Contents

20.6.2017

Study Programs in Computer Science

B.Sc. in Computer Science– general

To complete a B.Sc. in general computer science, students need to complete 180 ECTS, of which 120 ECTS are mandatory. Each course is 6 ECTS, except for the final project which is 12 ECTS. An example of a study plan can be seen in the table; however, courses can be arranged differently as long as rules of prerequisites are followed.

1. semester – fall term	2. semester – spring term
T-111-PROG – Programming T-114-VERK – Problem Solving T-107-TOLH – Computer Architecture T-117-STR1 – Discrete Mathematics I T-113-VLN1 – Practical Project 1 (3. Week course)	T-201-GSKI – Data Structures T-419-STR2 – Discrete Mathematics II T-213-VEFF – Web-Programming T-216-GHOH – Software Requirements and Design T-220-VLN2 – Practical Project 2 (3. Week course)
3. semester – fall term	4. semester – spring term
T-317-CAST – Calculus and Statistics T-301-REIR - Algorithms T-303-HUGB – Software Engineering T-202-GAG1 - Databases Elective Course (3. Week course)	T-501-FMAL – Programming Languages T-444-USTY – Fundamentals of Operating Systems or T-215-STY1 – Operating Systems Elective Course Elective Course X-204-STOF – Entrepreneurship and Starting New Ventures (3. Week course)
5. semester – fall term	6. semester – spring term
T-409-TSAM – Computer Networks Elective Course Elective Course Elective Course Elective Course (3. Week course)	Elective Course Elective Course Elective Course T-404-LOKA – Final Project (15. Week course)

Students that are taking Computer Science – general have the option of choosing between different emphasise lines, those are:

Artificial Intelligence

Due to prerequisite rules, we recommend that students take the courses T-101-STA1-Calculus 1, T-302-TOLF Statistics 1 to replace the course T-317-CAST Calculus and Statistics.

Mandatory courses:

- T-622-ARTI Artificial Intelligence
  - ✓ Prerequisite: T-301-REIR Algorithms
- T-504-ITML Introduction to Machine Learning
  - ✓ Prerequisites: T-301-REIR Algorithms, T-419-STR2 Discrete Mathematics II, T-317-CAST Calculus and Statistics or T-301-REIR Algorithms, T-103-STST Discrete Mathematics for Engineering students, T-101-STA1 Calculus 1

Elective courses – choose from at least three of the following courses:

- I-707-VGBI Business Intelligence
  - ✓ Prerequisites: None
- T-211-LINA Linear Algebra
  - ✓ Prerequisites: None
- T-403-ADGE Operation Research
  - ✓ Prerequisites: T-101-STA1 Calculus 1, T-302-TOLF Statistics 1
- T-502-HERM Simulation
  - ✓ Prerequisites: T-101-STA1 Calculus 1, T-302-TOLF Statistics 1, T-402-TOLF Statistics 2
- T-637-GEDE Game Engine Architecture
  - ✓ Prerequisites: T-301-REIR Algorithms, T-511-TGRA Computer Graphics
- E-409 LEIK Game Theory
  - ✓ Prerequisites: T-111-PROG Programming, T-103-STST Discrete Mathematics for Engineering students or T-101-STA1 Calculus 1 or T-117-STR1 Discrete Mathematics 1
- T-624- CGDD Computer Game Design & Development
  - ✓ Prerequisites: T-301-REIR Algorithms
- T-634- AGDD Advanced Game Design & Development
  - ✓ Prerequisites: T-624- CGDD Computer Game Design & Development

Computer Games Development

Some of the courses in this emphasize line have a limited number of students that can participate, for that reason students need to be signed up for the emphasize line. If students do not show good progress, they might be disqualified from the line.

Mandatory courses:

- T-511-TGRA Computer Graphics
- T-624- CGDD Computer Game Design & Development
- T-637-GEDE Game Engine Architecture
- T-414-AFLV Effective Programming and Problem Solving
  OR
  T-528-HLUT Object Oriented Programming in C++
- T-622-ARTI Artificial Intelligence
  OR
  T-634- AGDD Advanced Game Design & Development

Web and interface development

You need to take five courses to fulfil the emphasize line:

- T-427-WEPO Web Programming II
- T-514-VEFT Web Services
- T-542-HGOP Introduction to quality management and testing
- T-515-NOTH User-Centered Software Development
  AND OR
  T-636-SMAT Human Computer Interaction

Electives – one of the courses can be chosen:

- T-302-HONN Software Design and Implementation
- T-488-MAPP Mobile App Development
- T-416-VIFG Data Acquisition and Visualization
- T-611-NYTI New Technology

Students also have the opportunity to take 60 ECTS elective courses of their own choice. Students can choose elective courses within the department (i.e. School of Computer Science and technical courses in the School of Science and Engineering). If the choice is outside of the department, the following applies:

- Students who have completed comprehensive studies, at least 60 ECTS in higher education in other departments, can apply to get evaluated up to 60 ECTS.
- If there is no comprehensive study involved students can apply for an assessment of up to 36 ECTS outside of the department.
- Please note that the courses Applied Mathematics I, Information Technology and Applied Statistics 1 in the School of Business cannot be taken due to the overlap of core computer science courses. Note that you cannot get credit for both Data Processing and Databases because of overlap. Note that you cannot get credits for both Operating Systems and Fundamentals of Operating Systems because of overlap.

Additional Notes:

- Students who enrolled in the fall of 2013 do not need to take Computer Networks. It is sufficient to take Operating Systems or Operating Systems and Networks.
- Students who enrolled before fall 2016 are not required to take the course Entrepreneurship and Starting New Ventures.

Tips:

Instead of taking T-317-CAST – Calculus and Statistics students may take T-101-STA1 – Mathematics I and T-302-TOLF – Statistics I.

It is also possible to take T-103-STST–Discrete Mathematics for Engineering and T-211-Linear Algebra courses instead of T-117-STRI-Discrete Mathematics I and T-419-STR2 Discrete Mathematics II.

This choice offers more opportunities to take interdisciplinary courses in Engineering as an alternative. However, the student must meet the prerequisites from the rule of mathematical preparation of secondary school and have taken the course T-101-STAI Mathematics I.

B.Sc. in Computer Science – research based

To complete a B.Sc. in computer science research based students must complete 180 ECTS, of which 132 ECTS are mandatory, 12 ECTS in the capstone selection and 36 ECTS free choice. Of these 36 ECTS, 12 must be within the department (i.e. School of Computer Science and technical courses in the School of Science and Engineering) but 24 ECTS can be outside of these departments.

Each course is 6 ECTS, except the final project which is 12 ECTS. An example of a study plan can be seen in the table, however courses can be arranged differently as long as rules of prerequisites are followed.

1. semester – fall term	2. semester – spring term
T-111-PROG - Programming T-103-STST – Discrete Mathematics for Engineers T-101-STA1 – Mathematics I T-114-VERK – Problem Solving T-113-VLN1 – Practical Project 1 (3. Week course)	T-201-GSKI – Data Structures T-211-LINA- Linear Algebra T-213-VEFF – Web-programming T-216-GHOH – Software Requirements and Design T-220-VLN2 – Practical Project 2 (3. Week course)
3. semester – fall term	4. semester – spring term
T-202-GAG1 - Databases T-301-REIR – Algorithms T-107-TOLH – Computer Architecture T-303-HUGB – Software Engineering Elective Course (3. Week course)	T-498-GAGR- Data Analysis T-501-FMAL – Programming Languages T-215-STY1 – Operating Systems Elective Course X-204-STOF – Entrepreneurship and Starting New Ventures (3. Week course)
5. semester – fall term	6. semester – spring term
T-519-STOR – Theory of Computation or T-409-TSAM – Computer Networks Capstone course Elective Course Elective Course (3. Week course)	Capstone course Elective Course Elective Course T-404-LOKA – Final Project (15 week course)

Students that are taking Computer Science – research based have the option of choosing between different emphasise lines, those are:

Artificial Intelligence

Due to prerequisite rules, we recommend that students take the courses T-101-STA1-Calculus 1, T-302-TOLF Statistics 1 to replace the course T-317-CAST Calculus and Statistics.

