x		2	0	
T-723- VIEN Virtual Environments	8	x		x		2	0	
T-725-MALV Natural Language Processing	8	x				2	0	
T-732- FSAA Financial simulation and analysis-systems	6		x		x			x
T-732-ISIT Introduction to embedded systems and the internet of things	6		x	x				x
T-738- VIRH Virtual Humans	8	x		x		2	0	
T-742-CSDA Computer Security- Defence against the Dark Arts	8	x			x	2	0	
T-743-PLSA Semantics with application	6		x		x			x

T-747- RELE Reinforcement Learning	8	x		x		2	1	
T-749-INDS Independent study	2-16	x	x	x		N/A	N/A	N/A
T-754-SPLP Spoken Language Processing	8					2	2	
T-764-DATA Big Data Management	8	x			x	2	0	
T-814-INNO Creating a Complete Business Plan for a Technical Idea-Entrepreneurship and the Innovation Process (DE)	8	x		x		2	0	
T-786- APDS Applied Data Science	8	x			x	2	1	
T-809-DATA Datamining and Machine Learning (DE)	8	x		x		2	1	
T-810-OPTI Optimization Methods (DE)	8	x			x	2	0	
T-811-PROB-Applied Probability (DE)	8		x		x	2	0	
T-796- DEEP Introduction to Deep learning	6		x	x				x

1 ECTS = 25-30 hours
Each term is divided in to two periods, 12-week period and 3-week period
F = Fall term
S = Spring term
L = Lectures, 1= 2 x 45 min
E = Excersices, 1= 2 x 45 min
L+E = Lectures and exercises combined, taught in 3-week period, approx. 8 hours a day, 5 days a week

Description of Mandatory courses:

T- 519-STOR Theory of Computation

Credits: 6 ECTS
Year: one
Semester: fall
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 as the 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-701-REM4 Research Methodology

Credits: 8 ECTS
Year: one
Semester: spring
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: 12-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-reviewer	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, research papers etc.

T-707- MOVE Modelling and Verification

Credits: 8 ECTS
Year: one
Semester: spring
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: 12-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-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.

T-891-MSTD Master Thesis Defence

Credits: 6 ECTS
Year: two
Semester: 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.

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: 30 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 30 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 i n 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 ch osen area of specialization.
- Apply research methods, techniques, and problem solving approaches from the field of resear ch in which they 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.

T-891-MSTD Master Thesis Defence

Credits: 6 ECTS
Year: two
Semester: 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 objectives and skills:

- 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:

based on the master thesis of the student.

Descriptions of elective courses:

T-498-GAGR Data Analysis

Credits: 6 ECTS

Year: one

Semester: fall semester

Level of course: N/A

Type of course: elective

Prerequisites: T-201-GSKI, Data Structures, T-213-VEFF, Web-Programming

Structure: three-week course

Lecturer: Arnar Leifsson

Description

This course introduces app software development for mobile devices. The concepts studied are applied in a practical group project taking an application through a complete development cycle.

Learning outcomes

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

Knowledge

- Know the fundamentals of app development, including an app's life-cycle.
- Know best app design and implementation practices.
- Know how to program graphical user interfaces and touch screen interactions.
- Know different ways for apps to retrieve, store and share data.
- Know how to program responsive apps using asynchronous flow.

Skills

- Be able to use a selected app software development environment effectively.
- Be able to make interactive apps that handle all aspects of the life-cycle, run gracefully on different sized devices, e.g. smartphones and tablets, and that effectively retrieve, store and share data..
- Be able to work in groups on developing non-trivial apps.

Competence

- Be able to develop robust and responsive non-trivial interactive apps for different sized devices that behave in accordance with relevant standards and guidelines.

Course assessment

First week - 30%

Second week – 30%

Third week – 30%

Video demonstration 10%

Course workload

28 hours lectures

75 hours assignments

Reading Material

Slides from lecturer

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, Algorithms.

