Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Technology

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

Structure & Syllabus

B.Tech. (Computer Engineering)

Pattern 'B-19' Effective from Academic Year 2019-20

Prepared by: - Board of Studies in Computer Engineering

Approved by: - Academic Board, Vishwakarma Institute of Technology, Pune

Chairman – BOS **Chairman – Academic Board**

Vision of the Institution

"To be globally acclaimed Institute in Technical Education and Research for holistic Socio-economic development".

Mission of the Institution

- To ensure that 100% students are employable and employed in Industry, Higher Studies, become Entrepreneurs, Civil / Defense Services / Govt. Jobs and other areas like Sports and Theatre.
- To strengthen Academic Practices in terms of Curriculum, Pedagogy, Assessment and Faculty Competence.
- Promote Research Culture among Students and Faculty through Projects and Consultancy.
- To make students Socially Responsible Citizen.

Vision of the Department

"To be a leader in the world of computing education practising creativity and innovation".

Mission of the Department

- To ensure students' employability by developing aptitude, computing, soft, and entrepreneurial skills
- To enhance academic excellence through effective curriculum, blended learning and comprehensive assessment with active participation of industry
- To cultivate research culture resulting in knowledge-base, quality publcations, innovative products and patents
- To develop ethical consciousness among students for social and professional maturity to become responsible citizens

List of Programme Outcomes [PO]

PO	PO Statement
PO1	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the

professional engineering practice.

PO7 **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8 **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9 **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

List of PSO Statement

PSO1 Select and incorporate appropriate computing theory principles, data structures and algorithms, programming paradigms to innovatively craft scientific solution addressing complex computing problems.

PSO2 Adapt to new frontiers of science, engineering and technology by getting acquainted with heterogeneous computing environments and platforms, computing hardware architectures and organizations through continuous experimentation.

PSO3 Conceive well-formed design specifications and constructs assimilating new design ideas and facts for identified real world problems using relevant development methodologies and practices, architecture styles and design patterns, modeling and simulation, and CASE tools.

PSO4 Exercise research and development aptitude focusing knowledge creation and dissemination through engineering artifacts construction, preparation and presentation of engineering evidences using procedures, techniques, guidelines, and standards considering technology migration and evolution.

Program Educational Objectives (PEOs)

- Demonstrate application of sound engineering foundations to be a committed technology workforce
- Apply mathematical and computing theory knowledge base to provide realistic computer engineering solutions
- Exhibit problem-solving skills and engineering practices to address problems faced by the industry with innovative methods, tools, and techniques
- Develop professional and ethical practices adopting effective guidelines to acquire desired soft skills in the societal and global context
- Aim for continuing education and entrepreneurship in emerging areas of computing

INDEX

SN	Particular	Page No
1	Second Year Module -III Structure and Content	10
2	Second Year Module -IV Structure and Content	24
3	Third Year Module -V Structure and Content	37
4	Third Year Module -VI Structure and Content	71
5	Final Year Module -VII Structure and Content	90
6	Final Year Module -VIII Structure and Content	147

MODULE III

Issue 1, Rev. 1 , Dt. 01/07/2018

Title: Course Structure **FF No. 653**

Branch: Computer **Year:** S.Y. **A.Y.:** 2019-20 **Module:** III **Pattern:** B-19.

Subject No.	Subject Code	Subject Name	Teaching Scheme (Hrs/Week)			Examination Scheme							Credits
			Theory	LAB	Tut.	ISA				ESA		Total	
						Assignment (%)	Lab Assessment (%)	MSE (%)	GD/ PPT (%)	Viva/Lab Exam (%)	ESE (%)		
S1	CS2034	Data Structures-I	3	4	-	10	30	15	10	20	15	100	5
S2	CS2036	Digital Electronics and Logic Design	3	2	-	10	30	15	10	20	15	100	4
S3	CS2033	Computer Organization	3	-	-	10		30	10	20	30	100	3
S4	CS2038	Object Oriented Programming	3	2	-	10	30	15	10	20	15	100	4
S5	CS2031	Engineering Design & Innovation1	1	6	-	-	50	-	-	50	-	100	4
S6	CS2005	Discrete Mathematics	2	-	--	10	-105	30	10	20	30	100	2
		TOTAL	**15**	**14**	**0**	**50**	**140**	**105**	**50**	**150**	**105**	**600**	**22**

CS2034: DATA STRUCTURES - I

Credits: 5 **Teaching Scheme: Theory: 3 Hrs / Week**
Lab: 4 Hours/Week

Section 1: Topics/Contents

Arrays : Definition, Initialization, Memory map of allocation, 1D array manipulation, Two Dimensional array, Memory map of two dimensional array, 2D array Initialization, Manipulation. Passing arrays to functions and returning. Multi Dimensional arrays. Strings, String Manipulations using user defined functions and built in functions.

Structures and Unions: User defined Data types, **Structures:** Syntax, Initialization, Accessing Structure data. Arithmetic operations and structures. **Array and Structures:** Array as data member to structures, array of structures. Nested structures, **Structures and Functions:** Passing and returning structure element to functions. **Unions:** Definition, Syntax, Comparison between structures and unions, Nested Unions, Nesting of structures and Unions.

Pointers: Basics, Pointer Data types, Pointer Manipulations, Pointer arithmetic, Pointer to pointers, Void pointer, Dangling Pointer. Pointer to an array, Array of pointers, Pointers and structures, Function Pointers. **Functions and Pointers:** Call by Value, Call by Reference, Passing arrays and structures to functions using pointers.

Section2: Topics/Contents

Fundamentals of Files : Definition, Need of Files, Organization of files, Create, open, manipulate, and close files. Different opening modes of a File. Text files and Binary Files, Different inbuilt functions for manipulation of files. Simulation of commands Copy, Move, etc. Sequential access file : Creation and Manipulations, Direct access file : Creation and Manipulations,

Searching and Sorting : Time & Space Complexity Analysis. Sorting Techniques: Insertion, Bucket, Merge, Quick and heap sort. Binary search, Fibonacci search

Stack and Queue: Types of Data Structures : Primitive, Non primitive, Linear, Nonlinear, Static, Dynamic Data Structures **.**

Stack: Fundamentals of stack, representation using array, Applications of stack: Recursion, Validity of parentheses, Expression conversions and evaluations, mazing problem.

Queue: Fundamentals of queue, representation using array, Types of queues, Applications of Queue: Job Scheduling, Josephus problem etc.

List of Practicals:

1. Assignments based on application Arrays and Multidimensional arrays.
2. Assignments based on Structures and Unions.
3. Assignments based on Functions and Pointers.

4. Assignment based on File operations.
5. Assignment based on Sequential and Direct Access file.
6. Assignment based on Sorting and Searching.
7. Assignment based on Stack Application
8. Assignment based on Expressions Conversion using Stack.
9. Assignment based on Queue Application.
10. Assignment based on application of Queues in different Scheduling Algorithm.

List of Project areas: (For THP, TLP courses)

1. Job Scheduling.
2. String processing.
3. Dictionary and Search engines.
4. Modeling the real world problems using stacks and Queues.
5. Use of Multidimensional arrays in different fields.

Text Books:

1. Fundamentals of Data Structures in C", E. Horwitz , S. Sahani, Anderson-Freed, Second Edition, Universities Press.

2. Data structures using C and C++", Y. Langsam, M.J. Augenstein, A.M.Tenenbaum, Pearson Education, Second Edition

Reference Books:

1. An Introduction to data Structures with applications", J. Tremblay, P. soresan, TMH Publication, 2nd Edition.2.

Course Outcomes:

The student will be able to –

1. To interpret and diagnose the properties of data structures with their memory representations and time complexity analysis.
2. To use Structures and union data structures with their applications
3. To demonstrate the use of different files and to perform various operations on File data structures.
4. To apply operations like searching, sorting using different algorithms and perform their analysis.
5. To implement and demonstrate the use of Stack and Queue data structure in real life scenarios.
6. To design and analyze the appropriate data structure for real world application.

CS2036: DIGITAL ELECTRONICS & LOGIC DESIGN

Credits: 4 **Teaching Scheme: Theory: 3 Hours / Week**
Lab: 2 Hours/Week

Section 1: Topics/Contents

Combinational logic circuits: Codes : BCD code, Excess-3 code, Gray code. Code conversions, Multiplexers & De-multiplexers, design examples: code converter. Encoder: Priority encoders. Decoders: 74138. BCD adder and sub tractor. Parity generator and checker, Digital comparator: 7485.
Introduction of flip-flop (F.F): 1 bit memory cell, clocked S-R F.F., J-K F.F. race around condition, M/S J-K F.F, flip-flop truth table, excitation table, flip-flop conversion, flip-flop characteristics. T and D F.F,
Sequential logic circuits: Design of 4 bit UP-Down ripple counter using J-K flip-flop,
Design of Synchronous 3 bit up/down counter, mod-n counters (IC -7490),Moore/Mealy M/c's: representation techniques, state diagrams, state tables, state reduction, state assignment, implementation using flip-flops. Sequence generator. Shift register (modes of operation), 4 bit bi-directional universal shift register, application of shift registers (Ring counter, Sequence generator, Johnson's counter.),Sequence Detector.

Section2: Topics/Contents

ASM & RTL: ASM charts, notations, design of simple controller, multiplexer controller method. RTL notations and implementation. Examples on ASM, RTL.
Logic Families: Characteristics of Digital ICs: Speed, Power dissipation, fan-out, current and voltage parameters, noise margin, operating temperature etc., TTL: Operation of TTL NAND gate, Standard TTL, TTL Characteristics, Active pull-up, Wired-AND, totem pole, open collector, Unconnected Inputs. CMOS Logic: CMOS Inverter, CMOS NAND and NOR, CMOS characteristics. Wired-logic, Unconnected Inputs, Open-Drain Outputs, Comparison of TTL and CMOS, interfacing TTL to CMOS. Interfacing CMOS to TTL, Tri-state logic: tri-state buffers, inverters.
Programmable Logic Devices : Programmable Logic array: Input, Output Buffers, AND, OR, Invert/Non-Invert Matrix, Programming the PLA, Applications of PLAs to implement combinational and sequential logic circuits Introduction to :FPGA, CPLD. Comparison of FPGA and CPLD. Introduction to VHDL: Modeling Digital systems. Different types of modeling: Structural modeling, Behavior modeling and Data Flow modeling.Design and programming of combinational and sequential circuit based on VHDL modeling types.

List of Practical :

1. Verification of Logical Gates and Boolean Algebra.
2. Code converters, e.g. Excess-3 to BCD and vice versa using logical gates.
3. Multiplexer - e.g. 16:1 Mux using 4:1 Mux (IC 74153).
4. Decoder – e.g. 2 bit comparator (IC 74138).

5. BCD adder –using IC 7483.
6. Conversion of flip-flops. e.g. JK to D, T.
7. Ripple (asynchronous) mod –N counter using J-K F-F.
8. Ripple (asynchronous) mod –N counter using IC 7490.
9. Synchronous 2 bit Up /down counter using JK flip-flop.
10. Sequence generator using JK flip-flop
11. Pseudo random number generator using 74194.(universal shift register)
12. Sequence detector (Moore ckt) using JK flip-flop
13. Sequence detector (Mealy ckt) using JK flip-flop
14. Simple ASM using multiplexer controller method using Simulator.
15. Design of simple combinational circuit: half adder and subs tractor using VHDL language

List of Project areas:

1. Combinational Circuit
2. Sequential Circuits

Text Books

1. *R.P. Jain, "Modern Digital Electronics,"3rd Edition, Tata McGraw-Hill, 2003, ISBN 0 - 07 - 049492 – 4.*
2. *J. Bhaskar, Englewood Cliffs, "A VHDL Primer," 2nd Edition Prentice Hall, 1994, ISBN-13: 978- 0131814479.*

Reference Books

1. *M. Mano, "Digital Design", 3rd Edition, Pearson Education, 2002, ISBN - 81 - 7808 – 555 – 0.*
2. *A. Malvino, D. Leach, "Digital Principles and Applications", 5th Edition, Tata McGraw Hill, 2003, ISBN 0 - 07 - 047258 – 05.*

Course Outcomes:

Student will be able to –

1. Construct combinational circuits.
2. Design sequential circuits.
3. Develop the applications of sequential circuits.
4. Analyze internal structure of logic gates.
5. Select an appropriate design and give solution for the Programmable Devices.
6. Make use of advance programming techniques in Digital systems.

CS2033: COMPUTER ORGANIZATION

Credits: 3 **Teaching Scheme: Theory: 3 Hours / Week**

Section 1: Topics/Contents

Computer Evolution & Arithmetic: Organization & Architecture, Structure & Function, and Brief History of computers, Von Neumann Architecture, Integer Representation: Fixed point & Signed numbers. Integer Arithmetic: 2's Complement arithmetic, multiplication, Booth's Algorithm, Division Restoring Algorithm, Non Restoring algorithm, Floating point representation: IEEE Standards for Floating point representations.

Processor Design: CPU Architecture, Register Organization, Instruction types, Types of operands, Instruction formats, addressing modes and address translation. Instruction cycles. RISC Processors: RISC- Features, CISC Features, Comparison of RISC & CISC Superscalar Processors. Case Study of Processor.

Control Unit:

Fundamental Concepts: Single Bus CPU organization, Register transfers ,Performing an arithmetic/ logic operations, fetching a word from memory, storing a word in memory, Execution of a complete instruction. Micro-operations, Hardwired Control, Example-Multiplier CU. Micro-programmed Control: Microinstructions, Microinstruction-sequencing: Sequencing techniques, Micro-program sequencing, Multi Bus CPU organization.

Section2: Topics/Contents

Memory Organization: Need, Hierarchical memory system, Characteristics, Size, Access time, Read Cycle time and address space. Main Memory Organization: ROM, RAM, EPROM, E^2PROM, DRAM, Design examples on DRAM, SDRAM, DDR3, Cache memory Organization: Address mapping, Replacement Algorithms, Cache Coherence. Virtual Memory: Address Translation Virtual to Physical, Segmentation, Paging.

Pipelining: Basic concepts: role of cache memory, pipeline performance.

Data hazards: operand forwarding, handling data hazards in software, side effects.

Instruction hazards: unconditional branches, conditional branches and branch prediction.

Performance considerations: effect of instruction hazards, number of pipeline stages

Parallel Processing:

Necessity of high performance, Constraints of conventional architecture, Parallelism in Uniprocessor system, Evolution of parallel processors, Architectural Classification, Flynn's, Fengs, Handler's Classification, Multiprocessors architecture basics, Parallel Programming Models: Shared memory, Message passing, Performance considerations : Amdahl's law, performance indications.

Text Books

1. William Stallings, "Computer Organization and Architecture: Designing for Performance", 7th Edition, Pearson Prentice Hall Publication, ISBN 81-7758-9 93-8.
2. C. Hamacher, V. Zvonko, S. Zaky, "Computer Organization", 5th Edition, Tata McGraw Hill Publication, ISBN 007-120411-3.
3. Kai Hwang, "Advanced Computer Architecture", Tata McGraw-Hill ISBN 0-07-113342-9

Reference Books

1. Hwang and Briggs,"Computer Architecture and Parallel Processing", Tata McGraw Hill Publication ISBN 13: 9780070315563.
2. A. Tanenbaum, "Structured Computer Organization", Prentice Hall Publication, ISBN 81 – 1553-7, 4th Edition.

Course Outcomes:

Student will be able to -

1. Understand the structure, function and characteristics of computer systems.
2. Describe the working of Central Processing Unit and RISC and CISC Architecture.
3. Explore the knowledge about Control Unit Design.
4. Design memory with due consideration of tradeoffs and performance issues.
5. Analyze a pipeline for consistent execution of instructions with minimum hazards.
6. Acquaint the advanced concepts of computer architecture.

CS2038: **OBJECT ORIENTED PROGRAMMING**

Credits: 3 **Teaching Scheme: 3hrs /Week**
Lab: 2 Hours/Week

Section 1: Topics/Contents

Object Oriented Paradigm: Limitations of procedural thinking. Need for Object Oriented Paradigm. Reasons for OOP's popularity. Overview of Object Oriented Thinking - agents and communities, classes and instances, information hiding, communicating with agents: message and methods, message vs procedure calls, responsibilities, objects as service providers, class hierarchies, inheritance, abstract classes, method binding and overriding, polymorphism. Case Studies. Object Oriented Concepts- Hierarchy, Encapsulation, Modularity, Abstraction. Principles for object-oriented problem solving. Designing good programs - problem specification, problem decomposition, class design, data design, method design, algorithm design, coding, testing, debugging, revising. Case Studies.

Abstraction and Encapsulation: Objects, classes, object references, creating an object, heap storage, scope, fields and methods. Passing values and references. static keyword (for fields and methods). this keyword. this and static. Initialization & Cleanup: constructors, method overloading, finalization and garbage collection, member initialization. Hiding the implementation - package, access specifiers.

String Objects: string basics, finding things within a string, retrieving parts of strings, processing each character in a string, comparing strings, parsing text, string processing applications.

Reusing classes - composition and inheritance. Method overriding. Composition vs inheritance. protected access specifier. Upcasting. final classes and methods. Intialization and class loading in case of inheritance.

Polymorphism - static and dynamic. Concept of dynamic polymorphism. Method-call binding. Producing the right behavior. Abstract classes and methods. Constructors and Polymorphism. Inheritance based design. Interfaces.

Section2: Topics/Contents

Error handling with exceptions - basic exceptions, catching an exception, creating your own exceptions, finally keyword, exception hierarchy

Different types of inheritance in C++, polymorphism in C++, operator overloading in C++, virtual keyword in C++, Exception handling in C++, **Similarities and differences between C++**

and Java: destructors, access modifiers, inheritance, polymorphism, garbage collection, exception handling.

Input and Output – Objects for I/O. File, Streams, Readers and Writers. File Management / Processing, RandomAccessFile, New IO, Object serialization and deserialization.

Object Oriented Design of Graphical User Interfaces: Applets. Applet vs Application. AWT to Swing. Getting started with Swing. Model, View, Controller (MVC) architecture. Using Swing components - labels, buttons, check boxes, radio buttons, combo boxes, dialogs, file choosers, text components, list etc. Controlling layouts. Event handling, event listeners and call backs. Concept of inner classes. Handling mouse and keyboard events.

Concurrency: Motivation, threads, thread life cycle, thread Priority, Thread Methods. Issues with concurrency - improperly accessing resources, race conditions, colliding over resources, resolving shared resource contention, critical sections. Cooperation between threads (inter thread communication). Solving Producer-Consumer using object oriented principles.

Object Oriented Design Principles: SOLID – Single Responsibility Principle, Open-Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, Dependency Inversion Principle

Multithreading in C++, Concept of Deadlocks.

List of Practicals

1. Write a C++ program to implement the concept of objects, classes, constructors, destructors.
2. Write a C++ program to implement the concept of Inheritence and polymorphism.
3. Write a C++ program to use the concept of generic programming (generic functions and generic classes)
4. Write a C++ program to create a student class with details like Roll no, Name, Marks of five subjects, percentage, class (First, Second, etc) have following functions: parameterized constructor, destructor, Display, Calculate percentage and grade using inline function.
5. Creation & deletion of objects from bag
6. Friend function and function overloading
7. String operations using operator overloading
8. Multiple inheritance
9. Virtual function
10. File operations
11. Templates
12. Write a program to get a number from user and do addition of digits of given number. (Use of cout, cin, for)(ip=3214 op=10)
13. Write a program to implement class for Book details as follows
    Private: bookid, book_name, author, price
    Public: get_details(), print_details()
    Get details of some books from user and print in tabular form. Also find the total prize of all the books.
14. Write a program to implement Complex class as follows
    Private:real_value, imag_value

Public:Parameterized constructor(1 and 2 parameter), copy constructor, getter and setter functions.(Use of constructors and destructors)

15. Write a program to extend complex class with following details
    Public: add_complex(), subtract_complex(), multiply_complex(), print_complex(), getcount_complex() functions (Use of member functions and static variable)
16. Write a program to extend complex class with following details
    Public: Operator functions for +, - , multiply and print complex numbers. (Operator overloading using member and non-member functions)
17. Write a program to implement following inheritance hierarchy
    Student(name) -> Comp_Student(int Roll_Number), IT_Student(int Roll_Number)
    Comp_Student -> From_DSE_Student(float Diploma_Marks),
    From_FY_Student(float CPI) (5 classes in total)
    Keep all member variables private.
    Provide public member functions to set and get data.
18. Write a program to extend Book class with dynamic memory allocation.
    (Use of "new" keyword and object pointers)
19. Write a program to implement following class hierarchy using inheritance, object pointers and virtual functions.
    Animal(eat())->Dog(eat())->Labrador(eat())
    Every class is having eat() method. Apply concept of run-time polymorphism.
20. Write a program to extend Book class to get book details from user and store data to a file. Write functions to store data to file and get data from file.
21. Design a car racing game using object oriented concepts. Construct class Car with member functions like run_car(), movecar_left(), movecar_right() etc.
22. Extend the car racing game to provide support for User profile creations and maintain the levels of each user on secondary storage. Your program must be able to load previous state of user like levels completed, total score etc.