Mandatory courses:

- T-622-ARTI Artificial Intelligence
  - ✓ Prerequisite: T-301-REIR Algorithms
- T-504-ITML Introduction to Machine Learning
  - ✓ Prerequisites: T-301-REIR Algorithms, T-419-STR2 Discrete Mathematics II, T-317-CAST Calculus and Statistics or T-301-REIR Algorithms, T-103-STST Discrete Mathematics for Engineering students, T-101-STA1 Calculus 1

Elective courses – choose from at least three of the following courses:

- I-707-VGBI Business Intelligence
  - ✓ Prerequisites: None
- T-211-LINA Linear Algebra
  - ✓ Prerequisites: None
- T-403-ADGE Operation Research
  - ✓ Prerequisites: T-101-STA1 Calculus 1, T-302-TOLF Statistics 1
- T-502-HERM Simulation
  - ✓ Prerequisites: T-101-STA1 Calculus 1, T-302-TOLF Statistics 1, T-402-TOLF Statistics 2
- T-637-GEDE Game Engine Architecture
  - ✓ Prerequisites: T-301-REIR Algorithms, T-511-TGRA Computer Graphics
- E-409 LEIK Game Theory
  - ✓ Prerequisites: T-111-PROG Programming, T-103-STST Discrete Mathematics for Engineering students or T-101-STA1 Calculus 1 or T-117-STR1 Discrete Mathematics 1
- T-624- CGDD Computer Game Design & Development
  - ✓ Prerequisites: T-301-REIR Algorithms
- T-634- AGDD Advanced Game Design & Development
  - ✓ Prerequisites: T-624- CGDD Computer Game Design & Development

Computer Games Development

Some of the courses in this emphasize line have a limited number of students that can participate, for that reason students need to be signed up for the emphasize line. If students do not show good progress, they might be disqualified from the line.

Mandatory courses:

- T-511-TGRA Computer Graphics
- T-624- CGDD Computer Game Design & Development
- T-637-GEDE Game Engine Architecture
- T-414-AFLV Effective Programming and Problem Solving
  OR
  T-528-HLUT Object Oriented Programming in C++
- T-622-ARTI Artificial Intelligence
  OR
  T-634- AGDD Advanced Game Design & Development

20.6.2017

Web and interface development

You need to take five courses to fulfil the emphasize line:

- T-427-WEPO Web Programming II
- T-514-VEFT Web Services
- T-542-HGOP Introduction to quality management and testing
- T-515-NOTH User-Centered Software Development
  AND OR
  T-636-SMAT Human Computer Interaction

Electives – one of the courses can be chosen:

- T-302-HONN Software Design and Implementation
- T-488-MAPP Mobile App Development
- T-416-VIFG Data Acquisition and Visualization
- T-611-NYTI New Technology

Additional Notes:

- Please note that the courses Applied Mathematics, Applied Information and Applied Statistics 1 (in School of Business) cannot be taken due to the overlap with core courses of computer science. Note that you cannot get credit for both Data Processing and Databases because of overlap. Note that you cannot get credits for both Operating Systems and Fundamentals of Operating Systems because of overlap. It is also not possible to get both Statistics 1 and Data Analysis evaluated.
- Students who enrolled in the fall 2016 are not required to take the course Entrepreneurship and Starting New Ventures.

The 12 ECTS capstone selection can be chosen from the following courses:

School of Computer Science:

- T-622-ARTI Artificial Intelligence
- T-707 Modeling and Verification
- T-725-MALV Natural Language Processing
- T-724-SETA Semantics
- T-521-RELE Reinforcement Learning
- T-723-VIEN Virtual Environments
- T-605-TGRA Computer Graphics
- T-603-THYD Compilers
- T-445-GRTH Graph Theory
- T-624-CGDD Computer Game Design & Development
- T-504-ITML Introduction to Machine Learning
- T-419-CADP Concurrent and Distributed Programming
- T-513-CRNU Cryptography and Number theory

School of Science and Engineering:

- T-403-ADGE Operations Research

- T-306-MERK Signals and Systems
- T-301-MATH Mathematics III
- T-106-LIFV Molecular Cell Biology
- T-402-TOLF Statistics II
- T-406-TOLU Numerical Analysis
- T-865-MACH Machine Learning
- T-102-EDL1 Physics I

B.Sc. in Computer Science with a minor in Business

To complete a B.Sc. in Computer Science with a minor in Business students must complete 180 ECTS, of which 174 ECTS are mandatory. Each course is 6 ECTS, except the final project which is 12 ECTS. An example of a study plan can be seen in the table; however, courses can be arranged differently as long as rules of prerequisites are followed.

1. semester – fall term	2. semester – spring term
T-111-PROG – Programming T-114-VERK – Problem Solving T-107-TOLH – Computer Architecture T-117-STR1 – Discrete Mathematics I T-113-VLN1 – Practical Project 1 (3. Week course)	T-201-GSKI – Data Structures T-419-STR2 – Discrete Mathematics II T-213-VEFF – Web-Programming T-216-GHOH – Software Requirements and Design T-220-VLN2 – Practical Project 2 (3. Week course)
3. semester – fall term	4. semester – spring term
T-317-CAST – Calculus and Statics T-301-REIR - Algorithms T-303-HUGB – Software Engineering V-108-REHA - Accounting T-316-GAVI – Data Processing (3. Week course)	V-202-REGR – Business Process Analysis T-501-FMAL – Programming Languages V-201-RHAG - Microeconomics T T-444-USTY – Fundamentals of Operating Systems or T-215-STY1 – Operating Systems X-204-STOF – Entrepreneurship and Starting New Ventures (3. Week course)
5. semester – fall term	6. semester – spring term
V-307-GARS – Design and analysis of financial statements I-406-IERP – Introduction to ERP systems V-107-FJAR – Corporate Finance T-409-TSAM – Computer Networks T-168-ITST IT Strategy (3. Week course)	V-311-OPMA – Operations Management I-707-VGBI – Business Intelligence Elective Course T-404-LOKA – Final Project (15. Week course)

Students take 6 ECTS of their choice within or outside of the departments.

- Please note that the courses Applied Mathematics, Applied Information and Applied Statistics 1 (in School of Business) and Digital Technology (Science and Engineering) cannot be taken in the selection of the mandatory courses as they overlap the computer science courses. Note that you cannot get credit for both Data Processing and Databases because of overlap. Note that you cannot get credits for both Operating Systems and Fundamentals of Operating Systems because of overlap.

B.Sc. in Software Engineering

To complete a B.Sc. in Software Engineering, students need to complete 180 ECTS, of which 156 ECTS are mandatory. Each course is 6 ECTS, except the final project which is 12 ECTS. An example of a

study plan can be seen in the table, however courses can be arranged differently as long as rules of prerequisites are followed.

