Detailed Content

Semester 3 / Term 1

Analysis III

Course Description:
The notions already seen in the analysis of a single variable are extended to the case of several variables with the introduction of new concepts such as differentiability and multiple integration.
In addition, we introduce the necessary knowledge about the simple and uniform convergences of series of functions, the development of functions in integer series,

Prerequisite : Analysis I and Analysis II

Evaluation Method : Coursework (40%) + Final Exam (60%)

1. **Functions in several variables**
   - 1.1. Topologies of R^n (n=2,3), Norms and balls of R^n. Open and closed parts of R^n.
   - 1.2. Real functions in several variables, Definitions and generalities
   - 1.3. Limit of a real function in several variables, definition, partial limits, operations on limits.
   - 1.4. Continuity of a real function in several variables and operations on continuous functions.
   - 1.5. Partial derivatives, gradient, functions of class C^1 on an open and differentiability (directional derivatives)
   - 1.6. Partial derivatives of higher order, functions of class C^k on an open. Schwarz's lemma. Taylor formula.
   - 1.7. Extremums of numerical functions of two variables.
   - 1.8. Double integrals.
   - 1.9. Triple integrals
2. **Numerical series**
   - 2.1. Definitions and generalities
   - 2.2. Cauchy criterion
   - 2.3. Absolute convergence
   - 2.4. Comparison criterion for series with positive terms
   - 2.5. Comparison criterion at the limit
   - 2.6. Cauchy and d'Alembert's rule
   - 2.7. Series with any terms (Leibnitz and Abel criterion)
   - 2.8. Dirichlet's theorem
3. **Improper integral**

3.1. Definition.
3.2. Convergence of an improper integral.
3.3. Comparison theorem for positive functions.
3.4. Comparison theorem at the limit.
3.5. Applications (Poisson integral calculation).
3.6. Integrals of functions of any sign
3.7. Absolute convergence. Abel's theorem.
3.8. Improper integrals with several singularities.

4. Integer series

4.1. Definition and generalities.
4.2. Abel's lemma.
4.3. Radius of convergence R of an integer series.
4.4. Cauchy's and d'Alembert's rules for computing R.
4.5. Properties of the sum of an integer series (continuity, derivability and integrability).
4.6. Integer series development and applications to the solution of ordinary differential equations.

5. Laplace transform

5.1. Definition of the Laplace transform and examples.
5.2. Properties of the transform.
5.3. Convolution theorem and properties of convolution.
5.4. Derivation and integration of the Laplace transform.
5.5. Inverse Laplace transform and examples.
5.6. Applications to the solution of different types of problems.

References

1. James Stewart: Calculus, Cengage Learning, 2015.
2. Serge Lang, Undergraduate Analysis. Springer, 1997.
3. Mohamed Hazi Fonctions de plusieurs variables réelles, OPU, 2018.

Data Structures and Algorithms

Course Description

This module introduces the principal fundamentals of data structures and algorithms used in computer science. Data structures will be formulated to represent information in such a way that it can be conveniently and efficiently manipulated by the algorithms that are developed.
On successful completion of this module, students should be able to demonstrate understanding of abstract models of data and computation, to explain and apply data structures such as binary trees, heap-trees, graphs and tables, and to explain the differences between basic complexity classes of algorithms (constant, linear, quadratic, logarithmic, exponential).

Prerequisite: Introduction to Programming, Object-Oriented Programming

Evaluation Method : Coursework (40%) + Final Exam (60%)

Course Content

1. **Introduction**

2. **Basic data structures and operations**
   - 2.1. Arrays, Loops, Invariants
   - 2.2. Linked-Lists, Recursion
   - 2.3. Stacks, Queues, Sets

3. **Searching**
   - 3.1. Requirements and Specification
   - 3.2. Linear versus Binary Search

4. **Efficiency and Complexity**
   - 4.1. Measuring Efficiency
   - 4.2. Complexity classes and Big-O notation

5. **Trees**
   - 5.1. Representations and primitive operators
   - 5.2. Quad Trees
   - 5.3. Binary Trees
   - 5.4. Binary Search Trees, B-Trees
   - 5.5. Priority Queues and Binary Heap Trees
   - 5.6. Binomial Heaps, Fibonacci Heaps

6. **Sorting Algorithms**
   - 6.1. General principles and strategies
   - 6.2. Insertion Sort, Selection Sort, Bubblesort
   - 6.3. Tree Sort, Heapsort
   - 6.4. Quicksort, Merge Sort
   - 6.5. Non-Comparison Sorts

7. Storing Data

7.1. Storing in Trees
7.2. Hash Tables
7.3. Handling Collisions

8. Graphs

8.1. Representations
8.2. Planarity and Traversals
8.3. Shortest Paths (Dijkstra's and Floyds Algorithms)
8.4. Minimal Spanning Trees (Prim's and Kruskal's Algorithms)
8.5. Travelling Salesmen and Vehicle Routing Problems

References

1. Michael T. Goodrich, Roberto Tamassia, David M. Mount, Data Structures and Algorithms in C++ 2nd Edition, Wiley, 2011
2. John Bullinaria, Lecture Notes for Data Structures and Algorithms, School of Computer Science. University of Birmingham. Birmingham, UK. Version of 27 March 2019. https://www.cs.bham.ac.uk/~jxb/DSA/dsa.pdf
3. Malik, D S. Data Structures using Java (2nd Edition), Cengage Learning. 2009
4. Weiss, Mark A. Data Structures and Algorithm Analysis in Java (3rd Edition), Pearson. 2012

Computer Architecture I

Course Description

The course addresses the concepts, principles and techniques of designing digital systems. The course teaches the fundamentals of digital systems applying the logic design and development techniques. This course forms the basis for the study of advanced subjects like Computer Architecture and Organization. Students will learn principles of digital systems logic design.