Text Books:

1. "An Introduction to Programming through C++", Abhiram G Ranade, McGrawHill Education
2. "The C++ Programming Language", Bjarne Stroustrup, 4th Edition, Addison-Wesley Pearson Education

Reference Books:

1. "Thinking In Java – The Definitive Introduction to Object-Oriented Programming in the Language of the World-Wide Web", Bruce Eckel, Fourth Edition, Pearson Education, Inc.
2. "Java, java, Java – Object-Oriented Problem Solving", R. Morelli and R. Walde, 3rd edition, Pearson Education, Inc.
3. "Java – The Complete Reference", Herbert Schildt, 9th Edition, Oracle Press

Course Outcomes:

The student will be able to –

1. Apply object oriented thinking towards solving a real world problem

2. Evaluate the advantages of object-oriented approach over procedural approach of solving a problem

3. Implement a given object oriented design into code using an object-oriented programming language

4. Effectively use the object-oriented features of an object-oriented programming language towards developing a software solution

5. Assess the quality of a given design based on the well established principles of object oriented design.

CS2005: DISCRETE MATHEMATICS

Credits: 02 **Teaching Scheme: 2 Hours / Week**

Section 1: Topics/Contents

Propositional Logic, Applications of Propositional Logic, Propositional Equivalences, Predicates and Quantifiers, Nested Quantifiers, Rules of Inference, Sets, Set Operations, Cardinality of Sets, Countable, Uncountable sets, Russell's paradox, Functions, Some Important Functions, Relations, n-ary relations, Their Properties, Matrix Representation, Equivalence Relations, Partitions of a set, Lexicographic order, Posets, chains, antichains, Hasse Diagram, Lattices, Topological Sorting, Countable, uncountable sets. **Basic Counting Principles:** Sum rule and Product rule, Inclusion-Exclusion Principle, Permutation and Combination, Binomial Identities, Pascal's triangle, Pigeon-hole principle, generalized pigeon-hole principle, Ramsey theory, Mathematical Induction

Section2: Topics/Contents

Sequences and Summation, Recurrence relations, Modeling with Recurrence Relations, (Fibonacci numbers, Catlan numbers, Derangements, Tower of Hanoi, partitions) Solving Linear and non linear Recurrence Relations, Solving recurrence relations, Solving recurrence relations via generating functions

Graph Theory: Basic terminology of graphs, Types, Matrix representations, Graphs and Graph Models, Subgraphs, Representation of graphs, Konigsberg bridge problem, Eulerian graphs, Eulerian graphs and a characterization, Graph isomorphism, cliques and independent sets. Connectivity, connected components, cut edges, cut vertices, Planar graphs, Dual graph, graph coloring, Trees, Rooted Trees, Applications of Trees

Text Books: ***(As per IEEE format)***

1. *Discrete Mathematics and its applications with combinatorics and graph theory, by Kenneth H Rosen. Special Indian Edition published by Tata McGraw-Hill.*
2. *Introduction to Graph Theory, 2nd Edition, by Douglas B West. Eastern Economy Edition published by PHI Learning Pvt Ltd.*
3. *Discrete Mathematics, 2nd Edition, by Norman L Biggs. Indian Edition published by Oxford University Press.*

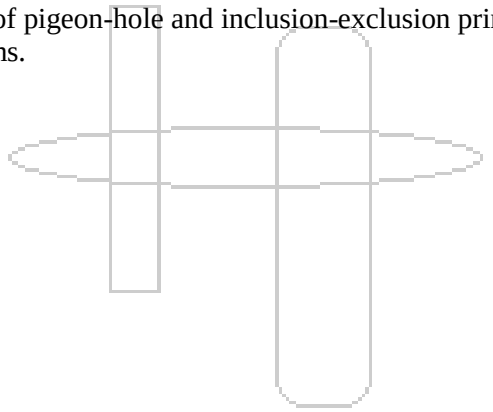
Reference Books: ***(As per IEEE format)***

1. *"Applied Combinatorics" by Alan Tucker (Wiley Publishing company)*
2. *"Combinatorics: Topics, techniques, algorithms" by Peter J. Cameron (Cambridge University Press)*
3. *Graph Theory by Reinhard Diestel (Springer Verlag Publishing Company)*
4. *Introduction to Graph Theory by Douglas B. West (Prentice-Hall publishers)*

Course Outcomes:

The student will be able to –

2. Reason mathematically about elementary discrete structures (such as functions, relations, sets, graphs, and trees) used in computer algorithms and systems
3. Describe the elementary properties of modular arithmetic and their applications in Computer Science like cryptography.
4. Summarize graph theory fundamentals and their applications
5. Develop recurrence relations for a wide variety of interesting problems
6. Express mathematical properties via the formal language of propositional and predicate logic
7. Demonstrate use of pigeon-hole and inclusion-exclusion principle in solving elegant and important problems.



MODULE IV

Issue 1, Rev. 1 , Dt. 01/07/2018

Title: Course Structure **FF No. 653**

Branch: Computer **Year:** S.Y. **A.Y.:** 2019-20 **Module:** IV **Pattern:** B-19.

8.

Subject No.	Subject Code	Subject Name	Teaching Scheme (Hrs/Week)			Examination Scheme							Credits
			Theory	LAB	Tut.	ISA				ESE		Total	
						Assignment (%)	Lab Assessment (%)	MSE %)	GD/PPT (%)	Viva/Lab Exam (%)	ESE (%)		
S1	CS2035	Data Structures-II	3	2	-	10	30	15	10	20	15	100	4
S2	CS2040	Data Communication	2	2	-	10	30	15	10	20	15	100	3
S3	CS2009	Database Management Systems	3	2	-	10	30	15	10	20	15	100	4
S4	CS2039	Microprocessors and Interfacing	3	2	-	10	30	15	10	20	15	100	4
S5	CS2032	Engineering Design & Innovation 2	1	6	-	-	50	-	-	50	-	100	4
S6	CS2037	Automata Theory	3		--	10	-	30	10	20	30	100	3
S7	CS2222	**General Profeciency 2											-
		TOTAL	**15**	**10**	**2**	**50**	**170**	**90**	**50**	**150**	**90**	**600**	**22**

FF No. :654

CS2035: DATA STRUCTURES - II

Credits: 4 **Teaching Scheme: Theory: 3 Hrs / Week**
Lab: 2 Hours/Week

Section 1: Topics/Contents

Linked Lists: Linked Lists using arrays, Dynamic memory allocation, Singly Linked Lists, Doubly linked Lists, Circular liked lists, and Generalized linked lists, Applications: Stack & Queue using linked list, Polynomial Manipulation using linked list.

Trees: Basic terminology, Normal trees and Binary trees, representation of Binary trees using array and linked list, Construction of an Expression Tree, Tree Traversals: Recursive and Non recursive In Order, Pre Order, Post order , traversals of a tree. BFS, DFS of trees. Binary Search trees(BST): Creation of BST, Insertion, deletion of a node from BST. Finding Height , leaf nodes and mirror of a tree.

TBT: Threaded Binary tree, Types of TBT, Construction of TBT from a given expression and perform traversals.

Section2: Topics/Contents

Advance Trees: AVL trees, Rotations on AVL trees. B- Trees, Red Black Trees.

Graphs: Terminology and representation, Traversals, Connected components and Spanning trees: Prims and Kruskal's Algorithm, Shortest Paths and Transitive Closures: Single Source All destinations (Dijkstra's Algorithm), all pair shortest path algorithm, Topological Sort.

Hashing techniques: Hash table, Hash functions, and Collision, Cuckoo Hashing, Overflow Handling Evaluation of Overflow techniques. Linear Probing without replacement, Linear Probing with replacement, Linear Probing without replacement with chaining, Linear Probing with replacement with chaining, Quadratic Probing, Double Hashing, Rehashing. Indexed sequential file: Creation and Manipulations.

List of Practicals:

1. Assignment based on different operations on linked list.
2. Assignment based on Trees.
3. Assignment based on BST traversals.
4. Assignment Based on TBT operations.
5. Assignment based on AVL tree.
6. Assignment Based on B-Trees and Red -Black Trees.
7. Assignment based on DFS and BFS.
8. Assignment based on MST.
9. Assignment Based on Shortest path problem.

10. Assignment based on Hashing.

List of Project areas: (For THP, TLP courses)

1. Application of trees in real world applications.
2. B-Trees and Red-Black trees in database.
3. Dictionary and Search engines.
4. Modeling the real world problems using Graphs.
5. Shortest path algorithms in real world applications.
6. Database management system using file structures and Hashing techniques.

Text Books:

1. Fundamentals of Data Structures in C", E. Horwitz , S. Sahani, Anderson-Freed, Second Edition, Universities Press.

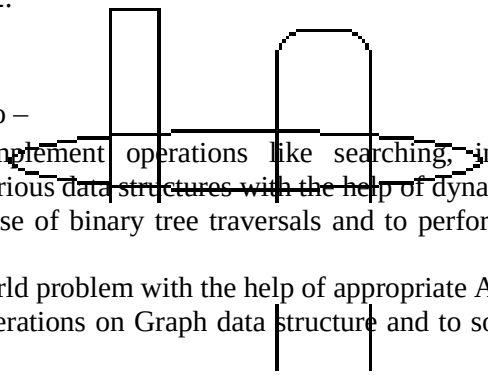
2. Data structures using C and C++", Y. Langsam, M.J. Augenstein, A.M.Tenenbaum, Pearson Education, Second Edition

Reference Books:

1. An Introduction to data Structures with applications", J. Tremblay, P. soresan, TMH Publication, 2nd Edition.2.

Course Outcomes:

The student will be able to –



1. To interpret and implement operations like searching, insertion, deletion, traversing mechanism etc. on various data structures with the help of dynamic storage representation.
2. To demonstrate the use of binary tree traversals and to perform various operations on non-linear data structures.
3. To Model the real world problem with the help of appropriate Advanced trees data structure.
4. To implement the operations on Graph data structure and to solve the applications of Graph data structure.
5. To manipulate the problems of indexing by applying various hashing techniques.
6. To design and analyze the appropriate data structure for real world application.

CS2040: DATA COMMUNICATION

Credits: 3 **Teaching Scheme:Theory:2 Hours / Week**
Lab:2 Hours / Week

Section 1: Topics/Content

Basics of Data Communication: Communication Model: Source, Transmitter, Transmission System, Receiver, Destination, Data Terminal Equipment (DTE), Data Communication Equipment (DCE). Data Communication System Tasks: Signal Generation, Interface, Data Synchronization, Exchange Management, Transmission System Utilization, Error Detection and Correction, Flow Control, Addressing, Routing. **Data Transmission:** Transmission Configurations: Point to Point and Multipoint. Transmission Modes: Synchronous and Asynchronous. Transmission Methods: Serial and Parallel. Communication Modes: Simplex, Half Duplex, Full Duplex, Time Domain Concepts: Continuous signal, discrete signal periodic signal. Frequency Domain Concepts, Bandwidth, Data Rate, Channel Capacity, Error Rate, Noise. Nyquist Sampling Rate, Shannon Channel Capacity, SNR. **Line Coding:** Unipolar NRZ, Polar NRZ, NRZ Inverted, Bipolar Encoding, Manchester Encoding, Differential Manchester Encoding. **Modulation:** Analog Modulation: Amplitude, Frequency, Phase. Pulse Modulation Techniques: PCM, PAM, PWM, PPM. Digital Modulation: ASK, FSK, MSK, GMSK, PSK, QAM, CPM

Section2: Topics/Content

Physical Layer: Introduction: LAN, MAN, WAN, PAN. Reference Models: OSI, TCP/IP. Design Issues: OSI Model and Layers. Network Architectures: Client-Server; Peer To Peer. Network Types: Infrastructure and Ad-hoc mode. Transmission Methods: Broadcasts, Point-To-Point. Transmission Mediums: CAT5, 5e, 6, OFC. Network Topologies: Star, Ring and Hierarchical. Network Devices: Bridge, Switch, Modem, Router and Access Point. **Medium Access Control Layer:** Channel Allocation: Static and Dynamic, Multiple Access Protocols: Pure and Slotted ALOHA, CSMA, WDMA. IEEE 802.3 Standard: Ethernet, Wiring Schemes and Frame Formats, CSMA/CD, Binary Exponential Back -off Algorithm. High Speed Standards: Fast Ethernet, Gigabit Ethernet Wireless Standards: Radio Spectrum, Frequency Hopping (FHSS) and Direct Sequence (DSSS), IEEE 802.11a/b/g/n and IEEE 802.15 and IEEE 802.16 Standards, CSMA/CA

List of Practical's:

Operating System recommended: - 64-bit Open source Linux or its derivative

Programming tools recommended: - Android Studio 3.x, JAVA

List of Laboratory Assignments:

Assignments on Line Encoding:

1. Unipolar NRZ, Polar NRZ, NRZ Inverted
2. Bipolar Encoding
3. Manchester Encoding, Differential Manchester Encoding
4. Serial Communication: RS-232 Signaling
5. Serial Communication: USB Signaling

Assignments on Modulation:

1. Modulation and demodulation: PCM for Voice Communication
2. Modulation and demodulation: FSK and BFSK modulation of digital signals onto an RF
3. Modulation and demodulation: MSK and GMSK for Mobile Communication
4. Modulation and demodulation: PSK for LANs, RFID and Bluetooth
5. Modulation and demodulation: QAM for TV Transmission

Assignments on Broadband Communication:

1. Frequency Hopping Spread Spectrum (FHSS)
2. Direct Sequence Spread Spectrum (DSSS)

Text Books: ***(As per IEEE format)***

1. *Fourauzan B., "Data Communications and Networking", 5th edition, Tata McGraw- Hill, Publications, 2006*
2. *Andrew S. Tenenbaum, "Computer Networks",5th Edition, PHI, ISBN 81-203-2175-8.*

Reference Books: ***(As per IEEE format)***

1. Kurose, Ross "Computer Networking a Top Down Approach Featuring the Internet", Pearson; 6th edition (March 5, 2012), ISBN-10: 0132856204
2. *Matthew S. Gast "802.11 Wireless Networks", O'Reilly publications; 2nd Edition.*
3. *C. Siva Ram Murthy and B. S. Manoj, "Ad Hoc Wireless Networks: Architectures and Protocols" Prentice Hall, 2004*
4. *Holger Karl and Andreas Willig, "Protocols and Architectures for Wireless Sensor Networks", Wiley, ISBN: 0-470-09510-5*

Course Outcomes:

The student will be able to –

1. Select line encoding methods for digital communication
2. Choose serial communication signaling for digital communication
3. Analyze modulation and demodulation methods for digital communication
4. Understand basic principle of transmission mediums for wired and wireless networks
5. Decide on physical layer technology for wired and wireless networks

CS2009: DATABASE MANAGEMENT SYSTEMS

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
: Lab 2 Hours / Week

Section 1: Topics/Contents

Introduction: Need of Database Management System, Evolution, Data Abstraction, Data Independence, System Architecture of DBMS, Life cycle of relational database, Codd's Twelve Rules for Relational DBMS; Data Models: Entity Relationship (ER) Model, Extended ER Model, Relational Data Model, Object Oriented Data model, Semi-structured Data Model: DTD or XML Schema

Database Design: Normalization: Need, Functional Dependency, Inference Rules, FD Closure, Minimal Cover, Decomposition Properties, Normal Forms (upto BCNF), Multi-valued Dependency (4NF), Relational Synthesis Algorithm, Trade – off

Query Languages: Formal Relational Query Languages: Relational Algebra, Tuple Relational Calculus, Domain Relational Calculus; SQL: DDL, DML, Select Queries, Join Queries, Subqueries, Date-Timestamp, String and Numerical Functions, DCL-Security and Authorization; PL/SQL: Procedure, Function, Trigger; Mapping of Relational Algebra to SQL

Section2: Topics/Contents

Storage and Querying: Storage and File structures, Indexed Files, Single Level and Multi Level Indexes, B+ Trees; Query Processing: Steps, Algorithms for Selection, Join Operation; Query Optimization: Transformation of Relational Expressions, Heuristics in Query Optimization, Selectivity and Cost Estimates in Query Optimization

Transaction Management: Transaction: ACID Properties, State diagram, Lock based Concurrency Control Protocols, Timestamp based Concurrency Control Protocol, Log based Recovery techniques, ARIES Recovery algorithm

Emerging Trends: NoSQL: RDBMS vs NoSQL, BASE properties, NoSQL Categories; NewSQL; Distributed Databases, Parallel Databases, Decision support systems, Data Warehouse, Data mining, Information Retrieval

List of Practicals: (For THL, TLP courses)

1. Choose a database system you propose to work on throughout the course. Perform requirements analysis in detail for design of the database. Design an entity-relationship (ER) data model for the selected database system.
2. Convert above ER model to relational model, semi structured data model. List functional dependencies. Normalize these relations up to 3NF/BCNF.
3. Consider a different database system. List functional dependencies [Include complex business logic.] Apply bottom - up approach using Relational Synthesis Algorithm for design of relational model for the chosen system. Verify decomposition properties.

4. Create tables with appropriate constraints for the relational schema. Create views, indices, and sequence. Alter the schema by adding/removing columns and constraints. Write DML queries.
5. Execute "SELECT" queries using order by, group by, aggregate functions, having clause, and set operators. Use SQL single row functions for date, time, string etc.
6. Write equijoin, non equijoin, self join and outer join queries. Write queries containing single row / multiple row / correlated subqueries using operators like =, in, any, all, exists etc. Write DML queries containing subqueries. Study a set of query processing strategies.
7. Write meaningful stored procedures in PL/SQL. Make use of cursors and different arguments. Write useful stored functions to perform complex computation. Write row level and statement level triggers in PL/SQL.
8. Implement a small database application for the above system using suitable front end and back end tool. Create a transaction by embedding SQL into an application program. Generate different useful reports.
9. Implementation of a small database using NoSQL and/or New SQL database system.

Text Books:

1. Abraham Silberschatz, Henry F. Korth, S. Sudarshan; "Database System Concepts"; 6th Edition, McGraw-Hill Education

2. Ramez Elmasri, Shamkant B. Navathe; "Fundamentals of Database Systems"; 6th Edition ;Pearson

Reference Books:

1. Thomas M. Connolly, Carolyn E. Begg," Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition ;Pearson

2. Raghu Ramakrishnan, Johannes Gehrke; "Database Management Systems", 3rd Edition ; McGraw Hill Education

Course Outcomes:

The student will be able to –

1. Design data models as per data requirements of an organization
2. Synthesize a relational data model upto a suitable normal form
3. Develop a database system using relational queries and PL/SQL objects
4. Apply indexing techniques and query optimization strategies
5. Understand importance of concurrency control and recovery techniques
6. Adapt to emerging trends considering societal requirements

CS2039: MICROPROCESSOR AND INTERFACINGS

Credits: 4 **Teaching Scheme: Theory: 3 Hours / Week**
Lab: 2 Hours/Week

Section 1: Topics/Contents

Introduction to 8086 Microprocessor: Introduction to 80x86 microprocessor, Internal Architecture, Generation of physical address, Minimum & Maximum Mode System, Ready and Reset pin significance ,study of 8086 supporting chips 8282(Latch), 8284(Clock Generator), 8286(Trans receiver), 8288(Bus Controller), Timing Diagram Read Write Machine Cycles, Real Mode.

Interrupt Structure and Programmable Interval: General Purpose Instructions, Address Translation, Addressing Modes, Introduction to Assembly Language Programming, Examples on Programming. Interrupt Structure , Interrupt service Routine, Interrupt Vector Table, Hardware and Software Interrupts, INTR,NMI , Interrupt Response, Execution of an ISR, Priority of Interrupts,8253/8254–(Programmable Interval timer/counter):Features, block diagram, control word & interfacing, Operating Modes of 8253.

Interfacing with 8086: 8255 (Programmable Peripheral Interface 8255)-block diagram, control word, Operating Modes of 8255, interfacing ADC (Successive Approximation Method), DAC (R – 2R ladder Network). 8251(USART): Features, Block Diagram, Control & status registers, Operating modes, Interfacing & Programming.

Section2: Topics/Contents

80386 Programming Model:80386– Features and Architecture , 80386 Programming Model: Register Organization, Segmentation- related instructions, descriptors, memory management through Segmentation, logical to linear/physical address translation.

80386 Paging& Protection: Virtual Memory, Paging – support registers, descriptors, linear to physical address translation, Control Registers. Protection in segmentation, DPL,RPL & CPL, Privileged instructions, page level protection, Translation Look aside Buffer.

Multitasking & Multi core Technologies: Concept of Multitasking ,Task State Segment, TSS Descriptor, Task Register, Task Gate Descriptor, Task Switching, Task Linking, Task Address Space. Limitation of Single Core, Introduction to parallel computers: Instruction Level Parallelism (ILP) vs. Thread Level Parallelism (TLP); Multi core Technologies.

List of Practical :

1. Study of 8086 Architecture and Execution of sample programs.
2. Write 8086 ALP to access marks of 5 subjects stored in array and find overall percentage and display grade according to it.
3. Write 8086 ALP to perform block transfer operation. (Don't use string operations) Data bytes in a block stored in one array transfer to another array. Use debugger to show execution of program.