1. semester – fall term	2. semester – spring term
T-111-PROG - Programming T-103-STST – Discrete Mathematics for Engineering T-101-STA1 – Mathematics I T-102-EDL1 – Physics I T-113-VLN1 – Practical Project 1 (3. Week course)	T-201-GSKI – Data Structures T-211-LINA- Linear Algebra T-201-STA2 – Mathematics II T-216-GHOH - Software Requirements and Design X-204-STOF – Entrepreneurship and Starting New Ventures (3. Week course)
3. semester – fall term	4. semester – spring term
T-202-GAG1 - Databases T-301-REIR – Algorithms T-107-TOLH – Computer Architecture T-303-HUGB – Software Engineering Elective Course (3. Week course)	T-213-VEFF – Web programming T-501-FMAL – Programming Languages T-215-STY1 – Operating Systems T-631-SOE2 – Software Engineering II - Testing T-220-VLN2 – Practical Project 2 (3. Week course)
5. semester – fall term	6. semester – spring term
T-519-STOR – Theory of Computation or T-603-THYD - Compilers T-409-TSAM – Computer Networks T-302-TOLF – Statistics I Elective Course Elective Course (3. Week course)	T-403-ADGE – Operation Research T-415-STAN – Statistical Analysis for Software Engineering Students Elective Course T-404-LOKA – Final Project (15 week course)

Students take 24 ECTS of elective courses of their own choice. Students can choose elective courses within the department (i.e. Computer Science and Science and Engineering). If the choice is outside of the department the following applies:

- If the choice is outside of the department students can apply for an assessment of up to 12 ECTS.
- Please note that the courses Applied Mathematics, Applied Information and Applied Statistics 1 (in School of Business) cannot be taken in the choice of the overlap with a core of software engineering. Note that you cannot get credit for both Data Processing and Databases because of overlap. Note that you cannot get credits for both Operating Systems and Fundamentals of Operating Systems because of overlap.

Additional Notes:

- Students who enrolled for the autumn of 2013 do not need to take Computer Networks, it is sufficient to take Operating Systems or Operating Systems and Graph Theory.
- Students who enrolled for the autumn of 2013 are not required to take the course Final Project.
- Students who enrolled for fall 2016 are not required to take the course Entrepreneurship and Starting New Ventures.

B.Sc. in Discrete Mathematics and Computer Science

To complete a B.Sc. in Discrete Mathematics and Computer Science students must complete 180 ECTS, of which 144 ECTS are mandatory, 18 ECTS are from capstone selection and 18 ECTS are elective credits. Each course is 6 ECTS, except the final project which is 12 ECTS. An example of a

study plan can be seen in the table, however courses can be arranged differently as long as rules of prerequisites are followed.

1. semester – fall term	2. semester – spring term
T-111-PROG - Programming T-103-STST – Discrete Mathematics for Engineering T-101-STA1 – Mathematics I T-114-VERK – Problem Solving T-113-VLN1 – Practical Project 1 (3. Week course)	T-201-GSKI – Data Structures T-211-LINA- Linear Algebra T-201-STA2 – Mathematics II T-218-ALCO - Algebra and Combinatorics T-219-REMO– Real-time Models (3. Week course)
3. semester – fall term	4. semester – spring term
T-202-GAG1 - Databases T-301-REIR – Algorithms T-107-TOLH – Computer Architecture T-302-TOLF - Statistics 1 E-402-STFO – Mathematical Programming (3. week)	T-501-FMAL – Programming Languages T-604-HGRE – Design and Analysis of Algorithms T-215-STY1 – Operating Systems or Elective Course Capstone Selection Course X-204-STOF – Entrepreneurship and Starting New Ventures
5. semester – fall term	6. semester – spring term
T-519-STOR – Theory of Computation T-409-TSAM – Computer Networks or Elective Course T-513-CRNU – Cryptography and Number theory Capstone Selection Course Elective Course (3. Week course)	T-505-ROKF – Logic in Computer Science Elective Course Capstone Selection Course T-404-LOKA – Final Project (15 week course)

Additional Notes:

- Students are required to take either Operating Systems or Computer Networks.
- Design and Analysis of Algorithms can be exchanged with Logic in Computer Science in the 4th and 6th semester.
- Algebra and Combinatorics is in rotation with Graph Theory
- Please note that the courses Applied Mathematics, Information Technology and Applied Statistics 1 (in School of Business) cannot be taken with a core of Software Engineering. Note that you cannot get both Data Processing and Databases evaluated because of overlap.
- Students who enrolled before fall 2016 are not required to take the course Entrepreneurship and Starting New Ventures

The 18 ECTS capstone selection can be chosen from the following courses:

School of Computer Science:

- T-622-ARTI Artificial Intelligence
- T-707 Modeling and Verification
- T-725-MALV Natural Language Processing
- T-724-SETA Semantics
- T-521-RELE Reinforcement Learning
- T-723-VIEN Virtual Environments
- T-605-TGRA Computer Graphics
- T-603-THYD Compilers

- T-445-GRTH Graph Theory
- T-624-CGDD Computer Game Design & Development
- T-504-ITML Introduction to Machine Learning
- T-419-CADP Concurrent and Distributed Programming

School of Science and Engineering:

- T-403-ADGE Operations Research
- T-306-MERK Signals and Systems
- T-301-MATH Mathematics III
- T-106-LIFV Molecular Cell Biology
- T-402-TOLF Statistics II
- T-406-TOLU Numerical Analysis
- T-865-MACH Machine Learning
- T-102-EDL1 Physics I

Diploma in Computer Science

To complete a Diploma in Computer Science students must complete 120 ECTS, of which 96 ECTS are from mandatory courses. Each course is 6 ECTS, except the final project which is 12 ECTS. An example of a study plan can be seen in the table, however courses can be arranged differently as long as rules of prerequisites are followed.

1. semester – fall term	2. semester – spring term
T-111-PROG - Programming T-114-VERK – Problem Solving T-107-TOLH – Computer Architecture T-117-STR1 – Discrete Mathematics I T-113-VLN1 – Practical Project 1 (3. Week course)	T-201-GSKI – Data Structures T-213-VEFF – Web programing T-216-GHOH – Software Requirements and Design Elective Course T-220-VLN2 – Practical Project 2 (3. Week course)
3. semester – fall term	4. semester – spring term
T-301-REIR - Algorithms T-303-HUGB – Software Engineering T-202-GAG1 – Databases T-409-TSAM – Computer Networks Elective Course (3. Week course)	T-444-USTY – Fundamentals of Operating Systems or T-215-STY1 – Operating Systems Elective Course Elective Course T-404-LOKA – Final Project (15. weeks)

Students take 24 ECTS electives. Students can choose elective courses within the department (i.e. Computer Science and Science and Engineering). If the course is outside of the department the following applies:

- In the case that the course is outside of the department students can apply for an assessment of up to 12 ECTS.
- Please note that the courses Information Technology (in School of Business) cannot be taken as a course with computer science due to overlap. Note that you cannot get credit for both Data Processing and Databases because of overlap. Note that you cannot get credits for both Operating Systems and Fundamentals of Operating Systems because of overlap.

Additional Notes:

- Students who enrolled before the autumn of 2013 do not need to take Computer Networks, it is sufficient to take Operating Systems or Operating Systems and Networks.

20.6.2017

Diploma in System Administration

To complete a diploma in Systems Administration students must complete 120 ECTS, of which 90 ECTS are mandatory. Each course is 6 ECTS, except the final project which is 12 ECTS. An example of a study plan can be seen in the table, however courses can be arranged differently as long as rules of prerequisites are followed.

1. semester- fall term	2. semester – spring term
T-111-PROG - Programming T-114-VERK – Problem Solving T-107-TOLH – Computer Architecture T-117-STR1 – Discrete Mathematics I T-113-VLN1 – Practical Project 1 (3. Week course)	T-201-GSKI – Data Structures T-213-VEFF – Web programming T-216-GHOH – Software Requirements and Design T-239-KERF – System Administration 1 T-429-UPVE – Computer Setup (3. Week course)
3. semester – fall term	4. semester - spring
T-202-GAG1 – Databases T-409-TSAM – Computer Networks Elective Course Elective Course Elective Course (3. Week course)	T-215-STY1 – Operating Systems Elective Course Elective Course T-431-HANE Practical Networks (3. Week course) T-417-TOOR Computer Security (taught during the 12 week summer term)

Students take 30 ECTS. Students can choose elective courses within the department (i.e. Computer Science and Science and Engineering). If the course is outside of the department the following applies:

- In the case that the course is outside of the department students can apply for an assessment of up to 12 ECTS.
- Please note that the course Information Technology (in School of Business) cannot be taken as a course with computer science due to overlap. Note that you cannot get credit for both Data Processing and Databases because of overlap. Note that you cannot get credits for both Operating Systems and Fundamentals of Operating Systems because of overlap.