Prerequisite : None

Evaluation Method : Coursework (40%) + Final Exam (60%)

Course Content

1. **Number Systems (3h)**
   - 1.1. Unsigned Binary Numbers
   - 1.2. Signed Binary Numbers (Sign and Magnitude ; Two's Complement)
   - 1.3. Addition and Subtraction
   - 1.4. Multiplication and Division
   - 1.5. Extending an n-bit binary to n+k-bits
   - 1.6. Floating point numbers
2. **Boolean Algebra (3h)**
   - 2.1. Logic Gates
   - 2.2. Theorems
   - 2.3. Simplification of Boolean Expression using Boolean Algebra
   - 2.4. Simplification of Boolean Expression using K-Maps.
3. **Combinational Logic Circuits (1.5h)**
   - 3.1. Standard Combinational Circuit Blocks
   - 3.2. Encoder
   - 3.3. Decoder
   - 3.4. Multiplexer
   - 3.5. Demultiplexer
   - 3.6. Adders
4. **Sequential Circuits (3h)**
   - 4.1. Flip-Flops
     - 4.1.1. Asynchronous Latches
     - 4.1.2. Synchronous Flip-Flops
   - 4.2. Finite State Machines
     - 4.2.1. State Transition Graphs
   - 4.3. Registers
   - 4.4. Standard Sequential Logic Circuits

4.4.1. Counters
4.4.2. Shift Registers

5. Computer Organisation (10.5h)

5.1. Von Neumann Architecture / Harvard Architecture
5.2. Data Path and Memory Bus
5.3. Arithmetic and Logic Unit (ALU)
5.4. Memory
5.4.1. Static Random Access Memory (SRAM)
5.4.2. Dynamic Random Access Memory (DRAM)
5.5. Control Unit (CU)
5.5.1. Register Transfer Language
5.5.2. Execution of Instructions
5.5.3. Microarchitecture
5.5.4. Complex and Reduced Instruction Sets (CISC/RISC)
5.6. Input/Output

References

1. William Stallings, Computer Organization and architecture : designing for performance 11th edition 2019
2. Andrew S. Tanenbaum, & Todd Austin, Structured Computer Organization, 6th edition 2012.
3. Paolo Zanella, Yves Ligier, Emmanuel Lazard, Architecture et technologie des ordinateurs : Cours et exercices corrigés - 6 Edition. Dunod. 2018.
4. Thomas L. Floyd. Digital Fundamentals, 11th Edition. Pearson, 2015.

Mathematical Logic

Course Description

This course aims to provide the basic notions of formal logic, mainly the syntax and semantics of propositional logic and first-order predicate logic. At the end of the course, the student will be able to manipulate logical expressions, check their well-formedness and deduce new expressions. The student will also be able to analyse the truth or falsity of a statement using the notions of satisfiability and model theory as well as verifying the validity of reasoning through systems of formal proof.

Prerequisite : Algebra

Evaluation Method

Coursework (40%) + Final Exam (60%)

Course Content

1. **Informal natural deduction**
   - 1.1. Proofs and sequents
   - 1.2. Arguments introducing 'and'
   - 1.3. Arguments eliminating 'and'
   - 1.4. Arguments using 'if'
   - 1.5. Arguments using 'if and only if'
   - 1.6. Arguments using 'not'
   - 1.7. Arguments using 'or'

2. **Propositional logic**
   - 2.1. LP, the language of propositions
   - 2.2. Parsing trees
   - 2.3. Propositional formulas
   - 2.4. Propositional natural deduction
   - 2.5. Truth tables
   - 2.6. Logical equivalence
   - 2.7. Substitution
   - 2.8. Disjunctive and conjunctive normal forms
   - 2.9. Soundness for propositional logic
   - 2.10. Completeness for propositional logic

3. **Quantifier-free logic**
   - 3.1. Terms
   - 3.2. Relations and functions
   - 3.3. The language of first-order logic
   - 3.4. Proof rules for equality
   - 3.5. Interpreting signatures
   - 3.6. Closed terms and sentences
   - 3.7. Satisfaction
   - 3.8. Diophantine sets and relations

3.9. Soundness for qf sentences
3.10. Adequacy and completeness for qf sentences

4. First-order logic

4.1. Quantifiers
4.2. Scope and freedom
4.3. Semantics of first-order logic
4.4. Natural deduction for first-order logic
4.5. Proof and truth in arithmetic
4.6. Soundness and completeness for first-order logic
4.7. First-order theories
4.8. Cardinality
4.9. Things that first-order logic can not do

References

1. Ian Chiswell. Mathematical Logic. Oxford University Press. 2007
2. Shawn Hedman. A first course in logic : an introduction to model theory, proof theory, computability, and complexity, Oxford: Oxford University Press, 2004

Semester 3 / Term 2

Linear Algebra II

Course Description

This course is the continuation of Linear Algebra 1. Diagonalization is revisited and the Jordan Canonical Form is introduced when a matrix is not diagonalizable. As an application, matrix exponential and systems of differential equations are studied. Orthogonality is focused on an algorithm for producing an orthonormal basis (the Gram-Schmidt Process) and the QR factorization theorem. The last chapter is dedicated to symmetric matrices and quadratic forms. In particular, the Spectral Theorem and the Singular Value Decomposition Theorem are introduced and some applications are presented.