Organization of course: twelve-week course

Teacher: Stephan Schiffel

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%

Workload:

54 hours in class (lectures, lab classes), 3 hours exam, 20 hours exam preparation, 25 hours homework assignments, 50 hours programming assignments.

Reading Material:

Artificial Intelligence: A modern Approach by Russel and Norvig.

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 Operation 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.

Workload: 36 hours lectures, 80-100 hours exercises and programming assignments, 20 hours exam preparation, 3 hours exam.

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%

Workload:

54 hours in class (lectures, lab classes), 3 hours exam preparation, 5 hours quizzes, 20 hours homework assignments, 50 hours programming assignments.

Reading Material:

Lecture notes provided by teacher.

T-535-CPSY Cyber Physical Systems

Credits: 6 ECTS

Year: one

Semester: fall term

Type of course: advanced undergraduate course for MSc programmes at DCS.

Necessary Prerequisites: T-315-STY1

Organization of course: twelve-week course

Teacher: Marcel Kyas

Language of teaching: English

Description:

Cyber-physical systems introduces students to the design
and analysis of computational systems that interact with physical processes. Applications of such systems include medical devices and systems, consumer electronics, toys and games, assisted living, traffic control and safety, automotive systems, process control, energy management and conservation, environmental control, aircraft control systems, communications systems, instrumentation, critical infrastructure control (electric power, water resources,
and communications systems for example), robotics and distributed robotics (telepresence, telemedicine), defense systems, manufacturing, and smart structures.

A major theme of this course is on the interplay of practical design with models of systems, including both software components and physical dynamics.
A major emphasis will be on building high confidence systems with real-time
and concurrent behaviours.

Topics include:

- The term embedded system, the main concerns in design, construction,
  and analysis of embedded systems, and the main areas of the field.
- Harvard architecture and the different implementations of it used for common embedded systems.
- Continuous time, ficticios time, discrete time, and logical time.
- Discrete and continuous behaviour, modeled by state machines and differential equations.
- The interface between a digital system and the physical world, analog/digital conversion, digital/analog conversion, Nyquist's theorem, quantization noise.
- Real-Time schedulers: earliest deadline first,
  rate monotonic scheduling, concepts of schedulability, …
- Programming of embedded systems in one of the common languages: Ada, C/C++,
  PLC languages; lab using a robot
- Analyse models in a commonly used tool: Matlab or Python/Sage.
- System security,
  e.g. attestation; security threads from peripherals and sensors, security of embedded devices
- Fault tolerance, redundancy, fail-safety, reliability, availability

Learning outcomes:

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

Knowledge:

- Describe a realtime or hybrid system as a system characterized by a known set of configuratio ns with transitions from one unique configuration (state) to another (state)
- Describe the distinction between systems whose output is only a function of their input (Comb inational) and those with memory/history (Sequential).
- Derive time-series behavior of a state machine from its state machine representation.
- List capabilities and limitation,
  like their uncertainties, of robot systems, including their sensors,
  and the crucial sensor processing that informs those systems, and
  in general terms how analog signals can be reasonably represented by discrete samples and ar ticulate strategies for mitigating these uncertainties.
- Identify physical attacks and countermeasures, attacks on non-
  PC hardware platforms and discuss the concept and importance of trusted path.
- Describe what makes a system a real-
  time system, explain the presence of and describe the characteristics of latency in real-
  time systems, and summarize special concerns that real-time systems present, including risk,
  and how these concerns are addressed.
- Explain the relevance of the terms fault tolerance, reliability,
  and availability, outline the range of methods for implementing fault tolerance,
  and explain how a system can continue functioning after a fault occurs.

Skills:

- Program a robot to accomplish simple tasks using deliberative, reactive,
  and/or hybrid control architectures.
- Integrate sensors, actuators, and software into a robot designed to undertake some task.

Competence:

- Design and implement an industrial application on a given platform (e.g., using Raspberry Pi).