Undergradute courses fall term 2017

12 week period

T-103-STST Discrete Mathematics for Engineering Students

T-107-TOLH Computer Architecture

T-111-PROG Programming

T-114-VERK Problem Solving

T-117-STRI Discrete Mathematics I

T-202-GAG1 Databases

T-301-REIR Algorithms

T-302-HONN Software Design and Implementation

T-303-HUGB Software Engineering

T-316-UPPL The Information and Technology Society

T-317-CAST Calculus and Statistics

T-404-LOKA Final Project

T-409-TSAM Computer Networks

T-504-ITML Introduction to Machine Learning

T-511-TGRA Computer Graphics

T-513-CRNU Cryptography and Number Theory

T-514-VEFT Web Services

T-519-STOR Theory of Computation

T-603-THYD Compilers

T-622-UROP Undergraduate Research Opportunity

I-406-IERP Introduction to ERP System

3 week period

T-113-VLN1 Practical Project 1

T-168-ITST IT-Strategy

T-308-PRLA The Python Programming Language

E-402-STFO Mathematical Programming

T-488-MAPP Mobile App Development

T-542-HGOP Introduction to Quality Management and Testing

T-624-CGDD Computer Game Design & Development

Description of courses

T-103-STST Discrete Mathematics for Engineering Students

Credits: 6 ECTS

Description

The main material in this course consists of various aspects of mathematics that are basic to an understanding of the fundamentals of Computer Science. Various topics are discussed and their relevance to practical issues in Computer Science demonstrated. Topics covered include: logic and set theory, functions, relations, matrices, mathematical induction, counting techniques and graph

theory. Further there is a brief introduction on the cardinality of infinite sets and computability. Finally we discuss formal languages, grammars and finite automata.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be familiar with various topics in discrete mathematics that are important for an understanding of the fundamentals of computer science.
- Know basic concepts in propositional logic and predicate logic.
- Have been introduced to logic and formal reasoning.
- Know basic set operations.
- Know basic properties af functions, in particular logarithmic and exponential functions, the floor function and ceiling function.
- Have learnt introductory matrix algebra.
- Know basic counting techniques.
- Know basic concepts of recurrence relations.
- Be familiar with basic material on relations.
- Know basic concepts in graph theory, for instance Euler and Hamilton paths, shortest path and graph coloring.
- Be familiar with the cardinality of infinite sets.
- Understand the concept of computability and the proof that the Halting problem is unsolvable.
- Know introductory material on formal languages, grammars and finite automata.

Skills

- Be able to construct truth tables, use basic logical equivalences in propositional logic and use quantifiers.
- Be able to construct direct and indirect proofs.
- Be able to construct proofs by mathematical induction and strong induction. Also be able to construct inductive definitions.
- Be able to prove formulas in set theory using basic set identities.
- Be able to solve simple problems involving logarithmic functions, exponential functions, the floor function and the ceiling function.
- Be able to use basic matrix operations, e.g. multiplication, for matrices with numbers as well as boolean matrices.
- Be able to solve simple counting problems for finite sets, e.g. using permutations and combinations.
- Be able to construct recurrence relations.
- Be able to use recurrence relations as a model to solve various problems.
- Be able to analyze basic properties of relations, in particular equivalence relations.
- Be able to solve problems in graph theory, e.g. involving Euler and Hamilton paths and counting the number of different paths of a certain length.
- Be able to use Dijkstra’s algorithm to find the shortest path in a graph.
- Be able to find the chromatic number of various graphs.
- Be able to use graph theory to solve certain practical problems.

- Be able to construct recursive definitions, e.g. for graphs and trees, and prove statements using structural induction.
- Be able to determine whether sets are countable and give proofs similar to the standard proofs for the set of rational numbers and the set of real numbers.
- Be able to construct regular grammars, regular expressions and finite automata (DFA and NFA) for simple problems. Also be able to convert one of these forms to another.
- Be able to construct context-free grammars.

Competence

- Be able to use logic to analyze statements in the English language.
- Be able to apply graph theory models in various situations outside the scope of the course.
- Be able to use the material in the course to understand formal reasoning in later courses.
- Be able to utilize the knowledge gained on formal languages, grammars and finite automata towards a deeper understanding of the structure of programming languages.

Prerequisites

None

T-107-TOLH Computer Architecture

Credits: 6 ECTS

Description

In this course, students will learn the fundamental operations of a computer, with a special emphasis on issues related to programmers. They will learn how and why the CPU works, how it uses binary math for calculations, and how numbers and data is presented in binary. Students become familiar with reading x86_64 assembly code. They learn how programs are loaded into memory (register, cache, RAM etc.). Students learn how to use common commands in the command line.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to describe the architecture of a computer system in terms of the major building blocks, for instance, CPU, I/O, main memory and operating system.
- Be able to explain what programs are and how they run on the hardware
- Be able to explain programs written in an instruction set of a CPU (x86_64).
- Be able to describe in detail how data, including numbers, are represented, stored, and retrieved in computers.
- Have gained basic proficiency with the UNIX / Linux operating system

Skills

- Be able to write and explain basic x86_64 assembly code.
- Be able to disassemble, trace and perform rudimentary debugging of programs written in Intel x86_64 assembly.
- Be able to write and debug simple programs in the C programming language.
- Be able to use command line tools for basic tasks in Linux or other Unix-based operating systems.

- Be able to implement basic mathematical functions using only binary operators.

Prerequisites

None

T-111-PROG Programming

Credits: 6 ECTS

Description

This is an introductory course in computer programming using the C++ language. Fundamental programming constructs are covered, e.g. variables, types, control structures, functions and pointers, as well as built-in data structures like arrays, strings and vectors. The concept of a class is introduced and how it supports encapsulation and information hiding in the context of object-oriented programming. Students learn to use both an Integrated Development Environment (IDE) and command prompt mechanisms for development and compilation.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to describe the meaning of the following concepts: encapsulation, information hiding and abstract data type.
- Be able to describe how the concept of a class supports the above concepts.
- Understand the difference between an interface and an implementation.

Skills

- Be able to use an integrated development environment (IDE) for developing and compiling a program.
- Be able to implement, test, debug, change and explain a program that uses each of the following basic programming constructs: variables, types, expressions and assignments, simple I/O, conditional and iterative statements, arrays and functions.
- Be able to choose appropriate conditional and iterative constructs for a given programming task.
- Be able to apply top-down design to break a program into smaller pieces.
- Be able to apply different methods of parameter passing.
- Be able to write a program that use pointers and dynamic arrays.
- Be able to apply overloading with regard to operations.
- Be able to design, implement, test, debug and explain a program that uses classes.

Competence

- Be able to design and implement a program for a problem that is described in a general manner.

Prerequisites

None

T-113-VLN1 Practical Project 1

Credits: 6 ECTS

Description

The course is based on the knowledge and experience the students have previously acquired during the programming course. Students will gain a greater understanding of the use of classes and object oriented programming by creating layered software projects. Troubleshooting and debugging will be covered. Students are introduced to the SQL programming language and response driven programming with a graphical user interface. Students will be introduced to a version control system that will be used throughout the course.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to identify the main types of UI programs and write simple such programs.
- Know the advantages of a layered architecture.
- Recognize the benefits of using a version control system.
- Be able to discuss copyright, intellectual property, data protection and security.