Prerequisite : Algebra ; Linear Algebra 1
Evaluation Method : Coursework (40%) + Final Exam (60%)

1. **Diagonalization Revisited**
   - 1.1. Cayley-Hamilton Theorem
   - 1.2. Complex Eigenvalues
   - 1.3. Jordan Canonical Form
   - 1.4. Matrix Exponential
   - 1.5. Applications to Systems of Differential Equations
2. **Orthogonality**
   - 2.1. Inner Product, Length, and Orthogonality
   - 2.2. Orthogonal Sets
   - 2.3. Orthogonal Projections
   - 2.4. The Gram–Schmidt Process
   - 2.5. Least Squares Problems
3. **Symmetric Matrices and Quadratic Forms**
   - 3.1. Diagonalization of Symmetric Matrices
   - 3.2. Quadratic Forms
   - 3.3. Constrained Optimization
   - 3.4. The Singular Value Decomposition
   - 3.5. Applications to Image Processing and Statistics

References

1. Mark Peter Deisenroth, A. Aldo Faisal, Cheng Soon Ong, Mathematics for Machine Learning, Cambridge University Press, 2020
2. David C. Lay, Steven R. Lay, Judi J. McDonald, Linear Algebra and Its Applications, Fifth Edition. Pearson, 2015,
3. Gilbert Strang, Introduction to Linear Algebra, Cambridge University Press, 2016

Probability and Statistics II

Course Description : This course covers the classical aspects of probability theory and focuses on the probabilistic model and its basic properties. It also considers random experiments whose characteristic of interest can be modelled by univariate or multivariate random variables (discrete or continuous). It introduces random vectors, sequences of random variables, and different aspects of convergence.

Prerequisite : Probability and Statistics I, Analysis

Evaluation Method : Coursework (40%) + Final Exam (60%)

Course Content

1. **Random Variables**
   - 1.1. Random Variables
   - 1.2. Discrete Random Variables
   - 1.3. Expected Value and Expectation of a Function of a Random Variable
   - 1.4. Variance
   - 1.5. The Bernoulli and Binomial Random Variables
   - 1.6. The Poisson Random Variable
   - 1.7. Other Discrete Probability Distributions
   - 1.8. Expected Value of Sums of Random Variables
   - 1.9. Properties of the Cumulative Distribution Function

2. **Continuous Random Variables**
   - 2.1. Introduction
   - 2.2. Expectation and Variance of Continuous Random Variables
   - 2.3. The Uniform Random Variable
   - 2.4. Normal Random Variables
   - 2.5. Exponential Random Variables
   - 2.6. Other Continuous Distributions
   - 2.7. The Distribution of a Function of a Random Variable

3. **Jointly Distributed Random Variables**
   - 3.1. Joint Distribution Functions
   - 3.2. Independent Random Variables
   - 3.3. Sums of Independent Random Variables
   - 3.4. Conditional Distributions: Discrete Case
   - 3.5. Conditional Distributions: Continuous Case
   - 3.6. Order Statistics
   - 3.7. Joint Probability Distribution of Functions of Random Variables
   - 3.8. Exchangeable Random Variables
   - 3.9. Probability Bounds

4. **Properties Of Expectation**
   - 4.1. Introduction
   - 4.2. Expectation of Sums of Random Variables
   - 4.3. Moments of the Number of Events that Occur
   - 4.4. Covariance, Variance of Sums, and Correlations

4.5. Conditional Expectation
4.6. Conditional Expectation and Prediction
4.7. Moment Generating Functions and Characteristic Functions

5. **Limit Theorems**
   5.1. Chebyshev's Inequality and the Weak Law of Large Numbers
   5.2. The Central Limit Theorem
   5.3. The Strong Law of Large Numbers

6. **Statistical Inference**
   6.1. Point Estimation
   6.2. Interval Estimation
   6.3. Hypothesis Testing

References

1. Sheldon M. Ross, A first course in probability, Pearson, 2018.
2. Hossein Pishro-Nik, Introduction to probability, statistics and random processes, Kappa Research, 2014.
3. Sheldon M. Ross, Introduction in probability and statistics for scientists and engineers, Academic Press, 2014.
4. David Forsyth, Probability and statistics for computer science, Springer, 2018
5. F.M. Dekking, C. Kraaikamp, H.P. Lopuhaa and L.E. Meester: A Modern Introduction to Probability and Statistics Understanding Why and How,, Springer, 2005.

Complexity Theory

Course Description :
The aim of this module is to provide the basic concepts for the theory of computational complexity. The main models of computability are explained providing various examples of undecidable problems. Further, students are taught the measures of the complexity of problems and of algorithms, based on time and space being used on abstract models. Complexity classes are explained along with the notion of completeness established through a thorough study of NP-completeness.