4. Write 8086 ALP to find and count zeros, positive number and negative number from the array of signed number stored in memory and display magnitude of negative numbers.
5. Write 8086 ALP to convert 4-digit HEX number into equivalent 5-digit BCD number.
6. Write 8086 ALP to convert 5-digit BCD number into equivalent 4-digit HEX number.
7. Write 8086 ALP for following operations on the string entered by the user.
    1. String length
    2. Reverse of the String
    3. Palindrome
8. Write 8086 ALP for following operations on the string entered by the user (Use Extern Far Procedure).
    1. Concatenation of two strings
    2. Find number of words, lines.
    3. Find number of occurrence of substring in the given string.
9. Write 8086 ALP to convert an analog signal in the range of 0V to 5V to its corresponding digital signal using successive approximation ADC.
10. Write 8086 ALP to interface DAC & generate following waveforms on oscilloscope. Comment on types of DAC's and write detailed specifications of the DAC used
    1. Square wave -- Variable Duty Cycle & frequency.
    2. Stair case wave
    3. Triangular wave
    4. Sine and Cosine wave
11. Write 8086 ALP to program 8253 in Mode 0 to Mode 5. Generate a square wave with a pulse of 10 ms.

List of Project areas:

1. Minimum mode system using peripheral IC like 8254, 8255 etc.

2. Maximum mode system using peripheral IC's like 8254, 8255 etc.

Text Books

1. *Douglas Hall, "Microprocessors and Interfacing",2nd Edition, Tata McGraw Hill Publications, ISBN 0-07-025742-6.*
2. *"Advanced 80386, programming techniques " , James Turley , Tata McGraw Hill Publications, ISBN – 0-07-881342-5*
3. *Intel 80386 Programmer's Reference Manual 1986, Intel Corporation, Order no.: 231630- 011, December 1995.*

Reference Books

1. *Ray Duncan, "Advanced MS DOS Programming," 2nd Edition BPB Publications ISBN 0 – 07-048677-8*
2. *Chris H. Pappas; William H. Murray, "80386 Microprocessor Handbook," McGraw-Hill Osborne Media, ISBN 10: 0078812429.*
3. *Peter Abel, "Assembly Language Programming," 5th Edition, Pearson Education Publications, ISBN 10:013030655.*
4. *Intel 80386 Manual.*

Course Outcomes:

Student will be able to –

1. Identify different operating modes of Microprocessor
2. Design Microprocessor systems using peripheral components.
3. Learn peripherals and their interfacing with Microprocessor.
4. Analyze different operating modes of Advance Processor
5. Evaluate various memory management & Protection techniques.
6. Understand concept of multitasking and multi-core processors.

CS2003: AUTOMATA THEORY

Credits: 3 **Teaching Scheme: 3 Hours / Week**

Section 1: Topics/Contents

Finite Automata: Introduction to Automata, Computability and Complexity theory, Automaton as a model of computation, Central Concepts of Automata Theory: Alphabets, Strings, Languages. Decision Problems Vs Languages. Finite Automata, Structural Representations, Deterministic Finite Automata(DFA)-Formal Definition, Simplified notation: State transition graph, transition table, Language of DFA, construction of DFAs for Languages and proving correctness, Product construction, Nondeterministic finite Automata (NFA), NFA with epsilon transition, Language of NFA, Conversion of NFA with epsilon transitions to DFA, Automata with output. Applications and Limitation of Finite Automata

Regular and Non Regular Languages:

Regular expression (RE), Definition, Operators of regular expression and their precedence, Algebraic laws for Regular expressions, Kleene's Theorem: Equivalence Regular expressions and DFAs, Closure properties of Regular Languages(union, intersection, complementation, concatenation, Kleene clousure), Decision properties of Regular Languages, Applications of Regular expressions Myhill-Nerode theorem and its applications: proving non-regularity, lower bound on number of states of DFA, State Minimization algorithm, Equivalence testing of DFAs. Non Regular Languages, Pumping Lemma for regular Languages.**Context Free Grammars (CFG):** Context Free Grammars: Definition, Examples, Derivation, Languages of CFG, Constructing CFG, correctness proof using induction. Closure properties of CFLs (Union, Concatenation, Kleene clousure, reversal). Derivation trees, Ambiguity in CFGs, Removing ambiguity, Inherent ambiguity. Simplification of CFGs, Normal forms for CFGs: CNF and GNF. Decision Properties of CFLs (Emptiness, Finiteness and Membership). Applications of CFG.

Section2: Topics/Contents

Push Down Automata: Description and definition, Language of PDA, Acceptance by Final state, Acceptance by empty stack, Deterministic, Non-deterministic PDAs, CFG to PDA construction (with proof). Equivalence of PDA and CFG (without proof). Intersection of CFLs and Regular language.

Pumping lemma for CFLs, non-Context Free Languages, Context Sensitive Languages, Definition and Examples of Context Sensitive Grammars, Linear Bounded Automata. Chomsky hierarchy.

Turing Machines: Basic model, definition and representation, Instantaneous Description, Language acceptance by TM. Robustness of Turing Machine model and equivalence with various variants: Two-way/One-way infinite tape TM, multi-tape TM, non-deterministic TM, Universal

Turing Machines. TM as enumerator. Recursive and Recursively Enumerable languages and their *closure properties*.

Introduction to Undecidability

Church-Turing Thesis and intuitive notion of Algorithm. Introduction to countable and uncountable sets (countability of set of natural numbers, integers, rationals. Uncountability of set of real numbers, points in plane), Encoding for Turing machines and countability of set of all Turing machines. *Existence* of Turing unrecognizable languages via Cantor's diagonalization. Undecidability of Halting problem. Examples of undecidable problems: Post Correspondence Problem, Hilbert's 10^{th} Problem, Tiling problem (without proof). Example of Turing unrecognizable language. Decision properties of R, RE languages and Rice's theorem

---

Text Books:

1. Hopcroft J, Motwani R, Ullman, Addison-Wesley, "Introduction to Automata Theory, Languagesand Computation", Second Edition, ISBN 81-7808-347-7.

2. Michael Sipser, Course Technology, "Introduction to Theory of Computation", Third Edition,ISBN-10: 053494728X.

Reference Books:

1. J. Martin, "Introduction to Languages and the Theory of Computation", Third edition, Tata McGraw-Hill, ISBN 0-07-049939-x, 2003.

2. Java: The Complete Reference, Herbert Schildt, TMG Publication, ISBN 9780070636774 , 7th Edition

Course Outcomes:

The student will be able to –

1. Infer the applicability of various automata theoretic models for recognizing formal languages.
2. Discriminate the expressive powers of various automata theoretic and formal language theoretic computational models.
3. Illustrate significance of non determinism pertaining to expressive powers of various automata theoretic models.
4. Comprehend general purpose powers and computability issues related to state machines and grammars.
5. Explain the relevance of Church-Turing thesis, and the computational equivalence of Turing machine model with the general purpose computers.
6. Grasp the theoretical limit of computation (independent of software or hardware used) via the concept of undecidability.

MODULE V

Title: Course Structure **FF No. 653**

Branch: Computer **Year:** T.Y. **A.Y.:** 2019-20 **Module:** V **Pattern:** B-19.

Subject No.	Subject Code	Subject Name	Teaching Scheme (Hrs/Week)			Examination Scheme							Credits
			Theory	LAB	Tut.	ISA				ESE		Total	
						Assignment (%)	Lab Assessment (%)	MSE %)	GD/PPT (%)	Viva/Lab Exam (%)	ESE (%)		
S1	CS3024	Operating Systems	3	2	-	10	30	15	10	20	15	100	4
	CS3006	Augmented And Virtual Reality	3	2	-	10	30	15	10	20	15	100	
S2	CS3025	Computer Networks	3	2	-	10	30	15	10	20	15	100	4
	CS3229	Problem Solving Using OOP	3	2	-	10	30	15	10	20	15	100	
	CS3030:	Principles Of Programming Languages	3	2	-	10	30	15	10	20	15	100	4
S3	CS3002	Design and Analysis of Algorithms	3	2	-	10	30	15	10	20	15	100	4
	CS3035	Algorithm and Complexity	3	2	-	10	30	15	10	20	15	100	
S4	CS3001	Software Engineering	3	2	-	10	30	15	10	20	15	100	4
	CS3036	Software Design	3	2	--	10	30	15	10	20	15	100	

		Paradigms											
S5	CS3031	Engineering Design & Innovation 3	1	6				50			50		4
		TOTAL	**19**	**18**	**0**	**40**	**120**	**110**	**40**	**80**	**110**	**500**	**20**

FF No. :654

CS3024: OPERATING SYSTEMS

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Introduction to OS: What is OS, Interaction of OS and hardware, Goals of OS, Basic functions of OS, OS Services, System Calls, Types of system calls
Types of OS: Batch, Multiprogramming, Time sharing, Parallel, Distributed & Real-time OS.
Structures of OS: Monolithic, Layered, Virtualization-Virtual Machines, Microkernels
Introduction to Mobile OS: Architecture & Overview of Android OS.
Process Management: Shell: Linux commands, OS shell, Shell programming. **Processes:** Process Concept, Process States: 2, 5, 7 state models, Process Description, Process Control.
Threads: Multithreading models, Thread implementations – user level and kernel level threads, Symmetric Multiprocessing.
Concurrency: Issues with concurrency, Principles of Concurrency
Mutual Exclusion: H/W approaches, S/W approach, OS/Programming Language support: Semaphores, Mutex and Monitors. **Classical Problems of Synchronization**: Readers-Writers problem, Producer Consumer problem, Dining Philosopher problem
Process Scheduling: Uniprocessor Scheduling: Scheduling Criteria, Types of Scheduling: Preemptive, Non-preemptive, Long-term, Medium-term, Short-term. **Scheduling Algorithms**: FCFS, SJF, RR, Priority.
Multiprocessor Scheduling: Granularity, Design Issues, Process Scheduling. Thread Scheduling, Real Time Scheduling.

Section2: Topics/Contents

Deadlocks: Principles of deadlock, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Deadlock Recovery
Memory Management: Memory Management concepts: Memory Management requirements, Memory Partitioning: Fixed, Dynamic Partitioning, Buddy Systems, Fragmentation, Paging, Segmentation, Address translation.
Placement Strategies: First Fit, Best Fit, Next Fit and Worst Fit.
Virtual Memory: Concepts, Swapping, VM with Paging, Page Table Structure, Inverted Page Table, Translation Lookaside Buffer, Page Size, VM with Segmentation,VM with combined paging and segmentation.
Page Replacement Policies: FIFO, LRU, Optimal, Clock. **Swapping issues:** Thrashing
I/O and File Management: I/O management: I/O Devices - Types, Characteristics of devices, OS design issues for I/O management, I/O Buffering.
Disk Scheduling: FCFS, SCAN, C-SCAN, SSTF.
File Management: Concepts, File Organization, File Directories, File Sharing. Record Blocking, Secondary Storage Management, Free Space management, Security.

Case study: Windows 7: Design Principles, Process Management, Scheduling, Memory Management, I/O Management and File Management

List of Assignments

1. Basic & Advanced Linux Commands
2. Shell Script
3. Process Synchronization Problems
4. Simulation of CPU Scheduling Algorithms
5. Simulation of Memory Partitioning Algorithms
6. Simulation of Page Replacement Algorithms
7. Simulation of Disk Scheduling Algorithms

List of Project areas:

1. Linux based application using Shell Scripting and POSIX threads.
2. Design and implementation of a Multiprogramming Operating System: Stage I
   i. CPU/ Machine Simulation
   ii. Supervisor Call through interrupt
3. Design and implementation of a Multiprogramming Operating System: Stage II
   i. Paging
   ii. Error Handling
   iii. Interrupt Generation and Servicing
   iv. Process Data Structure
4. Design and implementation of a Multiprogramming Operating System: Stage III
   i. I/O Channels
   ii. Multiprogramming
   iii. I/O Spooling

Text Books:

1. Stalling William; "Operating Systems", 6th Edition, Pearson Education.

2. Silberschatz A., Galvin P., Gagne G.; "Operating System Concepts", 9th Edition, John Wiley and Sons.

Reference Books:

1. Silberschatz A., Galvin P., Gagne G ; "Operating System Principles"; 7th Edition, John Wiley and Sons.

2. Yashavant Kanetkar; "Unix Shell Programming", 2nd Edition, BPB Publications.

3. Forouzan B. A., Gilberg R. F.; "Unix And Shell Programming", 1st Edition, Australia Thomson Brooks Cole.

4. Achyut S. Godbole , Atul Kahate; "Operating Systems", 3rd Edition, McGraw Hill.

Course Outcomes:

The student will be able to –

1. Examine the functions of a contemporary Operating system with respect to convenience, efficiency and the ability to evolve.
2. Demonstrate knowledge in applying system software and tools available in modern operating system (such as threads, system calls, semaphores, etc.) for software development.
3. Apply various CPU scheduling algorithms to construct solutions to real world problems.
4. Identify the mechanisms to deal with Deadlock.
5. Understand the organization of memory and memory management hardware.
6. Analyze I/O and file management techniques for better utilization of secondary memory.

CS3006 :: AUGMENTED AND VIRTUAL REALITY

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Computer-Mediated Reality. Milgram's Reality-Virtuality continuum: Reality, Augmented Reality, Augmented Virtuality, Virtual Environment and Mixed Reality. Taxonomy of Mixed Reality: real, virtual, Extent of Work Knowledge (EWK), Reproduction Fidelity (RF), Extent of Presence Metaphor (EPM).

Introduction to AR, VR and MR: Differentiation, Features, use-cases and examples.

Geometry of Virtual World and Illumination: Birds-Eye View. Geometric Modelling. Matrix algebra and 2D rotations. 3D rotations and Yaw, Pitch and Roll. Axis angle representation. Quaternions. Converting and multiplying rotations. Homogeneous transforms. The chain of viewing transforms. Eye transforms. Viewport transforms. Three interpretations of light.

Refraction.Visual Perception: Depth perception, motion perception and frame rates.Visual Rendering: Overview, Shading models, rasterization, Pixel shading, Distortion shading. Tracking:Orientation tracking, tilt and yaw drift correction. Tracking with camera. Position tracking techniques. Interaction:selection, manipulation, isomorphic and non-isomorphic, exocentric and ego-centric interaction.Locomotion andDesign consideration.

Section 2: Topics/Contents

Basics of Image Processing: Color model: RGB andgrey. Basic linear and nonlinear operations.Image Enhancement Algorithms: contrast enhancement, histogram equalization. Segmentation.

Object Recognition and Tracking: edge and transition detection, circle, line and corner detection, smoothing, blurring, perspective recovery, feature point extraction: SIFT, Harris Corner Detection.

Vision based AR: Marker creation and marker tracking. Object tracking: Lucas Kanade Tracker, Optical Flow. Pose estimation. Rendering Techniques: Drawbacks of standard graphics libraries, artificial looks and aliasing artifacts due to rasterization. Lighting conditions, reflective properties estimation. Local illumination, secondary illumination, global illumination. 3D rendering, specialized rendering, wireframe rendering, and polygon-based rendering. Occlusion handling.

Location based Augmented Reality: GPS, gyroscope working and features. Sensor data fusion. Case study: Nokia City Lens/ Pokemon GO.

List of Practical:

1. Introduction to unity and 3D workspace and assets
2. Scripting in Unity, creating virtual environment and deploying app
3. Testing Cardboard unity sdk
4. Configuring cardboard sdk default app with Forest Environment with auto-locomotion
5. Introduction to Vuforia SDK
6. Creating a marker and rendering virtual object

Text Books:

1. "Augmented Reality and Virtual Reality The Power of AR and VR for Business" by tom Dieck, M. Claudia, Timothy Jung, Publisher: Springer; 1st ed. 2019 edition (February 19, 2019) ISBN-13: 978-3030062453
2. "Creating Augmented and Virtual Realities: Theory and Practice for Next-Generation Spatial Computing" by Erin Pangilinan, Steve Lukas, Vasanth Mohan, Publisher: O'Reilly Media; 1 edition (April 14, 2019), ISBN-13: 978-1492044192

Course Outcomes:

1. Identify the most suitable technique for a given use case based on the understanding of the similarities and differences between virtual, augmented and mixed reality with the help of Flynn's taxonomy
2. Design various transformations for manipulating an object in 3 dimensional vector space
3. Analyze rendering problems and rectify to provide realistic experience
4. Track visual cues for marker-based and marker-less augmented reality experience
5. Create local, global and secondary illumination for higher extent of presence metaphor
6. Extract sensory data to implement location based augmented reality experience

CS3025: COMPUTER NETWORKS

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Logical Link Control: Overview of OSI and TCP/IP models, Design Issues: Services to Network Layer, Framing, Error Control and Flow Control. Error Control: Parity Bits, Hamming Codes and CRC. Flow Control Protocols: Unrestricted Simplex, Stop and Wait, Sliding Window Protocol, CSMA/CD, WAN Connectivity: PPP and HDLC, PPPoE, M2M for IoT, Examples on Network Performance, Ethernet To Wireless and Vice Versa. **Network Layer:** Switching Techniques: Circuit, Message and Packet Switching. Logical Addressing: IPv4 and IPv6 is addressing, Sub-netting, NAT, CIDR. Network Layer Protocols: IP, ARP, ICMP, Overview of IPSec and VPN, Routing Protocols: Distance Vector, Link State, and Path Vector. Routing in Internet: RIP, OSPF and BGP. MANET: AODV and DSR Protocol. Congestion Control and QoS, Sensors and Sensor Node communication in context of IoT.

Section2: Topics/Contents

Transport Layer: Services: Berkley Sockets, Addressing, Connection Establishment, Connection Release, Flow control and Buffering, Multiplexing. HTH Layer Protocols: TCP, TCP Timer management, UDP. Quality of Service: TCP Congestion Control. Traffic Shaping: AIMD. Real Time Transport protocol (RTP), Stream Control Transmission Protocol (SCTP), RTCP, RSVP, Quality of Service (QoS), Differentiated services, TCP and UDP for Wireless. Light UDP for IoT. Overview of SSL and TLS **Application Layer:** Web Architecture, HTML Versions Context, Address Resolution: Domain Name System (DNS). WWW: Hyper Text Transfer Protocol (HTTP) and HTTPS with SSL. Web Service. Email: SMTP, MIME and S-MIME, POP3 and Webmail. File Transfer: FTP, FTPS, Dynamic Logical Addressing: Dynamic Host Control Protocol (DHCP). Simple Network Management Protocol (SNMP).

List of Practical's:

Operating System recommended: - 64-bit Open source Linux or its derivative
Programming tools recommended: - Open Source C, C++, JAVA, and PYTHON, Programming tool like G++ /GCC, Wireshark, Ethereal and Packet Tracer

1. Setting Up Small Network:
Setup a wired LAN using Layer 2 Switch and then IP switch of minimum four computers. It includes preparation of cable, testing of cable using line tester, configuration machine using IP addresses, test it using PING utility and capture the packets using Wireshark Packet Analyzer Tool. Extend the same Assignment for Wireless using Access Point

2. Understanding Data Link Layer Design Issues.
Write a program to configure RS232D ports on two nodes to demonstrate Framing, Flow control and Error control at data link layer.

3. Error Detecting Codes:
Write a program to simulate Cyclic Redundancy Check using polynomial method.

4. Error Correcting Codes:
Write a program to simulate error detection and correction for 7/8 bits ASCII and 4- Check bits using Hamming Codes.

5. Sliding Window Protocol:
Write a program to simulate Go back N and Selective Repeat Modes of Sliding Window Protocol in peer to peer mode using RS232D and capture the packets between client and server using Wireshark Packet Analyzer Tool for peer to peer mode.

6. TCP Socket Programming using Single Thread
Write a program using TCP sockets for following

a. Say Hello to Each other
b. File transfer
c. Calculator

Capture the packets between client and server using Wireshark Packet Analyzer Tool for peer to peer mode.

7. UDP Socket Programming using Single Thread
Write a program using UDP Sockets to enable file transfer (Script, Text, Audio and Video one file each) between two machines. Capture the packets between client and server using Wireshark Packet Analyzer Tool for peer to peer mode.

8. Understanding protocol stack of Intranet
Analyze packet formats of Ethernet, IP, TCP and UDP captured through Wireshark for wired network.

9. Distance Vector Routing Protocol
Write a program to find shortest path using Bellman Ford Equation for Distance Vector Routing Protocol which is used by Routing Information Protocol (RIP).

10. Link State Routing Protocol
Write a program to find shortest path using Dijkstra Equation for **Link State Routing Protocol** which is used by Open Shortest Path First Protocol (OSPF) in the Internet.

11. Border Gateway Protocol
Write a program to implement Border Gateway Protocol which is used by BGP Gateways in the Internet.

12. Preparation of TCP, IP Packets (Demo Assignment)

Write a program to prepare TCP and UDP packets using header files and send the packets to destination machine in peer to peer mode. Demonstrate the packets captured traces using Wireshark Packet Analyzer Tool for peer to peer mode.