Skills

- Be able to write simple algorithms.
- Be able to give simple commands in the console.
- Know additional skills in input data validation and application debugging.

Competence

- Be able to use classes and object oriented programming when constructing a simple software project.
- Be able to set up small databases, retrieve data from them, and write data using SQL.

Prerequisites

To have been enrolled in the course(s) T-111-PROG Programming

T-114-VERK Problem Solving

Credits: 6 ECTS

Description

In this course students develop skills in practical use of fundamental computer science, including programming, discrete mathematics and logic circuits. Sophisticated working practice will be emphasized and students will develop their skills in technical writing, presentation, and problem solving. All assignments are solved in-class, where students obtain advice from teachers and teaching assistants.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Know the basic concepts of theoretical computer science such as, game theory, state automata, grammars, regular expressions, recursion, dynamic programming, proof by induction, and parallel processes.
- Know the practical use of computer science such as the field of genetics and data compression.

Skills

- Know general problem solving as well as problem solving in the field of game theory, logic and theoretical computer science.
- Be able to write technical texts.
- Be able to give talks and do group work.
- Be able to use LaTeX.
- Be able to write small applications to perform various tasks.

Competence

- Be able to solve puzzles individually or in association with others, and express the solutions in written and spoken language.

Prerequisites

None

T-117-STR1 Discrete Mathematics I

Credits: 6 ECTS

Description

The main material in this course consists of various aspects of mathematics that are basic to an understanding of the fundamentals of Computer Science. Various topics are discussed and their relevance to practical issues in Computer Science demonstrated. Topics covered include: logic and set theory, functions, relations, matrices, mathematical induction, counting techniques and graph theory.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be familiar with various topics in discrete mathematics that are important for an understanding of the fundamentals of computer science.
- Know basic concepts in propositional logic and predicate logic.
- Have been introduced to logic and formal reasoning.
- Know basic set operations.
- Know basic properties of functions, in particular logarithmic and exponential functions, the floor function and ceiling function.
- Have learnt introductory matrix algebra.
- Know basic counting techniques.
- Have learnt basic concepts of recurrence relations.
- Be familiar with basic material on relations.
- Know basic concepts in graph theory, for instance Euler and Hamilton paths, shortest path and graph coloring.
- *Skills*
- Be able to construct truth tables, use basic logical equivalences in propositional logic and use quantifiers.
- Be able to construct direct and indirect proofs.
- Be able to construct proofs by mathematical induction and strong induction. Also be able to construct inductive definitions.
- Be able to prove formulas in set theory using basic set identities.

- Be able to solve simple problems involving logarithmic functions, exponential functions, the floor function and the ceiling function.
- Be able to use basic matrix operations, e.g. multiplication, for matrices with numbers as well as boolean matrices.
- Be able to solve simple counting problems for finite sets, e.g. using permutations and combinations.
- Be able to construct recurrence relations.
- Be able to use recurrence relations as a model to solve various problems.
- Be able to analyze basic properties of relations, in particular equivalence relations.
- Be able to solve problems in graph theory, e.g. involving Euler and Hamilton paths and counting the number of different paths of a certain length.
- Be able to use Dijkstra's algorithm to find the shortest path in a graph.
- Be able to find the chromatic number of various graphs.
- Be able to use graph theory to solve certain practical problems.

Competence

- Be able to use logic to analyze statements in the English language.
- Be able to apply graph theory models in various situations outside the scope of the course.
- Be able to use the material in the course to understand formal reasoning in later courses.

Prerequisites

None

T-168-ITST IT-Strategy

Credits: 6 ECTS

Description

Strategy and strategic management is fundamentally about how to create competitive advantage for the company, what activities to perform to achieve those advantages and how different factors can contribute or threaten the achievement of those advantages. Information technology is one factor that has had the most sustained influence on the competitive positions of companies. Developments in information technology can create competitive advantages as well as erode existing advantages almost overnight.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to identify the main types of strategic planning.
- Be able to identify the useful role of the integration of information technology in strategic management.
- Be able to identify the key strategic elements of information technology.

Skills

- Be able to design a presentation on the policies of information technology.
- Be familiar with the elements of trade relating to information technology.
- Be able to define the steps in the management of information systems (based on CobiT).

Competence

- Be able to assess the need for different categories of information to support trade.
- Be able to plan the main steps in policy process.
- Be able to plan the main steps in the management of information technology (based on CobiT).

Prerequisites

None

T-202-GAG1 Databases

Credits: 6 ECTS

Description

The course is a hands-on introduction to information management in general and relational database management in particular, covering the following topics: the role and function of database management systems; the relational database model, including relational concepts and relational query languages; data modeling using the ER model and its conversion into a relational database schema; all major aspects of the SQL language, covered in detail, including DDL, DML, complex queries, views, procedures, triggers and transactions; transaction and administration concepts; and, finally, a brief discussion of alternative data models and approaches, such as unstructured databases, information retrieval and “big data”.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to discuss structured and unstructured databases in social and organizational context.
- Be able to describe concepts and measures related to reliability, scalability, efficiency and effectiveness.
- Be able to describe major components and functions of database management systems.
- Be able to describe and compare common data models.
- Be able to describe fundamental principles of the relational model.
- Be able to describe fundamental transaction concepts.
- Be able to describe basic database administration functions.
- Be able to discuss concepts and techniques for unstructured data and information retrieval.
- Be able to discuss major approaches to storing and processing large volumes of data.

Skills

- Be able to write SQL commands to create a complete relational database.
- Be able to write SQL commands to insert, delete, and modify data.
- Be able to write simple and complex SQL queries to retrieve data, including joins, aggregates, sub-queries, nested sub-queries, and division.
- Be able to write simple database views, stored procedures, triggers and transactions.
- Be able to write queries in relational algebra and tuple relational calculus.

Competence

- Be able to model data requirements and constraints using the ER-model.
- Be able to convert an ER-model into a corresponding relational schema.
- Be able to normalize a relational schema.
- Be able to select and create the appropriate indices for simple database queries and constraints.

Prerequisites

T-301-REIR Algorithms

Credits: 6 ECTS

Description

This course introduces the most important types of algorithms and data structures in use today. Emphasis is placed on algorithms for sorting, searching and graphs. The focus is on developing implementations, analyzing them or evaluating empirically, and assessing how useful they can be in actual situations.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to describe the efficiency of major algorithms for searching, sorting and hashing.
- Be able to describe the problem with exponential growth of brute-force solutions and its consequences.
- Be able to give examples of the use of graphs, trees, and symbol tables.
- Be able to explain the main features of the major methods.
- Be able to describe major versions of the symbol table.

Skills

- Be able to specify computational problems from general textual Description.
- Be able to apply different search methods on trees and graphs.
- Be able to trace the execution of operations on classic data structures: heaps, binary search trees, red-black trees and union-find.
- Be able to solve tasks with the fundamental algorithms for graphs, such as depth-first and breadth-first search, transitive closure, topological sort, and algorithms for shortest paths and minimum spanning trees.
- Be able to assess the impact of different implementation of abstract data types on the time complexity of algorithms.
- Be able to use “big-O”, omega and theta notations to give the asymptotic upper, lower and tight limits on the time and space complexity of algorithms.
- Be able to apply the scientific method in the timing of algorithms.
- Be able to implement generic data structures and apply them to different data.

Competence

- Be able to assess algorithms, choose between possible solutions, justify the choice of method and implement in programs.