Prerequisite : Introduction to Programming, Data Structures and Algorithms

Evaluation Method : Coursework (40%) + Final Exam (60%)

Course Content

1. **Introduction**
   - 1.1. Complexity theory
   - 1.2. Computability theory
   - 1.3. Mathematical notions
   - 1.4. Types of Proofs

2. **Automata theory**
   - 2.1. Regular Languages : Finite Automata, Non-determinism, Regular Expressions, nonregular languages.
   - 2.2. Context-free languages : Grammars, Pushdown automata

3. **Computability theory**
   - 3.1. Turing machines, recursively enumerable and recursive languages
   - 3.2. Church-Turing thesis
   - 3.3. Decidability
   - 3.4. Reducibility

4. **Complexity Theory**
   - 4.1. Complexity of algorithms and of problems
   - 4.2. Complexity classes P, NP, PSPACE
   - 4.3. Polynomial-time reduction
   - 4.4. NP-Completeness and Cook's theorem
   - 4.5. PSPACE-Completeness

References

1. Michael Sipser. Introduction to the Theory of Computation. 3rd Edition. 2012
2. Sanjeev Arora, Boaz Barak. Computational Complexity: A Modern Approach. 2009.
3. Ingo Wegener and R.Pruim. Complexity Theory: Exploring the Limits of Efficient Algorithms. 2005.
4. Kamal Bentahar. Online content https://github.coventry.ac.uk/pages/ab3735/5002CEM/ 2022.

Databases

Course Description :
The object of this course is to teach students the general concepts of relational databases and how to design a database that is anomaly free. Students will learn to design, create, populate, and query a database by working with a relational database engine and the SQL language. Students will also learn basic database administration skills such as creating users, granting/revoking privileges individually or collectively to several users through the use of roles. Further, students will learn how to create constraints with triggers in addition to the use of PL/SQL language.

Prerequisite : Programming, Data Structures

Evaluation Method : Coursework (40%) + Final Exam (60%)

Course Content

1. **Database Systems**
   - 1.1. Why Databases?
   - 1.2. Data versus Information
   - 1.3. Introducing the Database
   - 1.4. Why Database Design Is Important
   - 1.5. Evolution of File System Data Processing
   - 1.6. Problems with File System Data Processing
   - 1.7. Database Systems
2. **The Relational Database Model**
   - 2.1. A Logical View of Data
   - 2.2. Keys
   - 2.3. Integrity Rules
   - 2.4. Relational Algebra
   - 2.5. The Data Dictionary and the System Catalogue
   - 2.6. Relationships within the Relational Database
   - 2.7. Indexes
   - 2.8. Codd's Relational Database Rules
3. **Entity Relationship (ER) Modelling**
   - 3.1. The Entity Relationship Model
   - 3.2. Developing an ER Diagram
   - 3.3. The Extended Entity Relationship Model
   - 3.4. Transforming ER diagrams to logical models
   - 3.5. Entity Integrity: Selecting Primary Keys
4. **Normalisation of Database Tables**
   - 4.1. Functional dependencies
   - 4.2. Schema Refinement
   - 4.3. Database Tables and Normalisation
   - 4.4. The Need for Normalisation

4.5. The Normalisation Process
4.6. The Boyce-Codd Normal Form

5. Introduction to Structured Query Language (SQL)

5.1. Introduction to SQL
5.2. Basic SELECT Queries
5.3. SELECT Statement Options
5.4. FROM Clause Options
5.5. ORDER BY Clause Options
5.6. WHERE Clause Options
5.7. Aggregate Processing
5.8. Subqueries
5.9. Relational Set Operators
5.10. Crafting SELECT Queries

6. Advanced SQL (LABS)

6.1. Data Definition Commands
6.2. User Management
6.3. Creating Table Structures
6.4. Altering Table Structures
6.5. Data Manipulation Commands
6.6. Virtual Tables: Creating a View
6.7. Sequences
6.8. Procedural SQL
6.9. Embedded SQL
6.10. PLSQL
6.11. Triggers

7. Transaction Management and Concurrency Control

7.1. What Is a Transaction?
7.2. Concurrency Control
7.3. Concurrency Control with Locking Methods
7.4. Concurrency Control with Time Stamping Methods
7.5. Concurrency Control with Optimistic Methods
7.6. ANSI Levels of Transaction Isolation
7.7. Database Recovery Management

8. Database Performance Tuning and Query Optimization (OPTIONAL)

8.1. Database Performance-Tuning Concepts
8.2. Query Processing
8.3. Indexes and Query Optimization
8.4. Optimizer Choices
8.5. SQL Performance Tuning
8.6. Query Formulation
8.7. DBMS Performance Tuning

References

1. Carlos Coronel and Steven Morris. Database Systems: Design, Implementation, & Management, 13th Edition, 2018.

2. Raghu Ramakrishnan, Johannes Gehrke. Database Management Systems, McGraw-Hill Higher Education; 3rd edition. 2002
3. Ramez Elmasri, Shamkant B. Navathe. Fundamentals of Database Systems, Pearson; 7th edition, 2015.

Semester 4

Operational Research

Course description:

Operations research (OR) has many applications in science, engineering, economics, and industry and thus the ability to solve OR problems is crucial for both researchers and practitioners. Being able to solve real life problems and obtaining the right solution requires understanding and modeling the problem correctly and applying appropriate optimization tools and skills to solve the mathematical model. The aim of this module is to teach the student the techniques of modelling problems by a linear program or a graph, the methods of solving these models and the practical interpretation of the results.

Prerequisite: Linear Algebra. Analysis.