Project Areas:

1. TCP based Multithreaded FTP client server
2. TCP based Multithreaded HTTP client server
3. UDP based Multithreaded TFTP client server
4. TCP based Multithreaded SMTP and POP3 mail client server
5. TCP based Multithreaded SMTP and IMAP mail client server
6. TCP based Multithreaded Chat client server
7. UDP based Multithreaded Chat client server
8. UDP based Multithreaded Audio Conferencing client server
9. UDP based Multithreaded Video Conferencing client server
10. UDP and Multimedia Protocol based Multithreaded Audio Conferencing client server
11. UDP and Multimedia Protocol based Multithreaded Video Conferencing client server
12. Implementation of RIP/OSPF/BGP using Packet Tracer
13. Simulation of AODV routing protocol using NS2/NS3/OMNet

Text Books:

3. *Andrew S. Tenenbaum, "Computer Networks",5th Edition, PHI, ISBN 81-203-2175-8.*
4. *Kurose, Ross "Computer Networking a Top Down Approach Featuring the Internet", Pearson; 6th edition (March 5, 2012), ISBN-10: 0132856204*

Reference Books: (As per IEEE format)

5. *Fourauzan B., "Data Communications and Networking", 5th edition, Tata McGraw- Hill, Publications, 2006*
6. *Matthew S. Gast "802.11 Wireless Networks", O'Reilly publications; 2nd Edition.*
7. *C. Siva Ram Murthy and B. S. Manoj, "Ad Hoc Wireless Networks: Architectures and Protocols" Prentice Hall, 2004*
8. *Holger Karl and Andreas Willig, "Protocols and Architectures for Wireless Sensor Networks", Wiley, ISBN: 0-470-09510-5*

Course Outcomes:

The student will be able to –

1. Select network protocols for WAN connectivity.
2. Estimate reliability issues based on error control, flow control and pipelining by using bandwidth, latency, throughput and efficiency.

3. Analyse data flow between peer to peer in an IP network
4. Demonstrate sustainable engineering practice indicating the scientific purpose and utility of communication frameworks and standards.
5. Integrate real time multimedia services in service framework
6. Develop client servers using client-server architectures and prototypes by the means of correct standards, protocols and technologies

Attainment Levels: 1, 5, 3, 4, 2, 4

CS3029: **Problem Solving Using OOP**

Credits: 3 **Teaching Scheme: 3hrs /Week**
Lab: 2 Hours/Week

Section 1: Topics/Contents

Introduction to Problem Solving: Design of algorithms, Iterative versus recursive style, problem solving using a functional style, correctness issues in programming, efficiency issues in programming, time and space measures, Imperative style of programming.
Object Oriented Paradigm: Limitations of procedural thinking. Need for Object Oriented Paradigm. Reasons for OOP's popularity. Overview of Object Oriented Thinking - agents and communities, classes and instances, information hiding, communicating with agents: message and methods, message vs procedure calls, responsibilities, objects as service providers, class hierarchies, inheritance, abstract classes, method binding and overriding, polymorphism. Case Studies. Object Oriented Concepts- Hierarchy, Encapsulation, Modularity, Abstraction. Principles for object-oriented problem solving. Designing good programs - problem specification, problem decomposition, class design, data design, method design, algorithm design, coding, testing, debugging, and revising. Case Studies.
Abstraction and Encapsulation: Problems of Matrix and Sparse matrix, Polynomial using abstractions and encapsulation such as Objects, classes, object references, creating an object, heap storage, scope, fields and methods. Passing values and references. static keyword (for fields and methods). this keyword. this and static. Initialization & Cleanup: constructors, method overloading, finalization and garbage collection, member initialization. Hiding the implementation - package, access specifiers,.

String Objects: Problems like set operation using string basics, finding things within a string, retrieving parts of strings, processing each character in a string, comparing strings, parsing text, string processing applications.
Reusing classes - composition and inheritance. Method overriding. Composition vs inheritance. protected access specifier. Upcasting. final classes and methods. Intialization and class loading in case of inheritance.

Polymorphism – Problems such as Tree traversal and Threaded Binary tree using polymorphism concepts. static and dynamic. Concept of dynamic polymorphism. Method-call binding. Producing the right behavior. Abstract classes and methods. Constructors and Polymorphism. Inheritance based design. Interfaces.

Section2: Topics/Contents

Error handling with exceptions - basic exceptions, catching an exception, creating your own exceptions, finally keyword, exception hierarchy

Different types of inheritance in C++, polymorphism in C++, operator overloading in C++, virtual keyword in C++, Exception handling in C++, **Similarities and differences between C++ and Java:** destructors, access modifiers, inheritance, polymorphism, garbage collection, exception handling.

Input and Output – Objects for I/O. File, Streams, Readers and Writers. File Management / Processing, RandomAccessFile, New IO, Object serialization and deserialization.

Object Oriented Design of Graphical User Interfaces: Problems such as Graphs algorithm for MST like Prims and Kruskals visualization using Applets. Applet vs Application. AWT to Swing. Getting started with Swing. Model, View, Controller (MVC) architecture. Using Swing components - labels, buttons, check boxes, radio buttons, combo boxes, dialogs, file choosers, text components, list etc. Controlling layouts. Event handling, event listeners and call backs. Concept of inner classes. Handling mouse and keyboard events.

Concurrency: Problems such Deadlock using concurrency. Motivation, threads, thread life cycle, thread Priority, Thread Methods. Issues with concurrency - improperly accessing resources, race conditions, colliding over resources, resolving shared resource contention, critical sections. Cooperation between threads (inter thread communication). Solving Producer-Consumer using object oriented principles.

Object Oriented Design Principles: SOLID – Single Responsibility Principle, Open-Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, Dependency Inversion Principle

Multithreading in C++, Concept of Deadlocks.

List of Practicals

1. Write a C++ program to implement the concept of objects, classes, constructors, destructors.
2. Assignments based on Matrix and Sparse Matrix using OOPs concept.
3. Assignments based on Polynomial Evaluation, Addition, Multiplication using OOPs concept.
4. Write a C++ program to create a student class with details like Roll no, Name, Marks of five subjects, percentage, class (First, Second, etc) have following functions: parameterized constructor, destructor, Display, Calculate percentage and grade using inline function.
5. Assignments based on Set Operations using OOPs.
6. Friend function and function overloading.
7. String operations using operator overloading.
8. Assignment based on Threaded Binary tree using OOPs.
9. Assignments based on Graph Algorithm using OOPs.
10. File operations.
11. Templates
12. Write a program to implement class for Book details as follows
    Private: bookid, book_name, author, price

Public: get_details(), print_details()
Get details of some books from user and print in tabular form. Also find the total prize of all the books.

13. Write a program to implement Complex class as follows
    Private:real_value, imag_value
    Public:Parameterized constructor(1 and 2 parameter), copy constructor, getter and setter functions.(Use of constructors and destructors)
14. Write a program to extend complex class with following details
    Public: add_complex(), subtract_complex(), multiply_complex(), print_complex(), getcount_complex() functions (Use of member functions and static variable)
15. Write a program to extend complex class with following details
    Public: Operator functions for +, - , multiply and print complex numbers. (Operator overloading using member and non-member functions)
16. Write a program to implement following inheritance hierarchy
    Student(name) -> Comp_Student(int Roll_Number), IT_Student(int Roll_Number)
    Comp_Student -> From_DSE_Student(float Diploma_Marks),
    From_FY_Student(float CPI) (5 classes in total)
    Keep all member variables private.
    Provide public member functions to set and get data.
17. Write a program to extend Book class with dynamic memory allocation.
    (Use of "new" keyword and object pointers)
18. Write a program to implement following class hierarchy using inheritance, object pointers and virtual functions.
    Animal(eat())->Dog(eat())->Labrador(eat())
    Every class is having eat() method. Apply concept of run-time polymorphism.
19. Write a program to extend Book class to get book details from user and store data to a file. Write functions to store data to file and get data from file.
20. Design a car racing game using object oriented concepts. Construct class Car with member functions like run_car(), movecar_left(), movecar_right() etc.
21. Extend the car racing game to provide support for User profile creations and maintain the levels of each user on secondary storage. Your program must be able to load previous state of user like levels completed, total score etc.

Text Books:

3. "An Introduction to Programming through C++", Abhiram G Ranade, McGrawHill Education
4. "The C++ Programming Language", Bjarne Stroustrup, 4th Edition, Addison-Wesley Pearson Education

Reference Books:

4. "Thinking In Java – The Definitive Introduction to Object-Oriented Programming in the Language of the World-Wide Web", Bruce Eckel, Fourth Edition, Pearson Education, Inc.
5. "Java, java, Java – Object-Oriented Problem Solving", R. Morelli and R. Walde, 3rd edition, Pearson Education, Inc.

6. "Java – The Complete Reference", Herbert Schildt, 9th Edition, Oracle Press

Course Outcomes:

The student will be able to –

1. Apply object oriented thinking towards solving a real world problem

2. Evaluate the advantages of object-oriented approach over procedural approach of solving a problem

3. Implement a given object oriented design into code using an object-oriented programming language

4. Effectively use the object-oriented features of an object-oriented programming language towards developing a software solution

5. Assess the quality of a given design based on the well established principles of object oriented design.

CS3030: **Principles of Programming Languages**

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Introduction to Programming Languages :What is a programming language? Why to study principles of programming languages – design, implementation and application of programming languages. Language design principles: characteristics of a good programming language. Data and control abstractions in a programming language. Categorization of various programming languages (purpose, level, domain, paradigm). Programming paradigms – Imperative/Procedural, Object Oriented, Functional Programming, Logic Programming

OOP concepts: Classes and Objects, Java object storage, Different ways to create objects in Java, How to swap or exchange objects, Inheritance in Java, Encapsulation in Java, Abstraction in Java, Run-time Polymorphism in Java, Association, Composition and Aggregation, Access and Non Access Modifiers in Java, Access Modifiers, this reference, Method Overloading, Method Overriding, Understanding "static" in "public static void main" in Java, Overloading or Overriding static methods, Shadowing of static methods(Also called Method Hiding), Static methods vs Instance methods in Java, Assigning values to static final variables in Java, Covariant return types, Object class, Static class in Java, Flexible nature of java.lang.Object, Overriding equals method of Object class, Overriding toString() method of Object class, Instance Variable Hiding, Static blocks in Java, initializer block in java, instance initializer block in java(non-static block), Static vs Dynamic Binding,Why Java is not a purely Object-Oriented Language?, Understanding Classes and Objects in Java, All operators in Java.

Inheritance: Inheritance in Java, Multiple Inheritance, Why Java does'nt support Multiple, Inheritance – The Diamond Problem, Java Object Creation of Inherited Class, Inheritance and constructors, Interfaces and Inheritance, Using final with Inheritance, Accessing Grandparent's member, Override private methods, More restrictive access to a derived class method in Java, Parent and Child classes having same data member, Object Serialization with Inheritance, Referencing Subclass objects with Subclass vs Superclass reference, Does overloading work with inheritance.

Input and Output: Character Stream Vs Byte Stream, DoubleStream mapToObj() in Java, Command Line arguments, Scanner Class, Scanner and nextChar(), Scanner vs BufferReader Class, Formatted output, Fast I/O for Competitive Programming, Reading input from console.

Strings in Java: String Class(Practice), StringBuffer Class, StringBuilder Class, StringTokenizer class, StringJoiner in Java8, Initialize and Compare Strings, String vs StringBuilder vs StringBuffer, When to use StringJoiner over StringBuilder?, Integer to String Conversions, String to Integer– parseInt().

Arrays in Java: Arrays in Java(Practice), Default Array values, Util Arrays Class (Contains utility functions for Arrays), Reflect Array class in Java, util.Arrays vs reflect.Array in Java, Final arrays, Interesting facts about Array assignment in Java, Jagged Array, Array IndexOutofbounds Exception, Array vs ArrayList in Java.

Stream in Java: Java Stream, Java Stream.Builder, Java IntStream, Java IntStream.Builder, Java LongStream, Java LongStream.Builder, Java DoubleStream, Java DoubleStream.Builder, Java Stream Collectors.**Important Keywords in JAVA.**

C++ vs Java: C++ vs Java, Static keyword in C++ vs Java, Exception Handling in C++ vs Java, Inheritance in C++ vs Java, Virtual behavior differ in C++ vs Java, Foreach in C++ and Java, C/C++ Pointers vs Java References, Comparison of boolean data type in C++ and Java, Floating Point Operations & Associativity in C, C++ and Java.

Methods in Java: Methods, Parameters passing, Returning Multiple values, Throwable fillInStackTrace() method in Java, Valid variants of main(), Variable Arguments (Varargs) method, Method Overloading, Different ways of Method Overloading in Java, Method overloading and null error, Method Overloading with Autoboxing and Widening, Method Overloading and Ambiguity in Varargs, Overloading main(), Overriding equals method, Overriding toString() method, Private and final methods, Java is Strictly Pass by Value, Clone() method, Remote Method Invocation, Default Methods, Passing and Returning Objects in Java, Date after() method in Java, System.exit() method.

Section2: Topics/Contents

Constructors: Constructors in Java, Default constructor, Assigning values to static final variables, Copy Constructor, Constructor Chaining, Private Constructors and Singleton Classes.

Exception Handling: Exceptions, OutOfMemoryError Exception, 3 Different ways to print Exception messages in Java, flow control in try-catch-finally, Types of Exceptions, Catching base and derived classes as exceptions, Checked vs Unchecked Exceptions, Throw and Throws, User-defined Custom Exception, Infinity or Exception?, Multicatch, Chained Exceptions, Null Pointer Exception.

Interfaces and Abstract Classes: Interfaces, Access specifier for methods in interfaces, Access specifiers for classes or interfaces, Abstract Classes, Difference between Abstract Class and Interface in Java, Comparator Interface, Java Interface methods, Nested Interface, Nested Classes in Java, Inner class in java, Local Inner Class in Java, Anonymous Inner Class in Java, Functional Interfaces, What is a Marker interface.

Java Packages: Packages Introduction, java.io package, java.lang package, java.util package

Collection in Java: AbstractCollection, Collections Class in Java, Enumeration, Iterators and ListIterators, Convert an Iterable to Collection in Java, Using Iterators, Iterator vs Foreach, Types of iterator, Creating Sequential Stream from an Iterator in Java.

Multithreading: Multithreading, Lifecycle and states of a thread, Main thread, Methods to prevent thread execution, inter thread communication, Java.lang.Thread class, Start() function in multithreading , Java Thread Priority, Joining Threads in Java, Naming a thread and fetching name of current thread in Java, Synchronization, Method and Block Synchronization, Producer-Consumer solution, Thread Pools in Java, Semaphore in Java, Java.util.concurrent.Semaphore class in Java, CountDownLatch, Deadlock in java, Daemon thread, Reentrant Lock, Cyclic Barrier in Java, Callable and Future in Java, Runtime Class.

Garbage Collection: Garbage Collection, How to make object eligible for garbage collection in Java?, Mark-and-Sweep, Island of Isolation, Automatic Resource Management, Iterator vs Collection in Java.

Wrapper Classes: Wrapper Classes in Java, Primitive Wrapper Classes are Immutable in Java, Number Class, Integer class, Byte class, Short class, Long class, Float class, Double class, Boolean Class, Character Class, Autoboxing and Unboxing in Java.

File Handling: File class, Ways of Reading a text file in Java, file permissions in java, Moving a file from one directory to another using Java, Copying file using FileStreams, Delete a file using Java, Java program to delete duplicate lines in text file, Java program to merge two files alternatively into third file, Java program to List all files in a directory and nested sub-directories.

List of Practicals

23. Assignments based on basic OOPs concept in Java.
24. Assignments based on Inheritance.
25. Assignments based on Java Input and output.
26. Assignments based on Strings in Java.
27. Assignments based on Arrays in Java.
28. Assignments based on Java Stream.
29. Assignments based on Methods in Java.
30. Assignments based on Constructors and Exception handling in Java.
31. Assignments based on Interfaces and Abstract Classes.
32. Assignments based on Collections and Multithreading in Java.
33. Assignments based on Wrapper Classes in Java.
34. Assignments based on File Handling in Java.

Text Books:

5. " JAVA- The Complete Reference", Herbert Schildt, 11th Edition, McGraw Hill Education.

6. "Programming Languages Design and Implementation", T. W. Pratt, M.V. Zelkowitz, Publications, ISBN 10: 0130276782, 4th Edition

Reference Books:

7. "Thinking In Java – The Definitive Introduction to Object-Oriented Programming in the Language of the World-Wide Web", Bruce Eckel, Fourth Edition, Pearson Education, Inc.
8. "Java, java, Java – Object-Oriented Problem Solving", R. Morelli and R. Walde, 3rd edition, Pearson Education, Inc.

Course Outcomes:

The student will be able to –

1. Apply object oriented thinking towards solving a real world problem

2. Evaluate the advantages of object-oriented approach over procedural approach of solving a problem

3. Implement a given object oriented design into code using an object-oriented programming language

4. Effectively use the object-oriented features of an object-oriented programming language towards developing a software solution

5. Assess the quality of a given design based on the well established principles of object oriented design.

CS 3001: SOFTWARE ENGINEERING

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Software Engineering Paradigms: Overview of Software Engineering, Software Process Framework, Process Patterns, Personal and Team Process Models, Process Models: Code-and-Fix, Waterfall Model, Rapid Application Development, Incremental Models, Evolutionary Models, Iterative Development, The Unified Process, Cleanroom Methodology, Component-Based Software Engineering, CMMI, Impact of Processes and Outcomes, Process Selection and applicability, Software Engineering Principles and Practices Component-Assembly Process Model, Best Practices in Technology Selection, Formal Methods

Requirement Engineering: Requirements Engineering Tasks, Requirement Elicitation Techniques, Software Requirements: Functional, Non-Functional, Domain Engineering activities, Requirements Characteristics and Characterization, Eliminating Requirement Ambiguities, Conflict Identification and Resolution, Requirement Qualities, Requirement Specification, Requirement Traceability, Requirement Prioritization, Relationship of Requirement Engineering to other Framework Activities, System Scope Determination and Feasibility Study, Statement of Work Generation, Requirements Verification and Validation, Requirement Maturity, Technical Reviews, Stakeholder Management

Introduction to Agile Methodology: Introduction to Agile Project Management, Agile History and the Agile Manifesto, Agile Requirements: Team, Program, Portfolio Level, Scrum Overview, Scrum Framework, Agile Principles, Sprints Design, Time-Boxing, Kanban, and Theory of Constraints, Requirements and User Stories, Stakeholders, User Personas, and User Experiences, Product Backlog, Agile Planning, Estimation and Velocity, Product Backlog, Technical Debt

Section2:Topics/Contents

Agile: Scrum and Sprints: Stakeholders, System Acceptability, Levels of Abstractions, System Interfaces, Organizational Roles and Missions, Problem, Opportunity and Solution Spaces, System Environment and Mission Analysis, System Operational Model, Agile Roles: Product Owner, Scrum Master, Development Team, Development Team, Managers, Agile Planning: Scrum Planning Principles, Multilevel Planning, Portfolio Planning, Envisioning (Product Planning), Release Planning (Longer-Term Planning) Sprinting: Sprint Planning, Sprint Execution, Sprint Review, Sprint Retrospective, Agile Architecture and Re-architecting with

Flow, Agile Methods: Lean Software Development, DSDM, Extreme Programming, TDD, Ux Methodology, Lean Development

Software Design and Configuration Management: Analysis Concepts, Analysis Methods, Analysis Modeling Techniques, Data Flow diagrams, System Analysis Scenarios and Model Generation, Context Models, Behavioral Models, Data Models, Object Models, Structured Methods Design Concepts, The Design Model, Design Qualities, Characteristics of Design activities, Design Principles, Cohesion and Coupling, Software Architecture Vs Software Design, Software Reuse, Design Heuristics, User Interface Design: Rules, User Interface Analysis and Steps in Interface Design, Design Evaluation, Source Code Management, Build Engineering, Environment Configuration, Change Control, Release Management, Deployment, Hardware Configuration, Industry Standards and Frameworks, CM Values, CM Practices, CM Practices for Agile

Project Management Principles: Project Management Activities, Structures and Frameworks, Teamwork, Leadership, Project Planning, Project Scheduling, Risk Analysis, Introduction to Function Points, Empirical Estimation, COCOMO II model, Foundations of Software Testing: Terms, Testing Cycle, Outcomes, Principles,Unit and Acceptance testing, Software Verification and Validation, Classic Mistakes, Complex Systems, Critical Systems, Software Safety

---

List of Practicals: (For THL, TLP courses)

1. A real-world problem issue is required to be identified with manageable scope. The problem scenarios are required to be identified for target system to be developed. The scenarios are stated in the form of Statement-of-Work template. The SOW document shall address the vision, goals, and objectives of the project.
2. The initial requirements and feature set for the target system is required to be identified. The requirements are required to be synthesized with stakeholder participation. The project roles are assigned to the project team with clear indicator of responsibilities. The initial requirements summary document with adequate and minimal infrastructure is required to be developed using multiple iterations.
3. A concise requirement specification document is required to be prepared using Agile Requirements Practices with the help of user stories narration, user personas and collaborative communication. The Agile tools like Face-to-face communications, Daily standups, and Customer Idea Management shall be practiced.
4. The product backlog for the project aimed at maintaining a prioritized queue of project requirements shall be created.
   - It should be dynamic and should be continuously groomed as the project progresses. Agile projects generally use an iceberg strategy for grooming the product backlog.
   - The items that are near the top of the iceberg and are closest to going into development should get the most attention.
   - There should typically be about two to three sprints worth of stories at the top of the backlog that are well-groomed and ready to go into development in order to avoid a situation where the project team is waiting for work to do.
5. The feasibility of the project shall be prepared and stated in the form of Project Feasibility Study document mentioning finalized requirement set and dropped feature list along with requirement prioritization and traceability matrix.