Prerequisites

To have passed the course(s) T-201-GSKI Data Structures, T-117-STR1 Discrete Mathematics I or to have passed the course(s) T-201-GSKI Data Structures, T-103-STST Discrete Mathematics for Engineering Students

T-302-HONN Software Design and Implementation

Credits: 6 ECTS

Description

The main objective of the course is to show how to design and implement software enterprise solutions. To realize this, the focus is on object oriented architecture and module design. Many known design patterns are discussed and evaluated. The focus is on layered Internet systems and web APIs. The course covers how to build flexible system that are easy to adapt, maintain, and operate. Options facing architects and how to recognize important key design goals is covered. Topics like performance and scalability of enterprise solutions are also covered. The course uses the Java programming language along with several open source libraries, APIs, and open source tools.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

Be knowledgeable in the basics of design software.

Understand the different software architecture and what options are available.

Have gained insight into how the software is organized in today's software sector.

Skills

Know how to design software with different patterns.

Be trained in professional practices in software development.

Competence

Be able to build a software framework such that generic units are reused.

Be able to design and build flexible software.

Be able to design and build fast scalable solutions.

Prerequisites

To have passed the course(s) T-213-VEFF Web Programming

T-303-HUGB Software Engineering

Credits: 6 ECTS

Description

The main topic of this course is the software development lifecycle and working methods associated with the life cycle. We will look at the history of development models, the role of individual methods will be discussed, with an emphasis on Agile development models and practices within those models. Special emphasis is on how they should support an incremental delivery of value to the customer, and even more, the continuous improvement of the process an organization is using. This course covers the following: Development models, practices and supporting tools, project management, quality management, planning, cost estimates, software metrics, configuration management, the build process, continuous integrations, testing and teamwork. We will also seek to get experienced individuals from industry to share their vision on software development with students.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be familiar with the life cycle steps and the main life cycle models used in software development.
- Know how to set up a vision for the software project.
- Know how to perform requirement gathering and keep them organized.
- Know how to estimate effort, create and maintain project plan.
- Be familiar with different categories of version control systems.
- Be familiar with different categories of testing.

Skills

- Be able to analyse and organize requirements.
- Have gained experience in estimation and planning.
- Be able to use version control systems.
- Be able to do a test-driven development.
- Have gained experience in system testing.
- Be able to create automatic build process.

Competence

- Be able to select the appropriate method to organize software projects.
- Be able to set up a technical infrastructure for development, test and production environment.
- Be able to use diciplined and sophisticated methods while developing, operating and maintaing software systems with focus on quality.

Prerequisites

To have passed the course(s) T-216-GHOH Software Requirements and Design

T-308-PRLA The Python Programming Language

Credits: 6 ECTS

Description

The programming language Python has gained popularity in recent years, especially as a script language, but it has also been used to produce larger systems. Its popularity is due to the simplicity of the language, readability and support for many common programming paradigms, e.g. object oriented programming and functional programming. In this course we will focus on the use of Python as a script language and writing small programs. The course will go over the following ways Python can be used:

Interaction with web services

Fetch data from the web

Create small window applications

Parse data and store in a format that is easy to read

Work with data

Interacting with the file system

Launch processes

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to describe the main aspects of Python.
- Understand the syntax of Python.
- Know the advantages and disadvantages of using Python in solving problems.
- Be able to identify the main embedded application modules in Python.

Skills

- Be able to write a simple program in Python.
- Be able to use the basic built-in application modules for Python.
- Be able to take advantage of external libraries for Python.
- Be able to use Python in:
- The collection of data (for example, from the Internet and from files)
- Handling data
- The presentation of data
- Working in the file operating system
- To communicate with the operating system

Competence

- Be able to decide when it is appropriate to use Python when solving problems.
- Be able to take advantage of Python to solve various problems.

Prerequisites

To have passed the course(s) T-107-TOLH Computer Architecture, T-201-GSKI Data Structures

T-316-UPPL The Information and Technology Society

Credits: 6 ECTS

Description

This course exams the social, legal and ethical topics related to information and communication in modern society. The main themes of the course will be:

Privacy and security

Intellectual wealth

Computer crime and other legal issues

Computers and risk

Ethical base, instructions and warranty

Effects of computerization on the workplace, work practices, teamwork and professional culture

E-commerce and E-government

Society, internet culture and the impact on health and education

Emphasis will be placed on training students to write reports with their projects.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to describe the advantages and disadvantages of the information society.

- Be aware of the social, ethical and philosophical impact of computerization.
- Be able to understand the impact of information and communication technology in homes, schools, workplaces, recreation, health and education.
- Be able to identify key moral questions relating to computers and who is responsible when working with computers.
- Know the legal environment, information technology and legal issues such as privacy and security, intellectual wealth and computer crime.

Skills

- Be able to write reports and articles on topics related to computers.

Competence

- Be able to follow the development of the information society and be able to evaluate it critically.
- Be able to articulate a vision of the desired effect of computerization.

Prerequisites

None

T-317-CAST Calculus and Statistics

Credits: 6 ECTS

Description

The course is divided into two parts, calculus in the first half and statistics in the second half. Limits, differentiation and integration of functions of one variable are covered. An introduction to probability and statistics will be covered as well. The material includes:

Discrete probability distributions, in particular the binomial distribution

Continuous probability distributions with the main emphasis on the normal distribution

Confidence intervals and hypothesis testing

Correlation and linear regression

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Know basic properties of functions of one variable, such as polynomial, rational, logarithmic, exponential and trigonometric functions.
- Be familiar with the basic concepts of calculus, such as continuity and differentiability.
- Know the derivatives of common functions of one variable, such as polynomial, rational, logarithmic, exponential and trigonometric functions.
- Understand integration and know integrals for common functions of one variable. Be familiar with integration by substitution and integration by parts.
- Know what a differential equation is and have seen some examples of differential equations.
- Be familiar with discrete probability and some basic methods for its calculation, in particular permutations and combinations.
- Be familiar with discrete probability distributions, in particular the binomial distribution, and know how to compute their expected value and standard deviation.

- Know continuous probability distributions, in particular the normal distribution and the t-distribution.
- Be familiar with confidence intervals and hypothesis testing.
- Be familiar with correlation and regression.

Skills

- Be able to differentiate functions of one variable, such as polynomials, rational, logarithmic, exponential and trigonometric functions.
- Be able to use differentiation to sketch the graphs of various functions of one variable.
- Be able to integrate various functions of one variable, for instance using integration by substitution or integration by parts.
- Be able to solve various practical problems by constructing a function and using differentiation to find its maximum or minimum value.
- Be able to solve very simple differential equations, for example for exponential growth or exponential decay.
- Be able to calculate discrete probability using techniques such as permutations and combinations.
- Be able to calculate expected value and standard deviation for discrete probability distributions, such as the binomial distribution.
- Be able to compute probabilities for continuous variables, using for example the normal distribution or the t-distribution.
- Be able to compute confidence intervals and test hypotheses.
- Be able to compute the correlation coefficient and find a regression line.

Competence

- Be able to analyze various problems and apply the methods of the calculus of one variable to solve them.
- Be able to apply hypothesis testing to analyze sets of measured data.

Prerequisites

None

T-404-LOKA Final Project

Credits: 12 ECTS

Description

The Final Project consists of software development in collaboration with a customer and users outside the university. The purpose of the final project is to give students experience of working independently on specification, design and implementation of software and to use accepted methods in the development cycle. Normally 2 to 4 students work together in a project group. While working on the project, students gain practical experience of analysis, design, programming and testing. The projects are evaluated by the project supervisor and two other internal examiners. The grade is based on evaluation at various stages of development and takes into account all aspects of the development work. The projects conclude with a public presentation. To be able to register for a final project, students need to have finished at least 78 ECTS credits.

Learning Outcomes

Upon completion of the course, the student should:

Skills

- Have gained training and experience in presenting their work for different audiences, ie with or without technical backgrounds.
- Have gained experience working on a mid-sized software project with a team.
- Be able to use a version control system in software development.
- Be able to organize and work according to a schedule in making a software system.
- Have gained practical training in project management.