Evaluation Mode: Coursework (40%) + Final Exam (60%)

Content of the course

Part 1. Linear Optimization

1. **Generalities of linear programming**
    - 1.1. Concrete examples
    - 1.2. Modelling
    - 1.3. Graphical solution
    - 1.4. Optimal bases
2. **Simplex algorithm**
    - 2.1. Principle of the method
    - 2.2. Initialization
    - 2.3. Examples
3. **Post-optimal analysis**
    - 3.1. Optimality condition
    - 3.2. Post-optimal analysis of the second member of the constraints.
    - 3.3. Feasibility condition
4. **Duality**
    - 4.1. General duality theorems
    - 4.2. Primal-dual optimality conditions
    - 4.3. Complementary deviation theorem
    - 4.4. Dual simplex algorithm

Part 2. Graph Theory

5. **Definitions and basic concepts**
    - 5.1. Oriented graphs.
    - 5.2. Undirected graphs.
    - 5.3. Subgraph.

5.4. Partial graph.
5.5. Partial subgraph.
5.6. Fundamental theorem of graphs.
5.7. Special graphs.
5.8. Matrix representation of a graph and properties.
5.9. Isomorphisms of graphs.

6. Connections in graphs

6.1. Chains.
6.2. Cycles.
6.3. Paths.
6.4. Circuits.
6.5. Connectedness.
6.6. High connectedness.
6.7. Graphs without circuits: ordering.
6.8. Eulerian paths.
6.9. Hamiltonian paths.

7. Planar graphs

7.1. Characterization
7.2. Kuratowski's theorem.
7.3. Euler's formula
7.4. Dual graph of a planar graph.

8. Cycles and Cocycles

8.1. Base of cycles.
8.2. Base of cocycles (cyclomatic number).
8.3. Flows and tensions.
8.4. Orthogonality of two spaces

9. Partitioning problems

9.1. Stables and colouring in graphs.
9.2. Couplings and the chromatic index.
9.3. Link between coupling and transversal.
9.4. Link between overlap and stable.
9.5. Maximum coupling problem in bipartite graphs.

References

1. Frederick Hillier, Gerald Lieberman: Introduction to Operations Research, McGraw Hill, 2020.
2. R. J. Vanderbei. Linear Programming: Foundations and Extensions. Kluwer Academic Publishers, 1998.
3. Michael W. Carter, Camille C. Price: Operations Research: A Practical Introduction, Routledge, 2001.
4. West, D.B. Introduction to Graph Theory (2nd Edition), Prentice-Hall, 2000.
5. Wilson, R.J. Introduction to Graph Theory (5th Edition), Pearson, 2010.
6. C. Berge. Graphes. Book . Editions Gauthier-Villars. 1983.

Web Programming

Course Description:

This course is an introduction to programming for the World Wide Web. Students will be taught the concepts for developing web applications using various technologies and frameworks including HTML, CSS, JavaScript and server-side programming languages. The course will cover in depth the following aspects:

- HyperText Markup Language (HTML) for authoring web pages
- Cascading Style Sheets (CSS) for applying stylistic information to web pages
- JavaScript for creating interactive web pages
- Asynchronous JavaScript and XML (Ajax) with fetch and JSON for enhanced web interaction and applications
- PHP Hypertext Processor for generating dynamic pages on a web server
- Web services for handling and responding to client-side requests
- Security for Web Applications

Prerequisite : Introduction to Programming, Databases

Evaluation Method : Coursework (40%) + Final Exam (60%)

Course Content

1. **The Internet and World Wide Web**
   - 1.1. What is the Internet?
   - 1.2. The World Wide Web (WWW)
   - 1.3. Internet Technologies
2. **HTML Basics**
   - 2.1. HTML Structure
   - 2.2. HTML Elements
   - 2.3. Creating Forms
   - 2.4. Web Standards
3. **CSS for Styling**
   - 3.1. Basic CSS
   - 3.2. CSS Properties
   - 3.3. Page Layout
   - 3.4. Responsive CSS Design
   - 3.5. CSS Frameworks
4. **Javascript : Client-Side Programming**
   - 4.1. Key Javascript Concepts
   - 4.2. Javascript Syntax
   - 4.3. Program Logic
   - 4.4. JavaScript functions
   - 4.5. Document Object Model
   - 4.6. Arrays and Data types
   - 4.7. JSON Data format

4.8. Asynchronous JavaScript
4.9. JavaScript Frameworks

5. Server-Side Programming
5.1. Server-Side Basics
5.2. Cookies and Sessions
5.3. Programming Languages
5.3.1. PHP
5.3.2. Java/Servlet-JSP
5.3.3. Python
5.3.4. NodeJS
5.4. Web Frameworks

6. Integration & Extension
6.1. Database Programming
6.2. Web Services and APIs
6.3. Using Cloud Technologies
6.4. Large Scale Web Applications

7. Web Security
7.1. Security Principles
7.2. Cross-Site Scripting (XSS)
7.3. Validating Input Data
7.4. SQL Injection
7.5. Denial of Service
7.6. Securing Web Applications
7.7. Defending against Web Attacks

References

1. W3 Schools. Online content. https://www.w3schools.com (HTML, CSS, JS)
2. David Flanagan. The Definitive Guide: Master the World's Most-Used Programming Language, *7th Edition*, 2020
3. Robin Nixon. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (Learning PHP, MYSQL, Javascript, CSS & HTML5, 2018

Software Engineering

Course Description :

Successful software development depends on an in-depth understanding of how the phases and supporting activities of the software development life cycle work together. Each phase of the life cycle contributes to a reliable, maintainable product that satisfies user requirements. The application of good engineering practices throughout the cycle dramatically improves the likelihood of delivering a quality software project on time, in scope and within budget. While there are many rigorous methodologies, in fact most approaches and tools have a mixture of strengths and weaknesses.