6. The project plan of the project shall be prepared using Agile Planning Practices indicating level of uncertainty, technology considerations, and related risk nomenclature.
7. Sprint-level planning activity accommodating story points, planning poker shall be performed. The Sprint-plan and Sprint-design indicating detailed activity planner shall be developed.
8. The Software Configuration Management Plan (SCMP) shall be prepared to establish and maintain the integrity of the products of the software project throughout the project's software life cycle.The SCM practices identifying specific configuration items/units are contained in the key process areas that describe the development and maintenance of each configuration item/unit.
   - Software configuration management activities are planned.
   - Selected software work products are identified, controlled, and available.
   - Changes to identified software work products are controlled.
   - Affected groups and individuals are informed of the status and content of software baselines.
9. Working software shall be developed by performing Sprint Execution. The software artifacts created shall be verified and validated using unit/module testing.
10. A Sprint Review document shall be prepared using the Summarize, Demonstrate, Discuss, and Adapt approaches indicating Sprint Review Issues and Sign-offs.

Text Books:

1. *Ian Sommerville, 'Software Engineering', Addison-Wesley, 9th Edition, 2010, ISBN-13: 978-0137035151.*
2. *Roger S Pressman,'Software Engineering: A Practitioner's Approach', McGraw Hill, 6/e,2005*

Reference Books :

1. *SorenLauesen, Software requirements: Styles and techniques, Addison Wesley, ISBN 0201745704, 2002*
2. *Kenneth S. Rubin, Essential SCRUM: A Practical Guide To The Most Popular Agile Process, Addison-Wesley, ISBN-13: 978-0-13-704329-3, 2012*
3. *Dean Leffingwell, Agile Software Requirements, Addison-Wesley, ISBN-13: 978-0-321-63584-6, 2011*
4. *Charles G. Cobb, The Project Manager's Guide To Mastering Agile: Principles and Practices for an Adaptive Approach, Wiley Publications, ISBN: 978-1-118-99104-6 (paperback), ISBN 978-1-118-99177-0 (epdf), 2015*
5. *Bob Aiello and Leslie Sachs, Configuration Management Best Practices, Addison Wesley, ISBN-13: 978-0-321-68586-5, 2010*
6. *Mario E. Moreira, Adapting Configuration Management for Agile Teams, Wiley Publications, ISBN: 9780470746639, 2010*

Course Outcomes:

Upon completion of the course, graduates will be able to –

1. Summarize capabilities and impact of Software Development Process Models and justify process maturity through application of Software Engineering principles and practices

focusing tailored processes that best fit the technical and market demands of a modern software project.

2. Discriminate competing and feasible system requirements indicating correct real world problem scope and prepare stepwise system conceptual model using stakeholder analysis and requirement validation.
3. Formulate system specifications by analyzing User-level tasksand compose software artifacts using agile principles, practices and Scrum framework
4. Propose and demonstrate realistic solutions supported by well-formed documentation with application of agile roles, sprint management, and agile architecture focusing project backlogs and velocity monitoring.
5. Conform to Configuration Management principles and demonstrate cohesive teamwork skills avoiding classic mistakes and emphasizing on software safety adhering to relevant standards.
6. Analyze the target system properties and recommend solution alternatives by practicing project planning, scheduling, estimation and risk management activities.

CS 3036: SOFTWARE DESIGN PARADIGMS

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
: Lab 2 Hours / Week

Section 1: Topics/Contents

Business Process Management: Introduction to Business modeling, Introduction to Business Processes, Business Process Modeling Foundation, Process Orchestrations, Process Choreographies, Properties of Business Processes, Business Process Management Architectures, Business Process Methodology, Introduction to BPEL, Advantages of business modeling, Business process modeling methods, Discovery, analysis, design, validation, and implementation Modeling types, Definition of business area-BPM metadata, Importance of the BPMN standards ,Business process modeling standards, Flow objects, Connecting objects, Swimlanes, Artifacts, Steps in BPMs

System Behavior Specification: Static Behavior: Use Cases, Use Case Diagram Components, Use Case Diagram, Actor Generalization, Include and Extend, Template for Use Case Narrative, Building Domain Model, and capturing system behavior in use cases

Dynamic Behavior: Sequence diagrams, object lifelines and message types, Modeling collections multiobjects, Refining sequence diagrams, Collaboration diagrams, States, events and actions, Nested machines and concurrency, Modifying the object model to facilitate states, Modeling methods with activity diagrams, Activity Diagrams: Decisions and Merges, Synchronization, Iteration, Partitions, Parameters and Pins, Expansion Regions, Swimlanes, concurrency and synchronization, Communication Diagram, Interaction Overview Diagrams, Timing Diagrams

Software Architecture Primitives: Foundations of Software Architecture, Reference Architectures, Architectural Design: Software Architecture, Data Design and Architectural Design, Views, Viewpoints, Perspectives, Conceptual Architecture View, Module Architecture View, Execution Architecture View, Code Architecture View, Architecture styles: Repository, Layered, Pipe-Filter, Call-Return, Peer-Peer, Publish-Subscribe, Client-Server, Two-Tier, Three-Tier, N-Tier, Heterogeneity in Architecture

Section2:Topics/Contents

System Design Specification: Design of Software Objects, Features and Methods, Cohesion and Coupling between Objects, Coupling and Visibility, Interfaces, Interfaces with Ball and Socket Notation, Templates, Analysis model vs. design model classes, Categorizing classes: entity, boundary and control , Modeling associations and collections, Preserving referential integrity, Achieving reusability, Reuse through delegation, Identifying and using service packages, Improving reuse with design Packages and interfaces: Distinguishing between classes/interfaces, Exposing class and package interfaces, Subscribing to interfaces Component and deployment diagrams: Describing dependencies, Deploying components across threads, processes and processors

Design Patterns: Introduction to Design Pattern, Describing Design Patterns, Catalogue of Design Patterns Creational Patterns: Abstract Factory, Builder, Factory Method, Prototype,

Singleton, Structural Patterns: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy, Behavioral Patterns: Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor, Antipatterns, Applications of Design Patterns

Model Driven Development: Overview of Model Driven Development and Model Driven Engineering, Model Transformation, Introduction to Model Driven Architecture: MDA Terms and Concepts, Model Mappings, Marking Models, Executable Models, MOF, CWM, Introduction to XML, XMI, Introduction to UML Metamodel and UML 2.X diagrams, Extensibility Mechanisms and its usage, Introduction to OCL, Model Based Software Engineering, Domain-Specific Modeling: Fundamentals and Architecture, MDA Applications

---

List of Practicals: (For THL, TLP courses)

1. To narrate Requirement Definition Document for the target system with following three areas:Problem Identification, Problem Definition, and Problem Statement
2. To narrate System Requirements Specification Document for target system with reference to the IEEE 610.12.1990 Std guidelines.
3. To create Business Process Diagrams for all the scenarios identified using BPMN 2.0 and BPM practices. Process modeling captures the ordered sequence of activities within a process along with supporting information from end to end. In process modeling, the business process is framed in a BPD to reflect the activities, the roles that conduct those activities, conditional branching, and the sequence of the workflow between the activities.
4. To decompose and organize the problem domain area into broad subject areas and identify the boundaries of problem/system. Specify the behavior of the target system and map requirements to Use cases.
    a. The System Context Diagram depicts the overall System behavioral trace and Requirement Capture diagram depicts the hierarchical Use case Organization. The Use Case diagram should encompass
    b. Actors (External Users)
    c. Transactions (Use Cases)
    d. Event responses related to transactions with external agents.
    e. Detection of System boundaries indicating scope of system.
5. To depict the dynamic behavior of the target system using sequence diagram. The Sequence diagram should be based on the Scenarios generated by the inter-object Communication. The model should depict:
    a. Discrete, distinguishable entities (class).
    b. Events (Individual stimulus from one object to another).
    c. Conditional events and relationship representation.
6. To depict the state transition with the life history of objects of a given class model. The model should depict:
    a. Possible ways the object can respond to events from other objects.
    b. Determine of start, end, and transition states.
7. To depict the dynamic behavior using detailed Activity diagram. Activity is a parameterized behavior represented as coordinated flow of actions. The flow of execution is modeled as activity nodes connected by activity edges.
    - A node can be the execution of a subordinate behavior, such as an arithmetic computation, a call to an operation, or manipulation of object contents.

- Activities may form invocation hierarchies invoking other activities, ultimately resolving to individual actions.

8. To develop logical static structure of target system with Software Class diagram. To prepare Class Collaboration-Responsibility (CRC) cards for the Conceptual classes traced from System analysis phase. The design model should depict
    a. Relationship between classes: inheritance, Assertion, Aggregation, Instantiation
    b. Identification of objects and their purpose.
    c. Roles / responsibilities entities that determine system behavior.
9. To enhance Software Class diagram to Architecture diagram with appropriate design patterns. The patterns selected shall be justifiable and applied to individual and distinct hierarchies. Suitable Architectural Styles shall be selected and the structural elements shall be well-documented.
10. To represent physical module that provides occurrence of classes or other logical elements identified during analysis and design of system using Component diagram. The model should depict allocation of classes to modules. To narrate precise Program Design Language constructs separating computation from interface. To represent deployment view of the system through Architecture Diagram.

Text Books:

1. *Tom Pender, "UML Bible", John Wiley & sons, ISBN – 0764526049*
2. *Jim Arlow, IlaNeustadt, "UML 2 and Unified Process: Practical Object Oriented Analysis and Design.", 2nd Edition, Addison- Wesley, ISBN – 0321321278.*

Reference Books:

1. *Mellor, Scott, Uhl, Weise, "MDA Distilled", Pearson Education, ISBN 81-297-0529X*
2. *Grady Booch, James Rambaugh, Ivar Jacobson, "Unified Modeling Language Users Guide", 2nd Edition, Addison- Wesley, ISBN – 0321267974*
3. *ErichGamma,RichardHelm,RalphJohnson,"DesignPatterns:Elements of ReusableObject-Oriented Software"(Addison-WesleyProfessionalComputing Series),JohnVlissides,Publisher:Addison-Wesley Professional, ISBN-10: 0201633612 ISBN-13: 978-0201633610*
4. *Steven Kelly, Juha-PekkaTolvanen, Domain-Specific Modeling: Enabling Full Code Generation, John Wiley & Sons, Inc., ISBN 978-0-470-03666-2, 2008*
5. *Paul Clements, Felix Bachmann, Len Bass, David Garlan, Documenting Software Architectures: Views and Beyond Addison-Wesley Professional 2003, ISBN-10:0201703726, ISBN-13: 9780201703726*
6. *Charles S. Wasson, System Analysis, Design, and Development: Concepts, Principles, and Practices, John Wiley & Sons, Inc.,ISBN-13 978-0-471-39333-7, 2006*
7. *Essential Business Process Modeling, Michael Havey, First Edition August 2005 Oreilly, ISBN 10: 0-596-00843-0 | ISBN 13: 9780596008437*

Course Outcomes:

Upon completion of the course, graduates will be able to –

1. Examine and breakdown real-world problem scenarios into structured partitions depicting static and dynamic behavior of the system using business process management practices, object-oriented analysis principles and Model Driven Development practices.
2. Identify and formulate software requirements and behavioral models using static and dynamic behavioral views indicating structured problem partitioning and state-based exploration.
3. Compose system analysis and design specifications indicating logical, physical, deployment, and concurrency viewpoints using object-oriented analysis and design principles and Model Driven Engineering practices.
4. Construct and justify the evolutionary system description models expressing high-level architecture accommodating applicable architectural styles compatible to requirements and behavioral models using UML-supported modeling tools.
5. Comprehend the nature of design patterns by understanding a small number of examples from different pattern categories and apply these patterns in creating a correct design using design heuristics, published guidance, applicability, reasonableness, and relation to other design criteria resulting inwell-documented system profiles to the engineering and social community.
6. Propose multi-faceted defendable solutions demonstrating team-skills accommodating design patterns reducing the potential cost and performance impedance in order to realize system artifacts with the help of Model Driven Development practices.

CS3002: DESIGN AND ANALYSIS OF ALGORITHM

Credits: 03 **Teaching Scheme: Theory: 3 Hours / Week**
Lab: 2 Hours/Week

Section 1: Topics/Contents

Basic introduction, time complexity analysis, Divide and Conquer

Asymptotic notations (Big Oh, small oh, Big Omega, Theta notations). Best case, average case, and worst case time and space complexity of algorithms. Overview of searching, sorting algorithms. Adversary lower bounds (for comparison based sorting, for finding second minima). Using Recurrence relations and Mathematical Induction to get asymptotic bounds on time complexity. Master's theorem and applications. Proving correctness of algorithms.

Divide and Conquer: Analyzing Quick sort, Randomized Quick sort, Merge sort, Counting Inversions, Finding majority element, Finding Median, Efficient algorithms for Integer arithmetic (Euclid's algorithm, Karatsuba's algorithm for integer multiplication, fast exponentiation), Finding closest pair of points in plane, computing convex hull of points in plane, basic idea of FFT algorithm and applications.

Dynamic Programming:

General strategy, simple dynamic programming based algorithms to compute Fibonacci numbers, binomial coefficients, Matrix Chain multiplication, Optimal binary search tree (OBST) construction, Coin change problem, 0-1 Knapsack, Traveling Salesperson Problem, All pair shortest path algorithm, Longest increasing subsequence problem, Longest common subsequence problem, Largest independent set for trees.

Greedy:

Analysis and correctness proof of minimum spanning tree and shortest path algorithms, Huffman coding, conflict free scheduling, fractional knapsack.

Section 2: Topics/Contents

Backtracking Strategy, Linear Programming:

Backtracking: General strategy, n-queen problem, graph coloring, subset sum problem.

Linear Programming: Introduction to linear programming, geometric interpretation, LP duality, Simplex algorithm, Linear optimization problems and their LP formulation.

Flows and Matchings:

Flows: Flows in the network, Max-flow min-cut theorem, Ford Fulkerson's algorithm, LP formulation of flow problem, Applications (e.g. image segmentation, airline scheduling)

Matchings: Perfect matchings in bipartite graphs, LP formulation, Hall's marriage theorem, Konig's theorem, augmenting path algorithm for matchings.

Introduction to NP-completeness, Approximation Algorithms:

Complexity classes P, NP, coNP, and their interrelation, Notion of polynomial time many one reductions reduction. Notion of NP-hardness and NP-completeness. Cook's Theorem and implication to P versus NP question. NP-hardness of halting problem. NP-Complete problems (some selected examples from - Satisfiability problem, Circuit-SAT, 3-CNF SAT, vertex cover

problem, independent set problem, clique problem, Hamiltonian-circuit problem, subset sum problem.)

Introduction to Approximation algorithms, NP-optimization problems, Approximation algorithm for Vertex Cover, Traveling Sales Person Problem(TSP), Set-cover.

Text Books: ***(As per IEEE format)***

1. *Cormen, Leiserson, Rivest and Stein "Introduction to Algorithm" ,PHI 3nd edition, 2009. ISBN 81-203-2141-3*

2. Jon Kleinberg, Eva Tardos "Algorithm Design", Pearson, 1st edition, 2005. ISBN 978-81-317-0310-6

Reference Books: ***(As per IEEE format)***

1. *Bressard, Bratley "Fundamentals of Algorithmics." ,PHI, 2nd Edition, 1996, ISBN 81-203-1131-0*

2. Horowitz, Sahani, "Fundamentals of computer Algorithms", Galgotia. 2nd Edition, 1998.ISBN 81-7515-257-5

List of Practicals: (For THL, TLP courses)

1. Randomized quick sort
2. Median Finding
3. FFT algorithm
4. Closest pair of points
5. All pair shortest path
6. Longest Increasing subsequence
7. Traveling Salesperson Problem
8. Viterbi Algorithm
9. Huffman Coding
10. Ford Fulkerson Algorithm
11. Augmenting Path algorithm for maximum matching
12. Application of Network Flows in image segmentation
13. Solving NP-complete Problems using ILP solver
14. Approximation algorithm for metric TSP

Course Outcomes:

The student will be able to –

1. Formulate computational problems in abstract and mathematically precise manner
2. Design efficient algorithms for computational problems using appropriate algorithmic paradigm
3. Analyze asymptotic complexity of the algorithm for a complex computational problem using suitable mathematical techniques.
4. Formulate computational problem as linear program and apply LP, network flow, based techniques to design efficient algorithms for them.
5. Establish NPcompleteness of some decision problems, grasp the significance of the notion of NPcompleteness and its relation with intractability of the decision problems and design efficient approximation algorithms for standard NP-optimization problems.
6. Incorporate appropriate data structures, algorithmic paradigms to craft innovative scientific solution for a complex computing problems.

CS3035: ALGORITHMS AND COMPLEXITY

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Basic introduction, time complexity analysis, Divide and Conquer
Asymptotic notations (Big Oh, small oh, Big Omega, Theta notations). Best case, average case, and worst case time and space complexity of algorithms. Overview of searching, sorting algorithms. Adversary lower bounds (for comparison based sorting, for finding second minima). Using Recurrence relations and Mathematical Induction to get asymptotic bounds on time complexity. Master's theorem and applications. Proving correctness of algorithms.
Divide and Conquer: Analyzing Quick sort, Randomized Quick sort, Merge sort, Counting Inversions, Finding majority element, Finding Median, Efficient algorithms for Integer arithmetic (Euclid's algorithm, Karatsuba's algorithm for integer multiplication, fast exponentiation), Finding closest pair of points in plane, computing convex hull of points in plane, basic idea of FFT algorithm and applications.
Dynamic Programming:
General strategy, simple dynamic programming based algorithms to compute Fibonacci numbers, binomial coefficients, Matrix Chain multiplication, Optimal binary search tree (OBST) construction, Coin change problem, 0-1 Knapsack, Traveling Salesperson Problem, All pair shortest path algorithm, Longest increasing subsequence problem, Longest common subsequence problem, Largest independent set for trees.
Greedy:
Analysis and correctness proof of minimum spanning tree and shortest path algorithms, Huffman coding, conflict free scheduling, fractional knapsack.

Section 2: Topics/Contents
Backtracking Strategy:
Backtracking: General strategy, n-queen problem, graph coloring, subset sum problem.

Flows and Matchings:
Flows: Flows in the network, Max-flow min-cut theorem, Ford Fulkerson's algorithm, Linear Programming formulation of flow problem, Applications (e.g. image segmentation, airline scheduling)
Matchings: Perfect matchings in bipartite graphs, LP formulation, Hall's marriage theorem, Konig's theorem, augmenting path algorithm for matchings.

Introduction to NP-completeness, Approximation Algorithms:
Complexity classes P, NP, coNP, and their interrelation, Notion of polynomial time many one reductions reduction. Notion of NP-hardness and NP-completeness. Cook's Theorem and implication to P versus NP question. NP-hardness of halting problem. NP-Complete problems (some selected examples from - Satisfiability problem, Circuit-SAT, 3-CNF SAT, vertex cover

problem, independent set problem, clique problem, Hamiltonian-circuit problem, subset sum problem, Integer Linear Programming.) Reducing standard NP complete problems to ILP.

Introduction to Approximation algorithms, NP-optimization problems, Approximation algorithm for Vertex Cover, Traveling Sales Person Problem(TSP), Set-cover.

Text Books: ***(As per IEEE format)***

1. *Cormen, Leiserson, Rivest and Stein "Introduction to Algorithm" ,PHI 3nd edition, 2009. ISBN 81-203-2141-3*

2. Jon Kleinberg, Eva Tardos "Algorithm Design", Pearson, 1st edition, 2005. ISBN 978-81-317-0310-6

Reference Books: ***(As per IEEE format)***

1. *Bressard, Bratley "Fundamentals of Algorithmics." ,PHI, 2nd Edition, 1996, ISBN 81-203-1131-0*

2. Horowitz, Sahani, "Fundamentals of computer Algorithms", Galgotia. 2nd Edition, 1998.ISBN 81-7515-257-5

List of Practicals: (For THL, TLP courses)

1. Randomized quick sort
2. Median Finding
3. FFT algorithm
4. Closest pair of points
5. All pair shortest path
6. Longest Increasing subsequence
7. Traveling Salesperson Problem
8. Viterbi Algorithm
9. Huffman Coding
10. Ford Fulkerson Algorithm
11. Augmenting Path algorithm for maximum matching
12. Application of Network Flows in image segmentation
13. Solving NP-complete Problems using ILP solver
14. Approximation algorithm for metric TSP

Course Outcomes:

The student will be able to –

1. Formulate computational problems in abstract and mathematically precise manner
2. Design efficient algorithms for computational problems using appropriate algorithmic paradigm
3. Analyze asymptotic complexity of the algorithm for a complex computational problem using suitable mathematical techniques.
4. Formulate computational problem as linear program and apply LP, network flow, based techniques to design efficient algorithms for them.
5. Establish NPcompleteness of some decision problems, grasp the significance of the notion of NPcompleteness and its relation with intractability of the decision problems and design efficient approximation algorithms for standard NP-optimization problems.
6. Incorporate appropriate data structures, algorithmic paradigms to craft innovative scientific solution for a complex computing problems.