Competence

- Have gained skills in designing, analyzing and implementing software.
- Be able to choose and justify the choice of an approved method of software development.
- Be able to define and carry out the user, unit and system testing.
- Be able to analyze user needs and implement the software necessary to fulfill the user needs.
- Be able to explain the position of the project, what the project was created to perform, what is left, and give a project status based on schedule.

Prerequisites

T-216-GHOH Software Requirements and Design, T-220-VLN2 Practical Project 2, T-303-HUGB Software Engineering, to have completed at minimum 78 credits

T-409-TSAM Computer Networks

Credits: 6 ECTS

Description

The course begins with a short overview of network systems and services. After introduction, the focus will be on the layers of the OSI and IETF models. The following network layers will be studied in detail:

Application layer (WWW, HTTP, DNS, SMTP, FTP etc)

Transport layer (UDP and TCP)

Network layer (link state routing and distance vector routing, IP, IP-addresses)

Link layer (MAC, Ethernet, hubs and switches).

Finally an introduction to some more specific topics such as mobile networks, multimedia networking, and network security will be given. Students will also work on the topics through programming assignments and homework.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to explain the layers of the OSI and IETF network protocol stack and their interactions.
- Be able to explain basic application layer protocols such as HTTP, SMTP, and P2P applications.
- Be able to discuss and analyze the transport layer protocols TCP and UDP.
- Be able to explain details of the IP protocol.
- Be able to understand link-layer protocols and technology (MAC, access protocols, Wireless LAN, GSM, UMTS).
- Be able to discuss performance assumptions and scalability in computer systems and networks.

- Be able to explain basic security terminology, security threats in networks, countermeasures, symmetric and public key cryptography.
- Be able to understand how to write network services securely.

Skills

- Be able to partition a network into subnets according to user specifications.
- Be able to explain and analyze traces of real-world network traffic.
- Be able to work with application layer networking interfaces.

Competence

- Be able to write simple network service using network primitives.
- Be able to write secure network applications.

Prerequisites

T-201-GSKI Data structures

T-201-GSKI Data Structures

Credits: 6 ECTS

Description

This course discusses various data structures, like linked lists, stacks, queues, trees and hash tables. Recursive programming and sorting algorithms are also discussed. At the same time, emphasis is put on abstract data types, object-oriented programming, templates, and exception handling. The programming language used in this course is C++.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to identify the basic steps and inductive steps in the problem defined in a recursive manner.
- Be able to identify the basic steps and inductive steps in the problem defined in a recursive manner.
- Be able to describe the concepts of object-oriented design with regard to encapsulation, inheritance and polymorphism.
- Be able to describe and understand the meaning of complexity/growth function algorithms.

Skills

- Be able to write programs that use each of the following data structures: arrays, linked lists, stacks, queues, trees, and hash tables.
- Be able to implement various types of data structure by using linked list.
- Be able to implement simple recursive functions.
- Be able to design, implement, test and debug a program in object-oriented programming language.
- Be able to write a program that uses inheritance and polymorphism to solve a specific problem.

- Be able to write programs that responds to exceptions raised during execution.
- Be able to apply sequential search, binary search and ranking algorithm under various circumstances.
- Be able to use abstract data types by having access to their declaration only.

Competence

- Be able to design and develop programs for the programs described in general terms.
- Be able to choose the appropriate data structure for modelling of a given problem.

Prerequisites

T-111-PROG

T-488-MAPP Mobile App Development

Credits: 6 ECTS

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, the student should:

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.

Prerequisites

T-427-WEPO Web Programming II, T-302-HONN Software Design and Implementation, T-501-FMAL Programming Languages

T-504-ITML Introduction to Machine Learning

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

Upon completion of the course, the student should:

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, Naive 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.

Competence

- 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.

Prerequisites

T-301-REIR Algorithms, T-419-STR2 Discrete Mathematics II, T-317-CAST Calculus and Statistics or T-301-REIR Algorithms, T-103-STST Discrete Mathematics for Engineering Students, T-101-STA1 Mathematics I

T-511-TGRA Computer Graphics

Description

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

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

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

Skills

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

Competence

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

Prerequisites

To have passed the course(s) T-301-REIR Algorithms

T-513-CRNU Cryptography and Number Theory

Description

This class covers the basics of cryptography and number theory, starting with classical ciphers and the tools from number theory necessary for doing cryptography. Symmetric and asymmetric ciphers will be covered. Some topics from groups, rings and fields will be introduced and used, especially when looking at elliptic curve cryptography. There will be some programming exercises in addition to standard mathematical homework. The programming language Sage will be used.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Know the purpose of cryptography and its uses throughout history.
- Know the basics of number theory, especially relating to cryptography
- Know the Sage programming language, especially how to implement algorithms from number theory and cryptography.
- Know the most common algorithms used in cryptography, e.g. the RSA public key system.
- Know the basics of information theory.
- Know the basics of finite fields and how they are used in cryptography.
- Know the basics of elliptic curves and how they are used in cryptography.
- Know how cryptography is applied, e.g. in multi-party computation, zero knowledge proofs, digital cash and voting systems.

Skills

- Know how to use simple cryptographic methods to encrypt short texts by hand.
- Be able to write code in Sage to use powerful cryptographic methods to encrypt text.
- Be able to solve number theoretic problems, with pencil and paper, as well as with Sage.

- Be able to implement common algorithms in cryptography, e.g. Euclids algorithm for the greatest common divisor and Diffie-Hellman key exchange.

Competence

- Recognize where to apply the methods of cryptography and which methods are breakable.
- Be able to apply their knowledge of number theory to solve problems in other mathematical courses, especially where algebra is needed.
- Be able to use Sage as a tool in other programming and mathematical courses, for testing conjectures, drawing graphs, etc.

Prerequisites

T-419-STR2 Discrete Mathematics II, T-317-CAST Calculus and Statistics or T-101-STA1 Mathematics I , T103-STST Discrete Mathematics for Engineering Students

T-514-VEFT Web Services

Description

In this course students will learn to design and implement web services. The course will exam various server libraries, as well as examine the main pattern and practices in programming on the server. The course will cover the following material: server side frameworks such as ASP.NET Web API / C #, Node.js / JavaScript, REST web services (concepts, implementation), Design patterns, Logging / Monitoring, Security, Caching, Deployment and Microservices.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be familiar with programming libraries on the server side.
- Be familiar with REST services, and know the differences between REST and RPC services
- Be able to know the differences between services which deliver data only, and systems which render web pages on the server.
- Be familiar with the main security concerns when implementing web services.

Skills

- Be able to write backend services in at least two common environments.
- Be able to set up an operable webservice.

Competence

- Know which libraries will be suitable under different circumstances during server side development.
- Know when to use a web service, and when to write a system which generates web pages on the server.

Prerequisites

T-213-VEFF Web Programming, T-202-GAG1 Databases

T-515-NOTH User-Centered Software Development

Description

The course will focus on teaching methods for analysing, designing and evaluating software systems anticipating the users aspects in software development. Students will gain skills in using particular methods for analysis, design and evaluation of user interfaces. Furthermore, other methods for analysing, designing and evaluating user interfaces will be described. Research on user centred software development methods will be described, when it is best to use each method and how practitioners have ranked the methods. The integration of user centred software development methods into Scrum will be discussed, and the integration of user experience into lean software development. Furthermore, experiences from industry will be a part of the course. The methods taught in this course supplement those taught in the course Software Requirements and Design.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be familiar with several methods for analysing the user needs for software systems.
- Be familiar with evaluation with and without the participation of users.
- Be familiar with guidelines for good user interface design.
- Be familiar with the integration of user-centered design methods in the Scrum software process.

Skills

- Be able to make a vision (ie, Visioning) for a software project and explain it.
- Be able to analyse the context of use for software systems.
- Be able to perform contextual inquiries and derive the results using an affinity diagram.
- Be able to design an interface that is based on the relationship schema and test it with users.
- Be able to perform formal user evaluations.