The main objectives of the course are as follows:

- Describe and compare various software development methods and understand the context in which each approach might be applicable.
- Develop students' critical skills to distinguish sound development practices from ad hoc practices, judge which technique would be most appropriate for solving large-scale software problems, and articulate the benefits of applying sound practices.
- Expand students' familiarity with mainstream languages used to model and analyze object designs (e.g., UML).

Prerequisite : Programming

Evaluation Method : Coursework (40%) + Final Exam (60%)

Course Content

1. **Introduction**
    a. Software Development challenges
    b. Software scope
    c. Software Engineering Discipline
    d. Software Methodologies and Related Process Models
    e. The Human Side of Software Development (optional)
    f. Case studies
2. **Software processes**
    a. Software process models
    b. Process Activities
    c. Traditional Life Cycle Models
        i. Waterfall
        ii. V
        iii. Phased
        iv. Evolutionary
        v. Spiral
    d. Alternative Techniques
        i. Unified Process
        ii. RAD
        iii. JAD
        iv. Prototyping
    e. Agile Software Engineering Process Models
        i. Extreme Programming

        ii. Agile Software Development
        iii. DevOps
        iv. Site Reliability Engineering (SRE)
    f. IEEE Standards for Software Engineering Processes and Specifications (optional)
    g. ISO 12207: Life Cycle Standard (optional)
3. **Requirements engineering**
    a. Functional and non-functional requirements
    b. Requirements engineering processes
    c. Requirements elicitation
    d. Requirements specification
    e. Requirements validation
    f. Requirements change
    g. Agile Requirements Engineering (optional)
    h. Requirements Management Tools (e.g., IBM Rational Doors) (Optional/TP)
4. **System modelling**
    a. Modelling Languages (Unified Modelling Language UML, OCL…)
    b. Context models
    c. Interaction models
    d. Structural models
    e. Behavioural models
    f. Model-driven architecture
5. **Architectural design**
    a. Architectural design decisions
    b. Architectural views
    c. Architectural patterns
    d. Application architectures
    e. Using Open Source, free, freemium, paid, and Enterprise software components (optional)
    f. Design Tools (e.g., Sparx Enterprise Architect) (Optional/TP)
6. **Design and implementation**
    a. Object-oriented design using the UML
    b. Design patterns
    c. Language and platform issues
    d. Refactoring
    e. Test Driven Development (TDD) (optional)
    f. Distributed Development and Agile Methods Scalability (optional)
    g. Development Tools (Optional)
        1. IDEs (e.g., Xcode, Eclipse, IntelliJ IDEA, NetBeans, Microsoft Visual Studio, Atom)
        2. Source Control Management (e.g.,GitHub)
        3. Release Orchestration (e.g., OpenMake)
        4. Collaboration (e.g., Jira, Trello, Slack)
7. **Software Testing**
    a. Unit Testing
    b. Integration and System Testing
    c. Static Confirmation
    d. Dynamic Testing
    e. Automated Testing
    f. Test-driven development
8. **Advanced Topics In Software Engineering**

a. Software Evolution and Maintenance
b. Security Engineering of Software Systems
c. Quantifying Software Specifications Using Formal Methods
d. Data Science for Software Engineers
e. Software Engineering Ethics

References

1. Ian Sommerville. Software Engineering, Pearson Edition, 2015.
2. Roger S. Pressman, Bruce Maxim. Software Engineering: A Practitioner's Approach 8th Edition, McGraw Hill, 2014.
3. Stephens, Rod. Beginning software engineering, Wrox, a Wiley Brand, 2015.
4. Systems Analysis and Design: An Object-Oriented Approach with UML 6th Edition, Alan Dennis, Barbara Wixom, David Tegarden, Wiley, 2020.
5. Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations. IT Revolution. 2021

Introduction to AI

Course Description :

This course is intended to expose the students to the general subject of Artificial Intelligence. In particular, it is meant to explain what it means for a computer to be “intelligent” and what aspects need to be modelled in terms of knowledge representation and reasoning mechanisms. Further, the module would teach students the various applications and research areas where artificial intelligence is used as the pivotal component for the aim to give students a glimpse of future research areas in this arena.

Prerequisite : Data Structures and Algorithms

Evaluation Method : Coursework (40%) + Final Exam (60%)

Course Content

1. **Introduction to AI**
   - 1.1. What Is AI?
   - 1.2. The Foundations of Artificial Intelligence
   - 1.3. The History of Artificial Intelligence
   - 1.4. The State of the Art
   - 1.5. Risks and Benefits of AI
2. **Intelligent Agents**
   - 2.1. Agents and Environments
   - 2.2. Good Behaviour: The Concept of Rationality
   - 2.3. The Nature of Environments
   - 2.4. The Structure of Agents
3. **Search**
   - 3.1. Search Algorithms
   - 3.2. Uninformed Search Strategies
   - 3.3. Informed (Heuristic) Search Strategies
   - 3.4. Tree search and graph search
   - 3.5. A* algorithm and its properties.
   - 3.6. Memory efficiency search algorithms
   - 3.7. Local Search and Optimization Problems
4. **Adversarial Search and Games**
   - 4.1. Game Theory
   - 4.2. Optimal Decisions in Games
   - 4.3. Heuristic Alpha--Beta Tree Search
   - 4.4. Monte Carlo Tree Search
   - 4.5. Stochastic Games
   - 4.6. Partially Observable Games
   - 4.7. Limitations of Game Search Algorithms
5. **Constraint Satisfaction Problems**
   - 5.1. Defining Constraint Satisfaction Problems
   - 5.2. Constraint Propagation: Inference in CSPs
   - 5.3. Backtracking Search for CSPs