CS3010: DISTRIBUTED ALGORITHMS

Credits:4 **Teaching Scheme: Theory: 3 Hours / Week**
Lab: 2 Hours/Week

Section 1: Topics/Contents

Introduction: Introduction to distributed algorithms, First example: coloring paths. Coloring using unique identifier, Coloring with randomized algorithms, lower bound for 2-coloring of paths, Challenges of distributed algorithms.

Review of Prerequisite Topics: Graph theory covering graphs, subgraphs, walks, Connectivity and Distances, Isomorphism, Packing and Covering, Labeling and Partitions, Directed Graphs and Orientations.
Probability theory covering Markov's inequality, Chebyshev's inequality, Chernoff bounds, Markov chains and random walks.

Models of computing:
A distributed program, A model of distributed executions,Models of communication networks Port Numbering Model, Distributed algorithms in PN model: Coloring paths, bipartite graph maximum matching, vertex cover. Local Models, unique identifiers. Directed pseudo forests, merging colorings. CONGEST models and bandwidth limitations. Message passing and shared memory models, synchronous and asynchronous timing models, failure models. Complexity measures like time, space, and message complexity. Global state of a distributed system, Cuts of a distributed computation, Past and future cones of an event, logical time, scalar and vector time.

Section 2: Topics/Contents

Global state and snapshot recording algorithms: Introduction, System model and definitions, Snapshot algorithms for FIFO channels,Variations of the Chandy–Lamport algorithm, Snapshot algorithms for non-FIFO channels, Snapshots in a causal delivery system, Monitoring global state, Necessary and sufficient conditions for consistent global
snapshots.

Fundamental Problems on Distributed Networks:
Maximal independent set, minimum spanning tree, vertex coloring, dominating set, routing algorithms, leader election, Byzantine agreement, synchronizers, graph spanners, dynamic networks.
Storage and retrieval of data in peer-to-peer computing, coverage and routing in sensor networks, and rumor spreading in social networking.

Proving Impossibility results: Covering maps, Local neighborhoods, Ramsey theory and applications.

Text Books and Reference Books:

1. Distributed Algorithms, by Nancy Lynch.

2. Distributed Computing: Fundamentals, Simulations, and Advanced Topics, by Hagit Attiya and Jennifer Welch.

3. Distributed Computing: Principles,Systems and Algorithms. By Ajay D. Kshemkalyani and Mukesh Singhal

4. Randomized Algorithms, by Rajeev Motwani and Prabhakar Raghavan.

5. Principles of Distributed Computing, lecture notes by Roger Wattenhofer.

6. Preprint of book by Jukka Suomela https://users.ics.aalto.fi/suomela/da/da-print.pdf

Course Outcomes:

The student will be able to –

1. Formulate computational problems in abstract and mathematically precise manner
2. Design efficient algorithms for computational problems using appropriate algorithmic paradigm
3. Analyze asymptotic complexity of the algorithm for a complex computational problem using suitable mathematical techniques.
4. Formulate computational problem as linear program and apply LP, network flow, based techniques to design efficient algorithms for them.
5. Establish distributed systems with optimization of complexity.
6. Incorporate appropriate data structures, algorithmic paradigms to craft innovative scientific solution for a complex computing problems.

MODULE VI

Issue 1, Rev. 1 , Dt. 01/07/2018

Title: Course Structure **FF No. 653**

Branch: Computer **Year:** T.Y. **A.Y.:** 2019-20 **Module:** VI **Pattern:** B-19.

Subject No.	Subject Code	Subject Name	Teaching Scheme (Hrs/Week)			Examination Scheme							Credits
			Theory	LAB	Tut.	ISA				ESE		Total	
						Assignment (%)	Lab Assessment (%)	MSE %)	GD/PPT (%)	Viva/Lab Exam (%)	ESE (%)		
S1	CS3013	Artificial Intelligence	3	2	-	10	30	15	10	20	15	100	4
	CS3037	Intelligent Systems	3	2	-	10	30	15	10	20	15	100	
S2	CS3038	Client Server Computing	3	2	-	10	30	15	10	20	15	100	4
	CS3014	Web Technologies	3	2	-	10	30	15	10	20	15	100	
S3	CS3006	Augmented & Virtual Reality	3	2	-	10	30	15	10	20	15	100	4
	CS3018	Network Security	3	2	--	10	30	15	10	20	15	100	
S4	CS3012	Compiler Design	3	2	-	10	30	15	10	20	15	100	4
	CS3016	System Programming	3	2	-	10	30	15	10	20	15	100	
S5	CS3031	Engineering Design & Innovation 3	1	6			50			50			4

S6	CS3333	**General Proficiency 3											
		TOTAL	**25**	**22**	**0**	**40**	**120**	**110**	**40**	**80**	**110**	**500**	**20**

1.

CS 3013: ARTIFICIAL INTELLIGENCE

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Fundamentals of Artificial Intelligence: Introduction, A.I. Representation, Non-AI & AI Techniques, Representation of Knowledge, Knowledge Base Systems, State Space Search, Production Systems, Problem Characteristics, types of production systems, Intelligent Agents and Environments, concept ofrationality, the nature of environments, structure of agents, problem solving agents, problem formulation.

Searching: Depth First Search, Breadth First Search, Generate & test, Hill Climbing, Best First Search, A* and AO* Algorithm, Constraint satisfaction, Means-Ends Analysis. Game playing: Minimax Search, Alpha-Beta Cutoffs, Waiting for Quiescence. **Planning:** Blocks world, STRIPS, Implementation using goal stack, Partial Order Planning, Hierarchical planning, and least commitment strategy. Conditional Planning, Continuous Planning Machine Learning Algorithms.

Section2: Topics/Contents

Knowledge Representation: Knowledge based agents, Wumpus world, Propositional Logic: Representation, Inference, Reasoning Patterns, Resolution, First order Logic: Representation, Inference, Reasoning Patterns, Resolution, Forward and Backward Chaining. Basics of PROLOG: Representation, Structure, Backtracking, Expert System.

Uncertainty: Non Monotonic Reasoning, Logics for Non Monotonic Reasoning, Forward rules and Backward rules, Justification based Truth Maintenance Systems, Semantic Nets StatisticalReasoning, Probability and Bayes' theorem, Bayesian Network, Markov Networks, HiddenMarkov Model,Basis of Utility Theory, Utility Functions.

List of Lab assignments:

1. Analysis of AI and Non-AI technique by implementing any two player game
2. Analysis of Heuristic search techniques by Implementing any two Heuristic search techniques such as Hill Climbing, Best First Search, A*, AO* etc.
3. Analysis of Constraint satisfaction Problems
4. Implementation of Expert system in PROLOG.
5. Implementation of any real time problem using PROLOG.

Text Books:

1.Elaine Rich and Kevin Knight: "Artificial Intelligence." Tata McGraw Hill

2. Stuart Russell & Peter Norvig : "Artificial Intelligence : A Modern Approach", Pearson Education, 2nd Edition.

Reference Books:

1. Ivan Bratko : "Prolog Programming For Artificial Intelligence" , 2nd Edition Addison Wesley, 1990.
2. Eugene, Charniak, Drew Mcdermott: "Introduction to Artificial Intelligence.", Addison Wesley
3. Patterson: "Introduction to AI and Expert Systems", PHI
4. Nilsson : "Principles of Artificial Intelligence", Morgan Kaufmann.
5. Carl Townsend, "Introduction to turbo Prolog", Paperback, 1987

Course Outcomes:

Upon completion of the course, graduates will be able to –

1. Identify problems that are amenable to solution by AI methods, and which AI methods may be suited to solving a given problem.
2. Formalize a given problem in the language/framework of different AI methods (e.g., as a search problem, as a constraint satisfaction problem, as a planning problem, as a Markov decision process, etc).
3. Implement basic AI algorithms (e.g., standard search algorithms or dynamic programming).
4. Design and carry out an empirical evaluation of different algorithms on a problem formalization, and state the conclusions that the evaluation supports.
5. Use various symbolic knowledge representations to specify domains and reasoning tasks of a situated software agent.

CS3037: INTELLIGENT SYSTEMS

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Intelligence System Basics Data, Information, Knowledge and Intelligence, Model of an intelligent system, Intelligent agents, Environment, Agent theory, Agent types, Inter-agent communication

Problem-solving: Problem representation, Problem solving techniques, Informed (Heuristic) strategy, Heuristic Functions ,Search Best-first, A* search methods, AO*,Hill climbing algorithm, Important concepts of Game theory, Constraint Satisfaction Problem

Knowledge and Reasoning: Models of knowledge representations, Knowledge based system, Knowledge representation and reasoning, Reasoning types, Inference, propositional logic, Representation of knowledge using rules, knowledge acquisition, semantic representations: semantic networks, frames, frame/script systems

Section II: Topics/Contents

Uncertain Knowledge and Decision Theory : Uncertainty and methods, Basic Probability Notation, Bayes rule and its use, Bayesian techniques, Relational and First Order Probability Models, The Basis of Utility Theory,

artificial neural networks: Human brain, Neural Network, Notation, Back-propagation learning,Applications of neural network, Supervised learning, Unsupervised learning, Reinforcement Learning

Intelligent Applications: smart system, autonomous cars, flying cars, object recognition, robotic vision. application domains: configuration, diagnosis, intelligent user interfaces, input and output modalities and perception towards multimodal interaction, conversational agents, question answering, emotions and affective / accessible computing, user modeling, practical implications of choosing and applying AI solutions, intelligent Mobile apps and social networks, knowledge representation and the web, recent trends & applications in intelligent systems

List of Lab assignments:

1. Rule-based knowledge representation
2. Use of intelligent agents in data processing
3. Implementation of Expert system
4. Data acquisition and preprocessing for intelligent system
5. Implementation of robotic arm / blocks for object manipulation
6. Design of intelligent user interfaces for mobile app

Text Books:

1. *Stuart Russell and Peter Norvig (1995), Artificial Intelligence: A Modern Approach," Third edition, Pearson, 2003.*
2. *Robert J. Schalkoff: Intelligent Systems: Principles, paradigms, and pragmatics, Jones & Barlett Learning, ISBN-978-93-80853-16-1.*
3. *Elaine Rich and Kevin Knight: "Artificial Intelligence." Tata McGraw Hill*
4. *Learn Prolog Now! by Patrick Blackburn, Johan Bos and Kristina Striegnitz.*
5. *Dan W. Patterson: "Introduction to AI and Expert Systems",Pearson*

Reference Books:

1. *Ivan Bratko : "Prolog Programming For Artificial Intelligence" , 2nd Edition Addison Wesley, 1990.*
2. *Nilsson : "Principles of Artificial Intelligence", Morgan Kaufmann.*
3. *Carl Townsend, "Introduction to turbo Prolog", Paperback, 1987*

Course Outcomes:

Upon completion of the course, graduates will be able to –

1. Identify problems that are amenable to solution by AI methods, and which AI methods may be suited to solving a given problem.
2. Formalize a given problem in the language/framework of different AI methods (e.g., as a search problem, as a constraint satisfaction problem, as a planning problem, as a Markov decision process, etc).
3. Implement basic AI algorithms (e.g., standard search algorithms or dynamic programming).
4. Design and carry out an empirical evaluation of different algorithms on a problem formalization, and state the conclusions that the evaluation supports.
5. Use various symbolic knowledge representations to specify domains and reasoning tasks of a situated software agent.

CS3038: CLIENT SERVER COMPUTING

Credits: 4 **Teaching Scheme:Theory:3 Hours / Week**
Lab:2 Hours / Week

Section 1: Topics/Contents

Web Development Process, Front End Tools: Introduction to web technology, internet and www, Web site planning and design issues, HTML5: structure of html document, HTML elements: headings, paragraphs, line break, colors & fonts, links, frames, lists, tables, images and forms, , CSS, Bootstrap , XML. **Client Side Technologies:** HTML5 forms Validation, JavaScript: Overview of JavaScript, Data types, Control Structures, Arrays, Functions and Scopes, Objects in JS, DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM, JQuery: Introduction to JQuery, Loading JQuery, Selecting elements, changing styles, creating elements, appending elements, removing elements, handling events. Introduction to JSON, AJAX

Section 2: Topics/Contents

Server Side Technologies -I Server Side technology and Apache TOMCAT, Servlet: Introduction to servlet, need and advantages, Servlet Lifecycle, Creating and testing of sample servlet, session management. **Server Side Technologies-II :** JSP: Introduction to JSP, advantages of JSP over Servlet , elements of JSP page: directives, comments, scripting elements, actions and templates, JDBC, Servelet connectivity with Database: MySQL/MOngoDB SOAP, REST. **Client Side Framework:** React JS : Overview, MVC architecture, directives, expression, controllers, filters, tables, modules, forms, includes, views, scopes, services, dependency injection, custom directives, Internationalization

List of Project areas:
Operating System recommended: - 64-bit Open source Linux or its derivative

Programming tools recommended: - Android Studio 3.x, JAVA

1. Design and deploy web based application using front end technologies HTML5, CSS, Bootstrap and XML. Perform validation using Java script/JQuery/HTML5.
(For Example: Course Registration System, Voter System for Election, e-Shopping System, e-Governance System, On-line Trading System etc)

2. Develop dynamic web application essence as an extension to project 1 using either (JSP/Servlet, Tomcat, MySQL/MongoDB) or (PHP, Apache, MySQL/MongoDB) server side technologies.

Text Books: ***(As per IEEE format)***

1.Achyut Godbole & Atul Kahate," Web Technologies: TCP/IP to Internet Application Architectures" , McGraw Hill Education publications, Third Edition, 2016

2.Ralph Moseley & M. T. Savaliya, "Developing Web Applications", Wiley publications, Second Edition, 2014

Reference Books: ***(As per IEEE format)***

1. *Adam Bretz & Colin J Ihrig, "Full Stack Javascript Development with MEAN", SPD, First Edition 2015, Indian Reprint September 2015*
2. *GiulioZambon, " Beginning JSP, JSF and Tomcat", Apress Publication, Second Edition, 2013*
3. *Jeremy McPeak& Paul Wilton," Beginning JavaScript", Wrox Publication, Fifth Edition, 2015*
4. *Black Book, " JDBC 4.2, Servlet 3.1 & JSP 2.3", Dreamtech Press, 2016*
5. *Sandeep Panda, "Angular JS: Novice To Ninja", SPD, First Edition 2014, Indian Reprint 2015*
6. *J2EE Architecture, an illustrative gateway to enterprise solutions: concept to Application Design and deployment by B. V. Kumar, S. Sangeetha, S. V. Subrahmanya, Tata McGraw Hill Publishing Company.*

Course Outcomes:

The student will be able to –

1. Prepare the front end view of web pages using HTML5, CSS with Bootstrap framework
2. Perform client side web page forms validation.
3. Refine dynamic web pages with JSP, Servlet
4. Design realistic and extensible light weight web application using PHP.
5. Practice and utilize web framework paradigms and principles for Web development.
6. Develop reliable, efficient, scalable web services

CS3014: WEB TECHNOLOGY

Credits: 4 **Teaching Scheme: Theory:3 Hours / Week**
Lab:2 Hours / Week

Section 1: Topics/Content

Web Development Process, Front End Tools: Introduction to web technology, internet and www, Web site planning and design issues, HTML5: structure of html document, HTML elements: headings, paragraphs, line break, colors & fonts, links, frames, lists, tables, images and forms, , CSS, Bootstrap , XML.

Client Side Technologies: HTML5 forms Validation, JavaScript: Overview of JavaScript, Data types, Control Structures, Arrays, Functions and Scopes, Objects in JS, DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM, JQuery: Introduction to JQuery, Loading JQuery, Selecting elements, changing styles, creating elements, appending elements, removing elements, handling events. Introduction to JSON

Server Side Technologies –I: Server Side technology and TOMCAT, Servlet: Introduction to servlet, need and advantages ,Servlet Lifecycle, Creating and testing of sample servlet, session management. JSP: Introduction to JSP, advantages of JSP over Servlet , elements of JSP page: directives, comments, scripting elements, actions and templates, JDBC, MongoDB

Section 2: Topics/Content

Server Side Technologies-II: PHP: Introduction to PHP, Features, sample code, PHP script working, PHP syntax, conditions & Loops, Functions, String manipulation, Arrays & Functions, Form handling, Cookies & Sessions, File Handling, Exception Handling, E-mail, MySQL with PHP, AJAX

Web Services: Web Services: Overview, types of application web services, SOAP, REST, EJB, JNDI lookup, Content Management System(CMS)

Server Side Technologies –II: PHP: Introduction to PHP, Features, sample code, PHP script working, PHP syntax, conditions & Loops, Functions, String manipulation, Arrays & Functions, Form handling, Cookies & Sessions, File Handling, Exception Handling, E-mail, MySQL with PHP, AJAX

Web Technology Frameworks: Angular JS : Overview, MVC architecture, directives, expression, controllers, filters, tables, modules, forms, includes, views, scopes, services, dependency injection, custom directives, Internationalization, NodeJS.

List of Practical's: (For THP)

Operating System recommended: - 64-bit Open source Linux or its derivative
Programming tools recommended: - Eclipse, JavaBeans, WampServer

List of Project areas: (THP)

1. Design and deploy web based application using front end technologies HTML5, CSS, Bootstrap and XML. Perform validation using Java script/JQuery/HTML5.
(For Example: Course Registration System, Voter System for Election, e-Shopping System, e-Governance System, On-line Trading System etc)

2. Develop dynamic web application essence as an extension to project 1 using either (JSP/Servlet, Tomcat, MySQL/ MongoDB) or (PHP, Apache, MySQL/MongoDB) server side technologies.

Text Books:

1. Achyut Godbole & Atul Kahate," Web Technologies: TCP/IP to Internet Application Architectures" , McGraw Hill Education publications, Third Edition, 2016
2. Ralph Moseley & M. T. Savaliya, "Developing Web Applications", Wiley publications, Second Edition, 2014

Reference Books:

7. *Adam Bretz & Colin J Ihrig, "Full Stack Javascript Development with MEAN", SPD, First Edition 2015, Indian Reprint September 2015*
8. *GiulioZambon, " Beginning JSP, JSF and Tomcat", Apress Publication, Second Edition, 2013*
9. *Jeremy McPeak& Paul Wilton," Beginning JavaScript", Wrox Publication, Fifth Edition, 2015*
10. *Black Book, " JDBC 4.2, Servlet 3.1 & JSP 2.3", Dreamtech Press, 2016*
11. *Sandeep Panda, "Angular JS: Novice To Ninja", SPD, First Edition 2014, Indian Reprint 2015*
12. *J2EE Architecture, an illustrative gateway to enterprise solutions: concept to Application Design and deployment by B. V. Kumar, S. Sangeetha, S. V. Subrahmanya, Tata McGraw Hill Publishing Company.*

Course Outcomes:

The student will be able to –

7. Design the front end view of web pages using HTML5, CSS with Bootstrap framework
8. Perform client side web page forms validation.
9. Refine dynamic web pages with JSP, Servlet
10. Deliver realistic and extensible light weight web application using PHP.
11. Practice and utilize web framework paradigms and principles for Web development.
12. Develop reliable, efficient, scalable web services

CS 3012: COMPILER DESIGN

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Lexical Analysis and introduction to Syntax Analysis: Introduction to Compiler, Phases and Passes, Bootstrapping, Cross Compiler, Role of a Lexical Analyzer, Specification and Recognition of Tokens, LEX, Expressing Syntax, Top-Down Parsing, Predictive Parsers. Implementing Scanners, operator precedence parsers.

Syntax and Semantic Analysis: Bottom-Up Parsing, LR Parsers: constructing SLR parsing tables, constructing Canonical LR parsing tables, Constructing LALR parsing tables, using ambiguous grammars, YACC, Type Checking, Type Conversion. Symbol-Table Structure.

Syntax-Directed Translation and Intermediate Code Generation: Syntax-Directed Definitions, Bottom-Up Evaluation, Top-Down Translation, Intermediate Representations, Intermediate Code Generation. Error Detection & Recovery: Lexical Phase errors, syntactic phase errors semantic errors. More about translation: Array references in arithmetic expressions, case statements.

Section2:Topics/Contents

Code Generation: Issues in Code Generation, Basic Blocks and Flow Graphs, Next-use information, A simple Code generator, DAG representation of Basic Blocks, Peephole Optimization. Generating code from dags.