Competence

- Be able to choose which user-centered design methods are suitable in different cases.
- Know the advantages and disadvantages of user centred design methods.

Prerequisites

T-216-GHOH Software Requirements and Design

T-519-STO4 Theory of Computation

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

Upon completion of the course, the student should:

Knowledge

- Know deterministic and undeterministic Finite Automata (DFA and NFA), regular languages and their most important properties.
- Know what it means that two such automata are equivalent.
- Know that an NFA can always be transformed into an equivalent DFA.
- Know the standard language operators that preserve regularity.
- Know regular expressions and the language each of them describes.
- Know the correspondence between finite automata and regular expressions.
- Know the Pumping Lemma for regular languages.
- Know context free grammars, context free languages and push-down automata and the correspondence between the concepts.
- Know the Pumping Lemma for context free language.
- Know Turing machines and different variants of those.
- Know what it means for a language to be Turing decidable and Turing recognizable (acceptable)
- Know encoding of problems as formal languages.
- Know the Halting problem for Turing Machines and that it is Turing recognizable but not Turing decidable.
- Know what time complexity for Turing Machines means.
- Know the complexity classes P and NP.
- Know the concepts Turing-complete and reduction from one language to another.
- Know a few classical problems and know to which complexity class they belong (P, NP, NP-complete).

Skills

- Be able to draw DFAs and NFAs and describe in words the language they accept.
- Be able to draw an NFA (or DFA) for a simple regular language from the Description of that language.
- Be able to describe the strings in a regular language from a regular expression that describes it.
- Be able to write a regular expression that describes a simple regular language based on its Description in words.
- Be able to show that two simple automata or an automaton and a regular expression are equivalent.
- Be able to turn a NFA into an equivalent DFA.
- Be able to turn a regular expression into an equivalent NFA.
- Be able to turn a DFA into an equivalent regular expression.
- Be able to show that a given language is regular by using closure properties of language operators.
- Be able to show that a given language is not regular by using closure properties of language operators.
- Be able to show that a given language is not regular by using the Pumping Lemma for regular languages.
- Be able to describe in words a language that a context free grammar generates.
- Be able to create a context free grammar that generates a language from a Description in words.
- Be able to describe in words a language accepted by a push-down automaton.
- Be able to create a push-down automaton for a simple context-free language from a Description in words.
- Be able to turn a context free grammar into an equivalent push-down automaton.
- Be able to show that a language is context free by using closure properties for context free languages.
- Be able to show that a language is not context free by using proof by contradiction and closure properties for regular and/or context free languages.

- Be able to show that a language is not context free by using the Pumping lemma for context free languages.
- Be able to draw a Turing machine and describe the language it accepts.
- Be able to describe the Halting problem and prove that it is not decidable.
- Be able to show that several problems (when they are described as languages) regarding regular and context free languages can be decidable, acceptable or neither of these two.
- Be able to show that a language is decidable by using closure properties for such languages.
- Be able to use reduction of one language to another to reason about the decidability or recognizability of these languages.
- Be able to use reduction of one language to another to reason that a language is non-decidable or non-recognizable.
- Be able to determine time complexity for a simple Turing machine.
- In simple cases, be able to decide if a language belonged to the complexity class P or NP.
- Be able to use closure properties for these classes to reason about which of them languages belong to.
- Be able to use polynomial time reductions from a given NP language to a known NP-complete language to prove that former is NP-language too.

Competence

- Be able to use finite automata and their properties in computer science.
- Be able to use the properties of context free grammars in programming
- Be able to use the properties of context free grammars and corresponding push-down automata in the development of compilers for programming languages.
- Be able to reason about how difficult problems are to solve according to their possible decidability and their complexity class.

Prerequisites

T-301-REIR Algorithms, T-419-STR2 Discrete Mathematics II or T-301-REIR Algorithms, T-103-STST Discrete Mathematics for Engineering Students

T-542-HGOP Introduction to Quality Management and Testing

Description

The course will cover methods to ensure the quality of software, both the application code, user interface, delivery process and more. The technologies that were introduced in Software Engineering will be discussed in more detail. We will discuss the various types of tests and automation connections, such as unit testing, automated acceptance testing and automated software delivery. Property tests will also be discussed briefly.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to identify which items have the most impact on software quality.
- Be able to identify the main types of tests.
- Know which items need to be present to deliver the software repeatedly and reliably.

Skills

- Be able to write unit tests that cover the majority of the code system (code coverage).

- Be able to define and perform other types of tests, such as integration testing, load/performance testing, and manual testing such as usability testing and exploratory testing.
- Be able to set up a “delivery pipe” for web applications.

Competence

- Be able to determine how much work is required to test a system, and determine what kind of testing to focus on.

Prerequisites

T-303-HUGB Software Engineering, T-220-VLN2 Practical Project 2

T-603-THYD Compilers

Description

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

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

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

Skills

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

Competence

- Be able to design and build a simple compiler.

Prerequisites

To have passed the course(s) T-501-FMAL Programming Languages

T-622-UROP Undergraduate Research Opportunity

Description

Students receive training in research by working on research projects within the department in close collaboration with teachers. Activities can take various forms, all with the objective of increasing the skills and competences of students in the field of computer science or related fields. Projects can be independent research or development projects, or a part of a larger project.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to describe a research project and the area it belongs to.
- Be able to explain research and in particular research in computer science.

Skills

- Be able to define and follow a project schedule.
- Be able to follow the necessary steps to complete the goals set out.

Competence

- Be able to present and defend his/her findings in research to an audience.

Prerequisites

Research council approval

E-402-STFO Mathematical Programming

Description

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

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

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

Skills

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

Competence

- Be able to use a computer to test conjectures and run simulations.
- Be able to use dynamic programming to solve problems.
- Be able to use linear programming to solve problems.

- Be able to use search and other common methods to solve problems.
- Be able to choose an appropriate method to deal with different problems
- Be able to prove certain problems by hand, where running simulations is to time-consuming.

Prerequisites

T-301-REIR Algorithms, T-419-STR2 Discrete Mathematics II, T-317-CAST Calculus and Statistics or T-301-REIR Algorithms, T-103-STST Discrete Mathematics for Engineering Students, T-101-STA1 Mathematics I

I-406-IERP Introduction to ERP Systems

Description

The largest investment companies make in information technology is Enterprise Resource Planning, ERP. ERP is another way to describe the interaction between the processes and technologies in operation. The processes and technologies that are included are planning, procurement and product management, human resources, finances, inventory and sales. In recent years there has been a lot of change in this sector with systems becoming more versatile and powerful than before, and extends into the activity of companies. According to a recent survey by Gartner's, companies investment in information technology business systems will continue to increase in the coming years. The main obstacle for growth in this sector relates to the lack of personnel with expertise in this area, which is a combination of sustained knowledge of information technology and major business processes. The largest players in this market are SAO, Oracle and Microsoft, and the working environment is global. This course seeks to create a good base of knowledge for anyone who wants to further study the function and development of ERP systems. In the course different ways of implementation and the impact ERP systems have on business operaions will be discussed. The teaching in this course will be based on Microsoft Dynamics NAV. By the end of the course students will be able to understand the function and main features of ERP systems. During the course we will also "look under the hood" and give students the opportunity to work in the technology environment.

Learning Outcomes

Upon completion of the course, the student should:

Knowledge

- Be able to describe the functions and use of ERP systems and their development over time.
- Be able to define key components of ERP systems and describe their context.

Skills

- Be able to work with the main system components in Microsoft Dynamic NAV.
- Be able to program simple functionality with the ERP system.

Competence

- Be able to design processes within the ERP system

Prerequisites

T-111-PROG Programming