5.4. Local Search for CSPs
5.5. The Structure of Problems

6. Knowledge Representation and Reasoning

6.1. Knowledge representation
6.2. Semantic networks
6.3. Frames and rules
6.4. Inference and Logic
6.5. Inheritance, forward and backward chaining
6.6. Reasoning Systems

7. Automated Planning

7.1. Definition of Classical Planning
7.2. Algorithms for Classical Planning
7.3. Heuristics for Planning
7.4. Hierarchical Planning
7.5. Planning and Acting in Nondeterministic Domains
7.6. Time, Schedules, and Resources
7.7. Analysis of Planning Approaches

8. Key Application Areas for AI

8.1. Expert system, decision support systems
8.2. Speech and vision
8.3. Natural language processing
8.4. Information Retrieval

References

1. Russel, S. and Norvig, P. Artificial Intelligence, A Modern Approach (4th Edition), Pearson Education Limited. 2020.
2. Luger, G. F., Artificial Intelligence - Structures and Strategies for Complex Problem Solving, Addison Wesley, 6th Edition, 2009.
3. Poole, D., Mackworth, A, Artificial Intelligence - Foundations of Computational Agents, Cambridge University Press, Second Edition, 2017.

Introduction to Business

Course Description

This course provides a comprehensive, integrated, and step-by-step approach to creating innovative and highly successful products. It focuses on the iterative process that motivates students toward the foundation of a business. This course breaks down the necessary process into various steps detailed in the course content.

Prerequisite :

Evaluation Method : Coursework (40%) + Final Exam (60%)

Course Content

1. **Introduction**
   - **1.1. I**ntroduction
   - 1.2. Getting Started
   - 1.3. Ideation/Brainstorming
2. **Market Segmentation**
   - 2.1. Market Segmentation
   - 2.2. Market Segmentation Narrowing
   - 2.3. Talk to customers in the segments to validate your assumptions in Segment Narrowing.
3. **Beachhead Market Strategy**
   - 3.1. Select one market segment to begin your efforts. Your target market.
   - 3.2. Analyzes the segment and further segment if needed
4. **Customer Profile**
   - 4.1. Develop an understanding of the role of end-user, decision making unit (economic buyer/saboteurs/influencers)
   - 4.2. Learn who ultimately makes the decision to buy
   - 4.3. Build an end-user profile (age/gender/income/education/geography/etc.)
   - 4.4. Build this persona/profile for the one target market you will focus.
   - 4.5. This is your Beachhead Market
5. **Calculate Total Addressable Market (TAM) of Beachhead Market**
   - 5.1. Understand Top down and bottom-up methods to analyze TAM
   - 5.2. Understand Total Addressable Market (TAM), Serviceable Addressable Market and Target Market
   - 5.3. Learn how to use internet and database to calculate size of market segments
6. **Full Life Cycle Use Case**
   - 6.1. Learn and draw how will the customer use your product
   - 6.2. Understand how does your product fit into the customer’s workflow or use pattern
7. **High Level Product Specification**
   - 7.1. Create a napkin drawing of your product
   - 7.2. Create a brochure of your product
   - 7.3. Design products at a high-level using words a 10-year-old child would understand
8. **Quantify Value Proposition and Competitive Advantage**
   - 8.1. Understand what value your customer derives from your product
   - 8.2. Create an ‘as-is’ and ‘will-be possible’ scenario
   - 8.3. What are the key differences between you and your competitors?
   - 8.4. How are your customers solving the problem today?

8.5. Plot X/Y diagram of competitors
8.6. Develop a matrix of customer requirements versus competitors

9. Identify the Next Ten Customers

9.1. Based upon the original customer persona, develop profiles of the next ten customers
9.2. Validate these customers
9.3. Build a progression of customers

10. Design a Business Model

10.1. Understand the different business models
10.2. Select the model or models for your customers
10.3. What is the customer expecting you to do?
10.4. What is the best method to capture the most value?
10.5. What is the competition doing?

11. Process to Acquire a Customer and a Sales Process

11.1. Understand what process to use to get, keep and grow customers
11.2. Understand how you will distribute your product to your customers
11.3. Develop the first pricing strategy.

12. Mapping the Sales Process

12.1. What marketing and sales methods will you use to acquire customers?
12.2. Understand the differences between earned and paid media

13. Lifetime Value of a Customer and Cost of Customer Acquisition

13.1. Calculate the Lifetime Value of a customer
13.2. What are your sales and marketing costs?
13.3. Calculate the Cost of Customer Acquisition
13.4. Analyse the relationship between the two measures

14. Calculate the Total Addressable Market for Follow-on Markets

14.1. Assess the potential for the additional markets examined earlier
14.2. Based on this assessment, develop a growth and expansion strategy
14.3. Leverage one successful market to enter the next

15. Project Presentation and Assessment

15.1. Each team will have 15 minutes to present their project.
15.2. The contents of the slides must include:
15.2.1. Title slide with team name and team members
15.2.2. Description of the product or service
15.2.3. Description of the problem being solved
15.2.4. Description of the customers and how they will use the product
15.2.5. Description of the market and market size
15.2.6. Description of the business model
15.2.7. Description of the beachhead market strategy
15.2.8. Description of the lifetime value of a customer and the cost of customer acquisition
15.3. Panel of judges will evaluate the presentations and assign proposed grade points
15.4. The professor will analyse the judges results and assign final grades to students

References

1. Bill Aulet. Disciplined Entrepreneurship: 24 Steps to a Successful Startup. Wiley. 2013.

Computer Architecture II

Course Description:

This course provides an introduction to computer organisation, systems programming and the hardware/software interface. Topics include instruction sets, computer arithmetic, datapath design, data formats, addressing modes, memory hierarchies including caches and virtual memory, I/O devices, bus-based I/O systems, and multicore architectures. Students learn assembly language programming and design a pipelined RISC processor.