Assessment:

Programming assignments	30%
Assignments	40%
Oral exam	30%
Total	100%

Workload:

48 hours lecture, 24 hours lac classes, 36 hours self-study, 30 hours assignments. 30 hours programming project, 12 hours exam preparation and exam.

Reading material:

Peter Marwedel. Embedded System Desing: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things. 4th ed. Springer, 2021.

Derek Molloy. Exploring Rasberry Pi: Interfacing to the real world with Embedded Linux. Wiley, 2016.

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:

The course defines the function and structure of a compiler. Lexical and syntax analysis is discussed in detail, including use of regular expressions, finite automata, and top-down and bottom-up parsing approaches. Semantic analysis and (intermediate) code generation is also covered in some detail. The course also introduces tools for automatically generating lexers and parsers from formal specifications, both their use and underlying algorithms. Hands-on construction of a compiler/interpreter is 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 formal grammar for describing programming languages and understand how to implement to-down and bottom-up parsing methods.
- Be able to generate (intermediate) code from an abstract-syntex tree, e.g. for virtual machines.
- Be able to use prevalent software tools that automatically generate lexers and parsers from formal specifications.

Competences:

- Be able to design and build a simple compiler.

Assessment:

Exams (total)	40%
Homework (written)	5%

Labs 10%

Project (programming a compiler) 45%

Workload:

30 hours lectures, 5 hours exams, 30 hours lecture preparation, 15 hours exam preparation, 10 hours written homework, 15 hours labs, 60 hours project (programming an interpreter/compiler).

Reading Material:

Introduction to Compiler Design, Torben Ægidius Mogensen , Preface/Chapters 1-6 (161 pages).

Lecture Notes: Introduction to Compiler Construction, Yngvi Björnsson (36 pages)

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, Computer Graphics or Game Engine Design.

Organization of course: twelve-week course

Teacher: Steingerður Lóa Gunnarsdóttir, Páll Ragnar Pálsson, Þorlákur Sveinsson Lyngmo.

Language of teaching: English

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.

Workload:

Group work and presentations throughout the course.

18th of October
*Subject to change

Reading Material:

Lecture notes provided by teacher.

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: Steingerður Lóa Gunnarsdóttir

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

18th of October
*Subject to change

Workload:

39 hours classes, 19 hours lab, 110 hours estimated for project work.

Reading Material:
slides from lectures.

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 %

Workload: 36 hours lectures, 80-100 hours exercises, programming assignments and labs, 20 hours exam preparation, 3 hours exam.

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 get technical hands-on experience in C++ game development, including the use and development of supporting tool pipelines. The course includes visiting talks and Q&A from industry veterans.

Learning outcomes:

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

Knowledge:

- 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

Skills:

- 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.

Competence:

- 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%

Workload:

36 hours attending lectures

20 hours lecture preparation and study

14 hours lab work

16 hours problem set work

40 hours project work

24 hours final exam preparation.

Reading Material:

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

T-423-ENOP Engineering Optimization (DE)

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.

Workload: 36 hours lectures, 80-100 hours exercises and programming assignments, 20 hours exam preparation, 3 hours exam.

Reading Material:

Lecture notes provided by teacher.

T-424-SLEE Sleep (DE)

Credits: 6 ECTS

Year: one

Semester: Spring

Level of course: advanced elective undergraduate course for all MSc programmes.

Type of course: elective

Prerequisites: none

Structure: three-week course, on-site

Lecturer: Erna Sif Arnardóttir

Description

In the course the following topics will be addressed: early sleep research, the neural underpinnings of sleep and wakefulness, comparative sleep research, the role of sleep in learning and memory, sleep deprivation and the function of sleep active compounds. A special emphasis will be placed on understanding the application of information accumulated by sleep researchers.

Learning outcomes

On completion of the course, students should be able to:

- Be knowledgeable on the current understanding of sleep
- Understand the bottlenecks that preclude full understanding of sleep
- Know the most active fields within sleep research
- Understand the most common research methods employed by sleep researchers

Course assessment

Participation in classes 20%

Report of practical assignments 30%

Exam 50%

Course workload

36h of lectures
24h exercise classes
90h projects and reading
Total 150 hours

Reading Material

Lectures and slides from teachers.