Code Optimization and Run Time Environments: Introduction, Principal Sources of Optimization, Optimization of basic Blocks, Introduction to Global Data Flow Analysis, Runtime Environments, Source Language issues. Storage Organization, Storage Allocation strategies, Access to non-local names, Parameter Passing

List of Practicals:

1. Assignment to understand basic syntax of LEX specifications, built-in functions and Variables.
2. Implement a Lexical Analyzer using LEX for a subset ofC.
3. Implement a parser for an expression grammar using LEX and YACC.
4. Generate and populate appropriate Symbol Table.

5. Implement the front end of a compiler that generates the three address code for a simple language.

6. Generate an appropriate Target Code from the given intermediate code assuming suitable processor details.

7. A Register Allocation algorithm that translates the given code into one with a fixed number of registers.(Optional)

8. Implement local and Global Code Optimizations such as Common Sub-expression Elimination, Copy Propagation, Dead-Code Elimination, Loop and Basic-Block Optimizations. (Optional)

9. Design and Implement simple Compiler Using All Above Phases(Optional)

Text Books:

1. *"Compilers: Principles, Techniques and Tools", A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman, Addison Wesley, ISBN 978-81317-2101-8, Second Edition, 2007.*
2. *"Engineering a Compiler", K. Cooper, L. Torczon, Morgan Kaufmann, ISBN 1-55860-698-X, First Edition, 2003.*

Reference Books:

1. *"Advanced Compiler Design and Implementation", S. S. Muchnik, Morgan Kaufmann, ISBN 8178672413, First Edition, 1997.*
2. *"Lex & Yacc", J. R. Levine, T. Mason, D. Brown, "Lex & Yacc", J. R. Levine, T. Mason, D. Brown, O'Reilly,* ***ISBN*** *1-56592-000-7, Second Edition, 1992.*

Additional Reading

1. *"Compiler Construction: Principles and Practice", K. Louden, Course Technology,* ***ISBN 0-534-93972-4,*** *First Edition, 1997.*

Course Outcomes:

Upon completion of the course, graduates will be able to -

1. Design basic components of compiler including scanner, parser and code generator.
2. Perform semantic analysis in a syntax directed fashion using attributed definitions.
3. Apply local and global code optimization techniques.
4. Synthesize machine code for runtime environment.
5. Develop software solutions for the problems related to compiler construction.
6. Adapt themselves to the emerging trends in language processing.

CS3016: SYSTEMS PROGRAMMING

Credits: 04 **Teaching Scheme: Theory: 3 Hours / Week**
Lab: 2 Hours/Week

Section 1: Topics/Contents

Introduction to System Software: Introduction, software types, software hierarchy, components of system software, machine structure, interfaces, address space, levels of system software, recent trends in software development. Language processors: Programming languages and language processors, fundamentals of language processing, life cycle of a source program, language processing activities, data structures for language processing: search data structures, allocation data structures.

Macroprocessor: Introduction, macro definition and call, macro expansion, nested macro calls, design of macro processor, design issues of macro processors, two-pass macro processors, one-pass macro processors.

Assembler: Elements of assembly language programming, design of the assembler, assembler design criteria, types of assemblers, two-pass assemblers, one-pass assemblers, assembler algorithms, multi-pass assemblers, variants of assemblers design of two pass assembler, machine dependent and machine independent assembler features.

Compilers: Introduction to compiler phases, introduction to cross compiler, features of machine dependent and independent compilers, overview of types of compilers. Interpreters: compiler vs. interpreter, phases and working.

Linkers: Relocation and linking concepts, static and dynamic linker, subroutine linkages.

Loaders: Introduction to loader, loader schemes: compile and go, general loader scheme, absolute loaders, relocating loaders, direct linking loaders, MSDOS linker.

Section2: Topics/Contents

Systems Programming for Linux as Open Source OS: Essential concepts of linux system programming, APIs and ABIs, standards, program segments/sections, the elf format, linking and loading, linux dynamic libraries (shared objects), dynamic linking, API compatibility, dynamically linked libraries.

Advanced system programming concepts: Operating system interfaces, stack smashing. Multitasking and paging, address translation, memory protection, comparison with windows.

Encoding, Decoding: Encoding and decoding schemes for the X-86 processor.

Device Driver: Types of drivers, driver history, driver issues, kernel level device drivers, virtual device drivers(VxD), device driver stack buses and physical devices, static device drivers, dynamic device drivers, PnP, device namespace, and named devices.

DOS: Internals of DOS, DOS loading, DOS memory map, Internal commands, External commands, command interpreter, POST details, POST sequence, PSP (structure details), '.exe' and '.com' file structures, conversion of .exe to .com file.

BIOS: what and why, BIOS calls: int 10h calls, dos calls: int 21h calls, difference between DOS and BIOS.

TSR: types, structure, details of TSR loading, examples, writing TSRs.

List of Project areas:

1. Design and implementation of 2 Pass Macroprocessor.
2. Design and implementation of 2 Pass Assembler.
3. Simulation of linker & loader.
4. Implement a Lexical Analyzer using LEX for a subset of C.
5. Design and implementation of DLL on Linux shared library.
6. Design a device driver on Linux system.

Text Books:

1. D M Dhamdhere; "Systems Programming & Operating Systems"; Tata McGraw Hill Publications, ISBN - 0074635794

2. John J Donovan; " Systems Programming " ; Tata Mc-Graw Hill edition , ISBN-13 978-0-07-460482-3

Reference Books:

1. Robert Love, " Linux System Programming " ;O'Reilly, ISBN 978-0-596-00958-8

2. Mahesh Jadhav; " Easy Linux Device Driver "; HighTechEasy publishing, Second edition.

3. Ray Duncan; "Advanced MSDOS programming"; Microsoft press

Course Outcomes:

The student will be able to –

1. Discriminate among different System software and their functionalities.
2. Design language translators like Macroprocessor and Assembler.
3. Develop approaches and methods for implementing compiler, linker and loader.
4. Adopt the skills and methods for implementing different system-level software.
5. Interpret the methods and techniques about instructions Encoding-Decoding and Implementing device drivers.
6. Design TSR programs for real world applications.

CS3006 :: AUGMENTED AND VIRTUAL REALITY

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Computer-Mediated Reality. Milgram's Reality-Virtuality continuum: Reality, Augmented Reality, Augmented Virtuality, Virtual Environment and Mixed Reality. Taxonomy of Mixed Reality: real, virtual, Extent of Work Knowledge (EWK), Reproduction Fidelity (RF), Extent of Presence Metaphor (EPM).

Introduction to AR, VR and MR: Differentiation, Features, use-cases and examples.

Geometry of Virtual World and Illumination: Birds-Eye View. Geometric Modelling. Matrix algebra and 2D rotations. 3D rotations and Yaw, Pitch and Roll. Axis angle representation. Quaternions. Converting and multiplying rotations. Homogeneous transforms. The chain of viewing transforms. Eye transforms. Viewport transforms. Three interpretations of light.

Refraction.Visual Perception: Depth perception, motion perception and frame rates.Visual Rendering: Overview, Shading models, rasterization, Pixel shading, Distortion shading. Tracking:Orientation tracking, tilt and yaw drift correction. Tracking with camera. Position tracking techniques. Interaction:selection, manipulation, isomorphic and non-isomorphic, exocentric and ego-centric interaction.Locomotion andDesign consideration.

Section 2: Topics/Contents

Basics of Image Processing: Color model: RGB andgrey. Basic linear and nonlinear operations.Image Enhancement Algorithms: contrast enhancement, histogram equalization. Segmentation.

Object Recognition and Tracking: edge and transition detection, circle, line and corner detection, smoothing, blurring, perspective recovery, feature point extraction: SIFT, Harris Corner Detection.

Vision based AR: Marker creation and marker tracking. Object tracking: Lucas Kanade Tracker, Optical Flow. Pose estimation. Rendering Techniques: Drawbacks of standard graphics libraries, artificial looks and aliasing artifacts due to rasterization. Lighting conditions, reflective properties estimation. Local illumination, secondary illumination, global illumination. 3D rendering, specialized rendering, wireframe rendering, and polygon-based rendering. Occlusion handling.

Location based Augmented Reality: GPS, gyroscope working and features. Sensor data fusion. Case study: Nokia City Lens/ Pokemon GO.

List of Practical:

7. Introduction to unity and 3D workspace and assets
8. Scripting in Unity, creating virtual environment and deploying app
9. Testing Cardboard unity sdk
10. Configuring cardboard sdk default app with Forest Environment with auto-locomotion
11. Introduction to Vuforia SDK
12. Creating a marker and rendering virtual object

Text Books:

3. "Augmented Reality and Virtual Reality The Power of AR and VR for Business" by tom Dieck, M. Claudia, Timothy Jung, Publisher: Springer; 1st ed. 2019 edition (February 19, 2019) ISBN-13: 978-3030062453
4. "Creating Augmented and Virtual Realities: Theory and Practice for Next-Generation Spatial Computing" by Erin Pangilinan, Steve Lukas, Vasanth Mohan, Publisher: O'Reilly Media; 1 edition (April 14, 2019), ISBN-13: 978-1492044192

Course Outcomes:

7. Identify the most suitable technique for a given use case based on the understanding of the similarities and differences between virtual, augmented and mixed reality with the help of Flynn's taxonomy
8. Design various transformations for manipulating an object in 3 dimensional vector space
9. Analyze rendering problems and rectify to provide realistic experience
10. Track visual cues for marker-based and marker-less augmented reality experience
11. Create local, global and secondary illumination for higher extent of presence metaphor
12. Extract sensory data to implement location based augmented reality experience

CS3018: NETWORK SECURITY

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Introduction: Introduction to Security: Vulnerabilities, Threats, Threat Modeling, Risk, attack and attack types, Avoiding attacks, Security services. Trustworthiness, Ethical issues and practices, Tradeoffs of balancing key security properties - Confidentiality, Integrity, Availability. Protocol Vulnerabilities: DoS and DDoS, session hijacking, ARP spoofing, Pharming attack, Dictionary Attacks. Software vulnerabilities: Phishing, buffer overflow, Cross-site scripting attack, Virus and Worm Features, Trojan horse, Social engineering attacks, ransomware, SYN-Flooding, SQL- injection, DNS poisoning, Sniffing Bitcoin and Crypto currency system.

Private key cryptography: Mathematical background for cryptography: modulo arithmetic, GCD (Euclids algorithm), algebraic structures (Groups, Rings, Fields, Polynomial Field). Role of random numbers in security, Importance of prime numbers

Data Encryption Standard: Block cipher, Stream cipher, Feistel structure, round function, block cipher modes of operation, S-DES, Attacks on DES, S-AES, AES.

Public key cryptography: RSA: RSA algorithm, Key generation in RSA, attacks on RSA. Diffie-Hellman key exchange: Algorithm, Key exchange protocol, Attack. Elliptic Curve Cryptography (ECC): Elliptic Curve over real numbers, Elliptic Curve over Zp, Elliptic Curve arithmetic. Diffie-Hellman key exchange using ECC.Chinese remainder theorem

Section2:Topics/Contents

Authentication and access control: Message authentication and Hash Function. Authentication: One-Way Authentication, Mutual Authentication, SHA-512, Centralized Authentication, The Needham-Schroeder Protocol. Authentication Applications: Kerberos, X.509 authentication service, public key infrastructure. Access Control in Operating Systems: Discretionary Access Control, Mandatory Access Control, Role Based Access Control.

Security application and design: Network layer security: IPSec for IPV4 and IPV6. Transport layer security: SSL and TLS. Application layer security: Security services, S/MIME, PGP, Https, Honey pots. Security design: End-to-end security, Security composability, Open design, Cost and tradeoffs

Cyber Security: Cyber Attack, Cyber Reconnaissance, Crimes in Cyber Space-Global Trends & classification, e-commerce security, Computer forensics, facebook forensic, mobile forensic, cyber forensic, digital forensic, Introduction to MQTT and CoAP for IoT.

List of Practical's:

1. Implementation of Caesar and Vigenere Cipher

2. Implementation of Playfair Cipher
3. Implementation of Hill Cipher
4. Implementation of S-RC4
5. Implementation of S-DES
6. Implementation of S-AES
7. Implementation of RSA.
8. Implementation of Diffie-Hellman key exchange
9. Implementation of ECC algorithm.
10. Study of Nessus tool
11. Study of Cuckoo Sandboxing.

Text Books:

5. "Cryptography and Network Security-Principles and Practices" by William Stallings, Pearson Education, 2006, ISBN 81-7758-774-9, 4th Edition.
6. "Network Security and Cryptography", by Bernard Menezes, Cengage Learning, 2010, ISBN 81-315-1349-1, 1st Edition.

Reference Books :

1. "Computer Security: Art and Science", by Matt Bishop, Pearson Education, 2002, ISBN 0201440997, 1st Edition.
2. "Network security, private communication in a public world", by Charlie Kaufman, Radia Perlman and Mike Spencer, Prentice Hall, 2002, ISBN 9780130460196, 2nd Edition.
3. "Cryptography and Information Security", by V.K. Pachghare, PHI, 2015, ISBN-978-81-203-5082-3, Second Edition.

Course Outcomes:

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

1. Analyze cryptographic techniques using a mathematical approach by examining nature of attack.
2. Establish type of attack on a given system.
3. Identify different types of attacks.
4. Justify various methods of authentication and access control for application of technologies to various sections of industry and society.
5. Design a secure system for protection from the various attacks for 7 layer model by determining the need of security from various departments of an organization.
6. Estimate future needs of security for a system by researching current environment on a continuous basis for the benefit of society.

MODULE VII/ VIII

Issue 1, Rev. 1 , Dt. 01/07/2018

Title: Course Structure **FF No. 653**

Branch: Computer **Year:**B.Tech. **A.Y.:** 2019-20 **Module:** VII **Pattern:** B-19.

Subject No.	Subject Code	Subject Name	Teaching Scheme (Hrs/Week)			Examination Scheme						Total	Credits
			Th	LAB	Tut.	ISE				ESE			
						Assignment (%)	Lab Assessment (%)	MSE %)	GD/PPT (%)	Viva/Lab Exam (%)	ESE (%)		
S1	CS4001 CS4002 CS4013	* OE-1 • Human Computer Interaction • Enterprise Systems • Convergence Technologies	3	2	-	20		30		20	30	100	4
S2	CS4003	* OE-2 • Cloud Computing • Parallel	3	2	-	20		30		20	30	100	4

	CS4004 CS4011	Computing on GPU • Natural Language Processing											
S3	CS4005 CS4006 CS4007 CS4008 CS4009 CS4010 CS4012 CS4014 CS4015	**OE -3 • Component-based System Construction • Data Mining & Data Warehousing • Modelling and Simulation • Internet of Things • Image Processing • Model-based Systems Engineering • Genetic Algorithms • Software	3	2		20		30		20	30	100	4

	CS4017 CS4018	• Mobile Computing • • Testing and Quality Assurance • Software Architecture and Design • Data Science											
PRJ.	CS4000	Major Project1	-	8	-	-	-	30		-	70	100	4
		TOTAL	**9**	**14**	**-**	**60**		**120**		**60**	**160**	**400**	**16**

CS4001: HUMAN COMPUTER INTERACTION

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Introduction to Human-Computer Interaction (HCI)

Human, Definition of Human Computer Interaction, Interdisciplinary Nature, Goals, Human Factors, Measurable Factors – Learn ability, Speed, Efficiency, Satisfaction. Early Focus on Users, Ergonomics, Usability, Types of Usability, User Interface (UI), Contexts - Web, Business, Mobile, Gaming Applications, Categorization of Applications based on Human Factors, Accessibility and Security.

Principles and Models

Eight Golden Rules of Interface Design, Principles of Good Design, Faulty Designs, Miller's Principle, Norman's Action Model, Gulf of Execution and Evaluation, Errors – Mistakes, Slips, Lapses and Violations, Guidelines for Data Display, Guidelines for Data Entry, Conceptual, Semantic, Syntactic and Lexical Model, Task Analysis, GOMS, Keystroke-Level Model, User Persona, UI Standards and GUI Libraries.

Design Process and Interaction Styles

Design, Three Pillars of Design, Process of Design, Ethnographic Observations, Contextual Inquiry, Iterative Design, Participatory Design, Navigation Design, Visual Design, - Layout, Color, Fonts, Labeling, LUCID, Scenarios, Interaction Styles - Direct Manipulation, Menu Selection, Form-Filling, Commands, Natural Language, Internationalization, Interaction Design Patterns.

Section2:Topics/Contents

Evaluation Techniques and Interface Categories

Expert-based Evaluation, User-based Evaluation, Heuristic Evaluation, Cognitive Walkthrough, Semiotic Analysis, Expert Reviews, Usability Testing, User Surveys, Interviews, Think Aloud, Acceptance Tests, Statistical Methods, Touch Interfaces, Public Place Interfaces, Wearable Interfaces, Tangible Interfaces, Intelligent Interfaces, Ubiquitous and Context-Aware Interaction.

Documentation and Groupware

Classification of Documents, Printed Manuals, Reading from Displays, Online Help, Tutorial, Error / Warning Messages, Groupware, Goals / Dimensions of Cooperation, Asynchronous Interactions, Synchronous Interactions, Online Communities, Communityware

Miscellaneous

Case Studies: Web Usability, Mobile Usability, Embedded Systems, Social Networking Sites, Messengers, E-Governance Sites, Security Tools, e-Health applications

List of Project areas: (For THL, TLP courses)

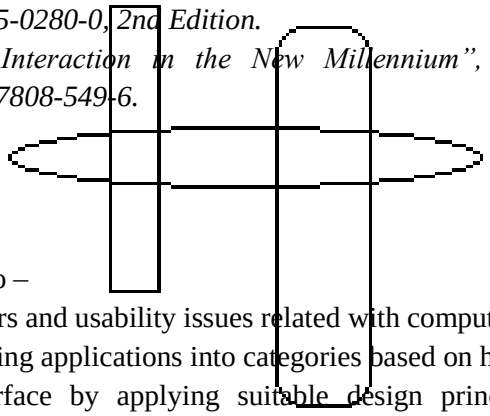
1. Identify specialized users and related facilities for a selected product / system and make necessary suggestions for its improved accessibility design.
2. Design user persona for the users of selected product / system.
3. Conduct a contextual inquiry for selected product / system.
4. Design an interface prototype for selected product / system.
5. Evaluate an interface using usability evaluation technique.

Text Books:

1. *"Human-Computer Interaction", Alan Dix, Janet Finlay, Gregory D. Abowd, Russell Beale, Pearson Education, ISBN 81- 297-0409-9, 3rd Edition.*
2. *"Designing the User Interface", Ben Shneiderman, Pearson Education, ISBN 81-7808-262-4, 3rd Edition*

Reference Books:

1. *"The Design of Everyday Things", Donald Norman, Basic Books, ISBN 100-465-06710-7, 2002 Edition*
2. *"The Essential Guide to User Interface Design", Wilbert O. Galitz, Wiley-dreamtech India (P) Ltd., ISBN 81-265-0280-0, 2nd Edition.*
3. *"Human-Computer Interaction in the New Millennium", John M. Carroll, Pearson Education, ISBN 81-7808-549-6.*



Course Outcomes:

The student will be able to –

1. Identify human factors and usability issues related with computing applications
2. Differentiate computing applications into categories based on human factors
3. Design a user interface by applying suitable design principles, models and usability guidelines
4. Integrate ethno-cultural and accessibility computing aspects into the user interface design
5. Display the impact of usability evaluation and testing in computing applications
6. Follow required processes and standards while designing user interfaces.