Prerequisite: Computer Architecture I, Data Structures and Algorithms

Evaluation Method: Coursework (40%) + Final Exam (60%)

Course Content

1. **Introduction of RISC-V Architecture**
   - 1.1. Computer Model, Abstractions, and Technologies
   - 1.2. RISC-V Design Philosophy
   - 1.3. Registers
2. **RISC-V Instruction Set**
   - 2.1. Data Processing Instructions
   - 2.2. Load/Store Instructions
   - 2.3. Branch/Conditional Instructions
   - 2.4. Instruction formats and machine code
3. **RISC-V Programming Model**
   - 3.1. From Assembly to C
   - 3.2. C Pointers and Array
   - 3.3. Functions Calls and Stack (Recursion)
4. **RISC-V: Processor Design**
   - 4.1. Datapath
   - 4.2. The Controller
5. **Pipelining**
   - 5.1. Data Hazard and Resolution
   - 5.2. Exceptions
6. **Cache Management**
   - 6.1. Memory Hierarchy,
   - 6.2. Fully Associative Caches
   - 6.3. Direct-Mapped & Set-Associative Caches,
   - 6.4. Cache Performance
7. **Virtual Memory**
   - 7.1. Protection and paging
   - 7.2. Page tables and TLB

8. **I/O & interrupts**
    - 8.1. Memory Mapped I/O
    - 8.2. Polling
    - 8.3. Interrupts
    - 8.4. Direct Memory Access (DMA)

9. **Advanced Computer Architecture (Optional)**
    - 9.1. Multi-Core Processors, Coherency, and Consistency
    - 9.2. SMT
    - 9.3. GPUs
    - 9.4. Accelerators and Custom Architectures

References

1. John L. Hennessy, David A. Patterson, Computer Organization and Design. The Hardware/Software. Interface: RISC-V Edition. 2nd Edition. 2020.

Operating Systems

Course Description

The purpose of this course is to provide an overview of computer operating systems. Topics to be discussed include a brief history of OS's and their design and development. The course will start with a presentation of a Linux OS and how to manipulate a Linux-like workstation for development purposes. Then, the course will cover major OS components and the underlying algorithms and implementation techniques. Programming assignments will be done on Linux machines using C.

Prerequisite : Computer Architecture I

Evaluation Method : Coursework (40%) + Final Exam (60%)

Course Content

1. **Introduction to Linux Administration and Development**
   - 1.1. Operating systems definition, functions and development history
   - 1.2. Linux OS structure and functions
   - 1.3. Sessions and users management
   - 1.4. The Shell, Linux command syntax and basic commands
   - 1.5. Directories and files
   - 1.6. Files manipulation (naming, access, listing, deletion, editing, copy, move, comparison, fusion, visualisation, ...)
   - 1.7. Links
   - 1.8. Meta-characters
   - 1.9. I/O and errors redirecting
   - 1.10. The pipes
   - 1.11. Regular expressions and grep, find, sed commands
   - 1.12. make
     - 1.12.1. Makefile
     - 1.12.2. Syntax of dependence rules
     - 1.12.3. Macros
     - 1.12.4. Predefined rules and macros

2. **Program execution Basics**
   - 2.1. From a program to a process
   - 2.2. Process context and states
   - 2.3. Linux commands on processes
   - 2.4. UNIX process creation: fork
   - 2.5. Interruptions and process context switching
   - 2.6. CPU Scheduling: scheduling criteria and algorithms
   - 2.7. Threads (POSIX Threads programming)

3. **I/O Management**
   - 3.1. I/O hardware aspects
   - 3.2. I/O software aspects
   - 3.3. I/O in UNIX systems

3.4. I/O UNIX descriptors

4. Process Synchronisation

4.1. Process synchronisation using wait and waitpid primitives
4.2. Critical Section Problem
4.3. Hardware solutions
4.4. Semaphores
4.5. Threads mutex
4.6. Classic Synchronisation Problems
4.7. Programming classic synchronisation problems using POSIX Threads
4.8. Deadlocks modelling and handling

5. Memory Management

5.1. Logical versus Physical Address Space
5.2. Memory allocation strategies
5.3. Virtual Memory Management: paging and segmentation
5.4. Thrashing and pages allocation/replacement algorithms

6. File management systems

6.1. Files (Attributes, Operations, File types, Structure, Access methods)
6.2. UNIX File management system and inodes
6.3. Protection: UNIX file access control (chmod, umask)
6.4. Allocation methods
6.5. Disks (Structure, Scheduling)

References

1. Silberschatz, A., Galvin, P.B., and Gagne, G. (2018) Operating System Concepts (10th Edition), John Wiley & Sons, Inc.
2. Tanenbaum, A. and Bos, H. Modern Operating Systems (4th edition), Prentice Hall. 2014.