T-502-HERM Simulation (DE)

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%

Workload: 36 hours lectures, 80-100 hours exercises and programming assignments, 20 hours exam preparation, 3 hours exam.

Reading Material:

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

T-717-SPST Speech Synthesis

Credits: 8 ECTS

Year: one

Semester: spring term

Type of course: elective course for all Master programmes at DCS.

Necessary Prerequisites: Good programming skills in Python are a requirement for the course.

Organization of course: three-week course

Teacher: Jón Guðnason and Atli Þor Sigurgeirsson

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 acoustic features
- Methods for transforming linguistic 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%

Workload:

15 hours lecture, 15 hours project session, 30 hours project and exam preparation

60 hours project work, 60 hours home work

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 centres on the issues of intelligence architecture, system autonomy, real-time 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%

Workload:

61 hours lectures, 80-125 hours exercises and programming assignments, 20 hours exam preparation, 3 hours exam.

Reading material:

Lecture notes provided by teacher. A total of about 300 pages of hand-picked and matched research papers in AI and philosophy.

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:

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

Knowledge:

- Know what constitutes a virtual environment, why they have been created throughout history and how they are used today.
- Know the difference between presence and immersion, and understand how these may be measured.
- Know what an avatar is and understand the issues that relate to level of control.
- Be familiar with the roles of characters in virtual environments and the common ways to make them autonomous and to animate them.

Skills:

- Be able to apply a range of perceptual depth cues to create the illusion of depth in a 2D image.
- Be able to create, build and share interactive virtual environments in Unity 3D, using scripting in c#, scene editing, 3d models, terrain editor, lighting techniques, materials, texturing, physics, animation, heads-up-display and shaders.

Competence:

- Understand how humans construct a mental image of their environment using visual cues.
- Understand and be able to apply the principles of effective action in virtual environments, including concepts such as flow, implicit constraints, explicit constraints and contextual action.
- Understand and be able to apply interdisciplinary techniques and concepts from classic animation, procedural rhetoric theory, theatre, comparative mythology, and sociology to increase the effectiveness of virtual environments.
- Be able to think critically about virtual environments as a novel interface paradigm for a range of problems.
- Be able to design and implement a virtual environment that effectively addresses a problem, and evaluate the results.
- Be able to conduct further research into the area of virtual environments.

Assignments:

Participation: 10% (Participation in discussion activities, lab participation)

Programming assignments: 20% (two 10% each)

Final Project: 40% (of which proposal presentation is 5% and final report is 5%)

Final Written Exam: 30% (from all materials, plus virtual environment design essay)

Workload:

22 hours attending lectures

6 hours in-class discussion

12 hours discussion preparation

24 hours lab work

30 hours programming assignments

60 hours final project work

26 hours final exam preparation.

Reading material:

A large number of articles and book chapters, made available as PDFs. Students are also expected to read technical documentation and various tutorials for the Unity 3D engine.

T-725-MALV Natural Language Processing

Credits: 8 ECTS

Year: one

Semester: fall

Type of course: advanced mandatory undergraduate course for MSc in Artificial Intelligence and Language Technology. Elective course for other master programmes at DCS.

Necessary Prerequisites: none

Organization of course: twelve-week course

Teacher: Hrafn Loftsson and Hannes H. Vilhjálmsson

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%

Workloads:

Lectures and preparation 6 hours per week * 12= 72 hours, lab + lab projects 3 hours per week= 36 hours, 3 homework assignments= 24 hours, Term project 60 hours, Final exam and preparation 12 hours. Total of 204 hours.

Reading Material:

"Speech and Language Processing", by Jurafsky & Martin.

"Natural Language Processing with Python", by Bird, Klein & Loper.

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%

Workload: 61 hours lectures, 80-125 hours exercises and programming assignments, 20 hours exam preparation, 3 hours exam.

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:

Knowledge:

- Explain the concepts and components of embedded systems and IoT devices

Skills:

- 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

Competence:

- 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%

Workload: 36 hours lectures, 80-100 hours exercises and programming assignments, 20 hours exam preparation, 3 hours exam.

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-738- VIRH Virtual Humans

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.

Organization of course: twelve-week course

Teacher: Hannes Högni Vilhjálmsson

Description:

This is a comprehensive course in both the theory and practice of Virtual Humans (VHs). Virtual Humans are digital graphical simulations of real humans that aim to produce life-like appearance and behavior, often in a social context. Application areas include virtual actors for movies, virtual patients for medical training, virtual tutors for education, non-player characters in games and conversational agents as natural interfaces for intelligent systems. Simulating humans is an inherently interdisciplinary endeavor that brings together fields that study humans, such as sociology and psychology, as well as technical fields, such as computer graphics and language technology. The course will introduce and explore this broad and emerging topic through literature discussions, state-of-the-art technical demonstrations and hands-on lab exercises. Students will get a chance to work on their own interactive Virtual Human prototype for their independent final project.

Learning outcomes:

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

Knowledge

- Know what virtual humans are, why they are being constructed and generally how they are created.
- Be familiar with a range of common techniques for modeling and animating virtual humans, as well as giving them a life of their own through simple decision mechanisms.
- Be familiar with models of human behavior and social function that have been implemented in virtual humans, as well as some of the cognitive architectures that have been built to control them.

Skills

- Be able to build interactive virtual humans using the Unity 3D game engine.
- Be able to use 3rd party Unity 3D plug ins such as Salsa for lip synching and Node Canvas for state machines and behavior trees.
- Be able to use classic virtual human stand-alone tools to create necessary assets from scratch, including 3D models, animation, textures and speech.

Competence

- Be able to think critically about applications of virtual humans in real-life settings, including ethical considerations.

- Understand how to use an interdisciplinary approach to designing and developing virtual humans, and to evaluate their effectiveness.

Assignments:

Participation: 10% (Participation in discussion activities, lab participation)

Programming assignments: 20% (two 10% each)

Final Project: 40% (of which proposal presentation is 5% and final report is 5%)

Final Written Exam: 30% (from all materials, plus virtual environment design essay)

Workload:

22 hours attending lectures

6 hours in-class discussion

12 hours discussion preparation

24 hours lab work

30 hours programming assignments

60 hours final project work

26 hours final exam preparation.

Reading material:

A large number of articles and book chapters, made available as PDFs. Students are also expected to read technical documentation and various tutorials for the Unity 3D engine.

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

Credits: 6 ECTS

Year: one

Semester: spring

Type of course: final undergraduate elective course for all Master programmes at DCS.

Necessary Prerequisites: Python/C++ programming, Operating systems, Computer networks.

Organization of course: twelve -week course

Teacher: Jacqueline Clare Mallett

Language of teaching: English

Description:

This course examines the theory and practice 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 informat ion 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 paper	50%
Continuous evaluation with weekly laboratories, quizzes and exercises	50%
Total	100%

Workload: 36 hours lecture, 36 hours weekly laboratory, 12 hours weekly quiz, 12 hours homework, 60 hours term paper.

Reading material:

Case studies and papers are provided for reading.
Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers T. J. O´Connor

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: T-501-FMAL Programming Languages

Organization of course: three-week course

Teacher: Tarmo Uustalu

Language of teaching: English

Description:

Semantics is about describing the meaning of programs formally, so they become amenable for rigorous analysis and trustworthy machine processing as data objects. Language processors, program analyzers, optimizers, verification tools are all based on semantics. The course covers the basics of operational and denotational semantics and some applications (compilation, some program analyses, e.g., secure information flow, their correctness).

Learning outcomes:

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

- To understand the value of formal descriptions of programming language semantics
- To know the distinguishing characteristics of main semantic description styles (big-step/small-step operational semantics, denotational semantics)
- To be able to read formal semantic descriptions, to devise such descriptions for simple constructs
- To understand the value of semantics-based reasoning
- To be able to read semantics-based proofs about programs, to construct simpler such proofs
- To be able to read semantics-based proofs of properties of compilers, program analyses and transformations

Assessment:

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

Workloads:

12..15 x 3 hrs combined lectures/practical sessions, independent study, work on assignments

Reading material:

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

T-747- RELE Reinforcement Learning

Credits: 8ECTS
Year: one
Semester: spring term
Type of course: elective course for all Master programmes at DCS.
Necessary Prerequisites: Algorithms, Programming, Linear algebra, Statistic, Calculus
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 able 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%

Workload:

54 hours in class (lectures, discussions, presentations), 3 hours exam, 20 hours exam preparation, 20 hours reading papers and preparing a presentation, 100 hours working on assignments and projects.

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.

Assessment:

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

Workload:

30 hours per ECTS defined for each individual project work.

Reading Material:

Defined for each project separately.

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%

Workload: 54 hours in class (lectures, discussions, presentations), 3 hours exam, 20 hours exam preparation, 20 hours reading papers and preparing a presentation, 100 hours working on assignments and projects.

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:

Throughout history, the amount of data produced and stored has been growing exponentially, but it is only in recent years that this exponential growth has really come to the fore. According to Wikipedia, big data is an "all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process them using traditional data processing applications." The management and analysis of such data sets lead to significant technical, administrative and ethical challenges, but also significant opportunities.

Learning outcomes:

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

Knowledge:

- 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.
- 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.

Skills:

- Apply a methodology for big data management projects.
- Use an automatically distributed computing framework to perform basic data analysis.

Competence:

- 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

18th of October
*Subject to change

Assessment:

20% of grade is based on in-class presentation of scientific papers (peer-review is used and influences grading).

30% of grade is based on group assignments and & another 30% is an individual project of student's choice. Both are graded by the teacher and the focus is 80% of the reports. Detailed feedback is given on the context, language and structure of the reports as the primary goal is to teach and prepare the student for academic work.
10% of grade is in-class participation.

Workload:

36 hours of lectures, 45 hours for reading the course book, 15 hours preparation for in class presentation & rehearsal with teachers, 36 hours for reading papers presented by other students, 48-52 hours for group projects, 46 hours for the individual project.

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. In additions the students will read various scientific research papers, 8 obligatory, 7 chosen by students (but all must read) and potentially 2-3 papers the student finds on his own.

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

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.

Workload:

54 hours in class (lectures, discussions, presentations), 3 hours exam, 20 hours exam preparation, 20 hours reading papers and preparing a presentation, 100 hours working on assignments and projects.

Reading Material:

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

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%

Workload:

24 hours lectures, 12 hours labs, 24 hours homework, 60 hours work on assignments, 30 hours final project.

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 (DE)

Credits: 6 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:

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 real life 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:

- 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.

18th of October
*Subject to change

Assessment:

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

Workload: 24 hours lectures, 12 hours labs, 24 hours homework, 60 hours work on assignments, 30 hours final project.

Reading material:

Christopher M. Bishop; “Pattern Recognition and Machine Learning”

Duda, Hark, Stork; “Pattern Classification”

T-810-OPTI Optimization Methods (DE)

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%

Workload: 24 hours lectures, 12 hours labs, 24 hours homework, 60 hours work on assignments, 30 hours final project.

Reading material:

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

T-811-PROB-Applied Probability (DE)

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%

Workload: 54 hours in class (lectures, discussions, presentations), 3 hours exam, 20 hours exam preparation, 20 hours reading papers and preparing a presentation, 100 hours working on assignments and projects.

Reading material:

Lecture notes which will be sufficient for successfully completing 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-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:
- Independently propose a research project,
  plan its execution, undertake its development, evaluate its outcome and report on its results i n 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 ch osen 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.
- 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

Workload:

- 1350 hours

Reading material:

- Defined for each thesis separately.