CS 4002: ENTERPRISE SYSTEMS

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Flexible Systems: The Reality of Imperfect Knowledge and Flexibility, Characterizing Flexible Software, Software Change, Evolution of Flexibility, Software basis: Outcome, Not Methodology, Software Engineering Shifts, Roles and Responsibilities in Traditional and Flexible Development, Guidelines for Flexible Software Design, The Importance of Stable Identifiers, Regulation: Managing Artificial Limits., Stable Information Structures, The Generic-Entity Cloud, Flexibility and Strategic Systems Planning, Requirements Determination for Flexible Systems, System Design with an Eye on Flexibility, Implementing Stable Identifiers, Testing and Maintenance of Flexible Software, Identifying, Managing Quality Flexible Software

SOA Fundamentals: Service-Oriented Computing and SOA , Introduction to Service-Oriented Computing, The Evolution of SOA, Principles of Service-Orientation, Goals and Benefits of Service-Oriented Computing, Service-Orientation, Problems Solved by Service-Orientation, Challenges Introduced by Service-Orientation, Effects of Service-Orientation on the Enterprise, Origins and Influences of Service-Orientation, Understanding Design Principles, Principle Profiles, Design Pattern References, Principles and Design Granularity

Service Contracts (Standardization and Design): Contracts principles, Types of Service Contract Standardization, Contracts and Service Design, Versioning, Technology / Development Tool Dependencies

Service Coupling (Intra-Service and Consumer Dependencies): Coupling principles, Service Contract Coupling Types, Service Consumer Coupling Types, Service Loose Coupling and Service Design, Enterprise Service Bus, Web Services and Primitive SOA, Web Services and Contemporary SOA, Service Layers

SOA Design Principles: Service Abstraction (Information Hiding and Meta Abstraction Types): Abstraction principles, Types of Meta Abstraction, Measuring Service Abstraction, Service Abstraction and Service Design, Risks with Service Abstraction

Service Reusability (Commercial and Agnostic Design): Reuse Principle, Service Reuse in SOA, Service Reusability and Service Design

Service Autonomy (Processing Boundaries and Control): Autonomy Principle, Types of Service Autonomy, Measuring Service Autonomy, Service Contract Autonomy (services with normalized contracts) Autonomy and Service Design

Service Statelessness (State Management Deferral and Stateless Design): State Management, Measuring Service Statelessness, Statelessness and Service Design

SOA Delivery Strategies, Service-Oriented Analysis: Introduction, Service Modeling, Service-Oriented Design: Introduction, SOA Composition Guidelines), Service Design, Importance of WSDL, SOAP, The use of registries via UDDI

Section2:Topics/Contents

SOA Technology and Implementation: Service Discoverability (Interpretability and Communication): Discoverability, Types of Discovery and Discoverability, Measuring Service Discoverability, Discoverability and Service

Service Composability (Composition Member Design and Complex Compositions): Composition, Composition Concepts and Terminology, Complex Service Composition, Measuring Service Composability and Composition Effectiveness Potential, Composition and Service Design, Service-Orientation and Object-Orientation, Mapping Service-Orientation Principles to Strategic Goals, SOA Platforms

Enterprise Architecture: Introduction to Enterprise Architecture, State of the Art and Foundations of Enterprise Architecture, Communication of Enterprise Architecture, Language for Enterprise Modeling, Viewpoints and Visualization, Architecture Analysis, Architecture Alignment, Tool Support, Domain-Driven Architecture, Resource-Oriented Architecture, Defining EAI, The EAI Process, Data-Level EAI, Application Interface-Level EAI, Method-Level EAI, User Interface-Level EAI, EAI Inrteroperability

Enterprise Architecture Frameworks: The Open Group Architecture Framework (TOGAF), Zachman Enterprise Architecture Framework, Extended Enterprise Architecture Framework (E2AF), Federal Enterprise Architecture Framework (FEAF), Treasury Enterprise Architecture Framework (TEAF), Capgemini's Integrated Architecture Framework (IAF), Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance, (C4ISR) and DoDAF, TAFIM, Computer Integrated Manufacturing Open System Architecture (CIMOSA), Purdue Enterprise Reference Architecture (PERA), IEEE Std 1471-2000

List of Practicals: (For THL, TLP courses)

1. To narrate concise Requirement Definition Document and System Requirements Specification Document for target system with reference to the IEEE 610.12.1990 Standard guidelines clearly indicating the requirements considerations.
2. To decompose and organize the problem domain area into broad subject areas and identify the boundaries of problem/system along with identification of Business Processes and develop full detail Business Process diagrams.
3. To develop Domain-driven vocabulary of the target system indicating domain lexicon and context-based terminologies.
4. To identify and categorize the target system services with detailed service specifications modeled with component diagram incorporating appropriate architectural style and coupling.
5. To design the service layers and tiers modeled with deployment diagram accommodating abstraction, autonomy, statelessness and reuse.
6. To map the service levels and primitives to appropriate Strategies for data processing using XML / XQuery/ JSON / JAXB.
7. To produce, invoke, compose Web Services using SOAP, WSDL and UDDI.
8. To implement and integrate the components of the target system using .NET / J2EE platforms adhering to Service specifications.

9. To create the balanced scorecard for the target system indicating the standards and principles applied.

Text Books:

1. *Mathias Weske, Business Process Management, Concepts, Languages, Architectures, ISBN 978-3-540-73521-2 Springer Berlin Heidelberg New York, 2007*
2. *Thomas Erl, Service-Oriented Architecture: Concepts, Technology, and Design. ISBN: 0-13-185858-*0, Publisher: Prentice Hall PTR, 2005

***Reference Books* :**

1. *Thomas Erl, SOA Principles of Service Design, Pearson Education, Inc., ISBN 0-13-234482-3, 2007*
2. *Eric A. Marks, Michael Bell., Executive's guide to service-oriented architecture, John Wiley & Sons, Inc.ISBN-13: 978-0-471-76894-4, 2006*
3. *Daniel Minoli, Enterprise Architecture A to Z, Frameworks, Business Process Modeling, SOA, and Infrastructure Technology, Auerbach Publications, Taylor & Francis Group, ISBN 978-0-8493-8517-9, 2008*
4. *SetragKhoshafian, Service Oriented Enterprises, Auerbach Publications, Taylor & Francis Group, ISBN 0-8493-5360-2, 2007*
5. *Mike Rosen, Boris Lublinsky, Kevin T. Smith, Marc J. Balcer, Applied SOA: Service-Oriented Architecture and Design Strategies, Wiley Publishing, Inc., ISBN: 978-0-470-22365-9, 2008*
6. *Marc Lankhorst et al., Enterprise Architecture at Work, Modelling, Communication and Analysis, Second Edition, ISBN 978-3-642-01309-6, Springer-Verlag Berlin Heidelberg 2009*
7. *David S. Linthicum,Enterprise Application Integration, Addison-Wesley Professional 2003, ISBN-10: 1402052626*

Course Outcomes:

Upon completion of the course, graduates will be able to -

1. Model business requirements and business processes using BPMN 2.0 standard encompassing Process Orchestrations and Choreographies.
2. Discover the set of services with composite services creation and designing services to facilitate integration and understand interrelationships among SOA, Web Services, OOD and IT infrastructure.
3. Explore the concepts, guidelines and technology for service orchestration to integrate a Business Process Management Solution in an Enterprise SOA in societal context.
4. Prepare well-formed specifications and reports for service composition and delivery to the stakeholders.
5. Understand case studies and lessons learned with utilization of Enterprise Architecture Integration and Frameworks knowledge towards planning and implementing complex enterprise projects.
6. Create sustainable Enterprise System design supported by enterprise modelling, architecture analysis and alignment.

CS4013: CONVERGENCE TECHNOLOGIES

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Introduction to digital and IP Telephony: Digital Telephony: Circuit Switched networks, SS7, ISDN, Exchanges, E.164 Numbering Plans. IP Telephony: Packet Switched Networks, Signaling & Media separation, Media Encapsulation RTP and RTCP, Audio and Video Codecs.

VoIP Protocols: H.323 Network Elements: Terminals, Gateway, Gatekeeper, Multipoint Control Unit. H.323 Protocol: RAS Channel, H.225 Call signaling, H.245 Media signaling.H.323 Call flows: Basic Audio and Video Call flows. SIP Network Elements: Registrar, Proxy, UAS, UAC, B2BUA. SIP Protocol: Requests & Responses, Methods, Headers & Parameters, Message Structure, Transactions & Dialogs, Session Description Protocol. SIP Call Flows: Basic Audio and Video Call Flows. H.248 Protocol: Media Gateways, Media Gateway Controllers, Commands, Transactions, Contexts, Terminations, Descriptors, Packages

Unified Communications: Local and Network features: Call Forward, Call Coverage, Automatic Call Back, User Displays, Resource Optimization. Voice & Data Integration: IM, presence, voice mail. Collaboration: Call Conferencing, Voice, Video, Data & Content integration. Mobility: Mobile Clients, Session Border Controllers. Business Applications: Framework for custom applications, Computer Telephony Interface, Application Sequencing.

Section2: Topics/Contents

Contact Center and Applications: Contact Centers : Introduction & evolution of Call Centers to Context Centers, Services to Markets, ACD, VDN, Skills, Agents & States, Selection Algorithms, Service Observing, Call Recording, Assisted Service, Self Service, Intelligent Routing, Business use cases. Inbound/Outbound Contact Centers: Introduction, Services to Markets, Inbound channels/ outbound channels (Voice, SMS, Email & Chat), Text-to-Speech Analytics, Agent and Notification Campaigns, Pacing Algorithms, Business use cases. Reporting & Analytics: Blended Contact centers, Real-time and Historical reporting, Types of Reports, Business use cases. Analytics: Agent Performance, Occupancy, Campaign details, Collections

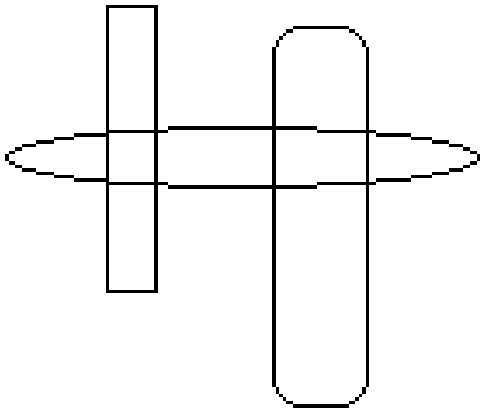
Emerging technologies in Telecommunications: High Availability: Load balancing, Reliability, Failover & Failback, Location Redundancy, Hardware footprint, Cloud Computing: Applications in Telecommunications Analytics in Voice & Data, Diagnostics & Management Emerging Technologies: Google Glass, webRTC, Cloud Hosting.

List of Practical: (For THL, TLP courses)

Text Books:

Reference Books:

Course Outcomes:
The student will be able to –



CS 4003: CLOUD COMPUTING

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
: Lab 2 Hours / Week

Section 1: Topics/Contents

Overview of computing paradigm: Recent trends in Computing - Grid Computing, Cluster Computing, Distributed Computing, Utility Computing, Cloud Computing. Evolution of cloud computing - Business driver for adopting cloud computing.

Introduction to Cloud Computing: Cloud Computing - Introduction to Cloud Computing, History of Cloud Computing, Cloud service providers. Properties, Characteristics & Disadvantages - Pros and Cons of Cloud Computing, Benefits of Cloud Computing, Cloud computing vs. Cluster computing vs. Grid computing. Role of Open Standards.

Cloud Computing Architecture: Cloud computing stack - Comparison with traditional computing architecture (client/server), Services provided at various levels, How Cloud Computing Works, Role of Networks in Cloud computing, protocols used, Role of Web services. Service Models (XaaS) - Infrastructure as a Service(IaaS), Platform as a Service(PaaS), Software as a Service(SaaS). Deployment Models, Public cloud, Private cloud, Hybrid cloud, Community cloud

Infrastructure as a Service (IaaS): Introduction to IaaS - IaaS definition, Introduction to virtualization, Different approaches to virtualization, Hypervisors, Machine Image, Virtual Machine(VM). Resource Virtualization - Server, Storage, Network. Virtual Machine(resource) provisioning and manageability, storage as a service, Data storage in cloud computing(storage as a service). Renting, EC2 Compute Unit, Platform and Storage, pricing, customers.

Platform as a Service (PaaS): Introduction to PaaS - What is PaaS, Service Oriented Architecture (SOA). Cloud Platform and Management - computation, storage

Software as a Service (SaaS): Introduction to SaaS, Web services, Web 2.0, Web OS, Case Study on SaaS

Section2:Topics/Contents

Service Management in Cloud Computing: Service Level Agreements(SLAs), Billing & Accounting, Comparing Scaling Hardware: Traditional vs. Cloud, Economics of scaling: Benefitting enormously, Managing Data - Looking at Data, Scalability & Cloud Services, Database & Data Stores in Cloud, Large Scale Data Processing

Cloud Security: Infrastructure Security - Network level security, Host level security, Application level security. Data security and Storage - Data privacy and security Issues, Jurisdictional issues raised by Data location: Identity & Access Management, Access Control, Trust, Reputation, Risk,

Authentication in cloud computing, Client access in cloud, Cloud contracting Model, Commercial and business considerations

Case study on Open Source and Commercial Clouds – Amazon EC2, Google Compute Engine, Microsoft Azure, Cloudfoundry, OpenStack

---

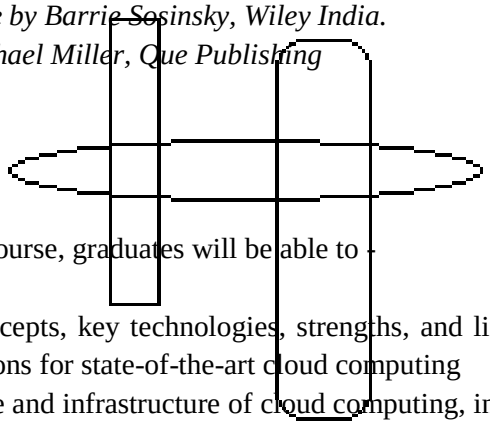
List of Practicals: (For THL, TLP courses)

Text Books:

1. *Cloud Computing for Dummies by Judith Hurwitz, R. Bloor, M.Kanfman, F.Halper (Wiley India Edition).*
2. *Enterprise Cloud Computing by Gautam Shroff,Cambridge.*
3. *Cloud Security by Ronald Krutz and Russell Dean Vines, Wiley-India*

Reference Books :

1. Google Apps by Scott Granneman,Pearson.
2. Cloud Security & Privacy by Tim Malhar, S.Kumaraswammy, S.Latif (SPD,O'REILLY)
3. Cloud Computing : A Practical Approach, Antohy T Velte, et.al McGraw Hill,
4. Cloud Computing Bible by Barrie Sosinsky, Wiley India.
5. Cloud Computing, Michael Miller, Que Publishing



Course Outcomes:

Upon completion of the course, graduates will be able to -

1. Describe the main concepts, key technologies, strengths, and limitations of cloud computing and the possible applications for state-of-the-art cloud computing
2. Explain the architecture and infrastructure of cloud computing, including SaaS, PaaS, IaaS, public cloud, private cloud, hybrid cloud, etc.
3. Collaboratively research and write a paper on the state of the art (and open problems) in cloud computing.
4. Identify problems, and explain, analyze, and evaluate various cloud computing solutions.
5. Choose the appropriate technologies, algorithms, and approaches for the related issues.
6. Display new ideas and innovations in cloud computing.

CS4004: PARALLEL COMPUTING ON GPU

Credits: 4 **Teaching Scheme: Theory 3 Hours / Week**
Lab 2 Hours / Week

Section 1: Topics/Contents

Introduction to Parallel Computing: Motivating Parallelism, Scope of Parallel Computing, Parallelism Vs. Concurrency, Types and levels of parallelism, Flynn's classification, Parallel Algorithms, Analysis of parallel algorithms,Amdhal's law, Performance issues in algorithms.Parallel computer architectures : PRAM, Distributed memory systems ,Shared memory systems and cache coherence, Heterogeneous system architecture. Parallel programming models: Foster's design methodology. Parallel Programming constructs: Synchronization, Deadlocks, Critical sections, Data sharing etc**.** Comparison of GPU and CPU architectures,GPUs as Parallel Computers,Applications of GPU . Parallel Programming Languages and Models.

Introduction to OpenMp: Multicore Architecture,Programming Environment,Programming on Multi-Core Processors, Pthreads & OpenMP,Performance Issues: Example programming on Numerical & non-numerical computations using Pthreads; Multi-Threaded I/O, MPI-OpenMP, MPI-Pthreads,Example programs using Performance Visualization Tools- Intel Vtune Performance analyzer,Open source Software tools - PAPI - on Multi-Core Processors; Compiler Optimization Techniques;

Parallel Programming Model: Common Unified Device Architecture(CUDA), CUDA programming model, Concept of grid, block and thread, thread index generation, warp. Programming for GPU's in C/C++ using CUDA API: Memory transfers, Writing and executing kernel functions, Writing device functions, Thread synchronization, Data Dependences and Race Conditions.,Organizing Parallel Threads.

Section2: Topics/Contents

GPU Architecture: GPU architecture, Overview of the graphics pipeline, Components of GPU : Parallel streaming processors ,Multiprocessors ,Shared instruction caches ,Memory hierarchy – Global, Constant, Shared, and Texture memory; Case studies: NVIDIA Kepler K20/K40/AMD.

Memory Organization and Optimization: Global, Shared, constant and texture memory. Memory coalescing, memory banks and bank conflicts, Page locked host memory.Reduction operation, CUDA code optimization. Need of profilers and analyzers, Introduction to CUDA Tools: MemCheck, Command line & Visual Profilers.

Problem solving using GPUs : Single vs. double precision, Sparse matrix representations, Fast Fourier transforms, Binomial coefficients, light weight scientific computing exercises, Matrices etc .

List of Project areas: (For THP, TLP courses)

1. Analysis of Parallel Algorithms
2. Implementation using OpenMP
3. GPU kernel implementation for the given application.
4. Performance analysis using GPU memories.
5. Kernel reduction.
6. Profiling an application.

Text Books: ***(As per IEEE format)***

1.

3.

Reference Books:

1.*Hwang and Briggs,"Computer Architecture and Parallel Processing", Tata McGraw Hill Publication ISBN 13: 9780070315563.*

Course Outcomes:

The student will be able to –

1.

CS4011: Natural Language Processing

Credits: 4 **Teaching Scheme: Theory: 3 Hours / Week**
Lab: 2 Hours / Week

Section 1: Natural Language Processing

Origins of NLP, Challenges of NLP, Language and Knowledge, Processing Indian Languages, **Formal Language Theory:** Basic Notions, Regular Expressions and Automata:, Basic Regular Expression Patterns, Disjunction, Grouping and Precedence, Advanced Operators, Substitution, Finite State Automata, NFSA, **Words and Transducers:** Morphology, Inflectional Morphology, Derivational Morphology, Finite State Morphological Parsing, Construction of Finite State Lexicon, Finite State Transducers, FST for Morphological Parsing, **Language Modelling:** Grammar-based language models, lexical functional Grammar(LFG), Government and Binding (GB), Lexical functional Grammar Model, Generative grammars, **Theory of parsing / Syntactic Analysis:** Context Free Grammar, parsing, Top-down Parsing, Bottom-up parsing, Probabilistic parsing, Indian Languages parsing **Semantic Analysis:** Meaning Representation, Lexical Semantic, Ambiguity, Word Sense Disambiguation

Section2: Computational Linguistics Using Machine Learning

Mathematical Foundations for Machine Learning: Hypothesis, Target Function, Cost Function, Gradient, Training, Testing, Cross-validation, Linear and Logistic Regression, K-Nearest Neighbor, K-Means

Statistical Language modeling: N-gram model, Machine Transliteration: Rule-based, Phonology and Stress Analysis based and Statistical based, Support vector machine, Memory Entropy Model, Conditional Random Fields

Machine Translation: Rule-Based, Statistical-Based using MT Tools - GIZA++, SRTLM and Moses

List of Practical's: (For THP)

Operating System recommended: - 64-bit Open source Linux or its derivative
Programming tools recommended: - Open Source C, C++, JAVA, and PYTHON, Programming tool like G++ /GCC, WEKA, GIZA++, MOSES, SRTLM

1. **Project 1 :**
   POS Taggers For Indian Languages
2. **Project 02:**
   Rule based Machine Translation for phrases/form labels

3. Project 02:

Machine Translation/Transliteration using Statistical Machine Translation

Text Books:

5. *Tanveer Siddiqui and U S Tiwary, "Natural Language Processing and Information Retrieval" Fourth Impression, Oxford, ISBN-13:978-019-569232-7.*

6. *Daniel Jurafsky and James H Martin., "Speech and Language Processing", 2nd edition, Pearson, Second Impression-2014,ISBN: 978-93-325-1841-4*

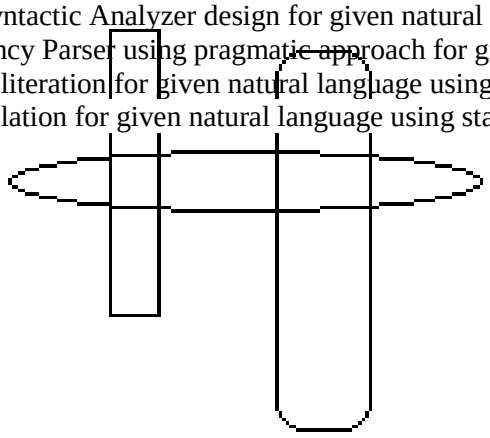
Reference Books:

9. *Alexander Clark, Chris Fox and Shalom Lappin "The Handbook of Computational Linguistics and Natural Language Processing",Wiley-Blackwell-2013, ISBN-978-1-118-34718-8*

Course Outcomes:

Course Outcomes:

1. Analyze Morphology for given natural language
2. Apply different approaches to design Lexical Analyzer for given natural language
3. Evaluate and devise Syntactic Analyzer design for given natural language
4. Design Type Dependency Parser using pragmatic approach for given natural language
5. Develop machine transliteration for given natural language using statistical approach
6. Develop machine translation for given natural language using statistical approach



CS 4005: COMPONENT-BASED SYSTEM CONSTRUCTION

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
: Lab 2 Hours / Week

Section 1: Topics/Contents

Overview of CBSE: Introduction, Market versus technology, Standards, What a component is and is not, Components, interfaces, and re-entrance, Basic Concepts in CBSE, Specification of Software Components, Architecting Component-Based Systems, Component Models and Technology, CBD Process, Adapting the V-Model for CBD

Software Composition: Different Views of Software Composition, Software Composition Mechanisms, Algebraic Composition Mechanisms, Mathematical Composition Operators, Semantic Integrity in CBD, Role-Based Component Engineering, Dispelling the Myth of Component Evaluation, Component Composition and Integration, Predicting System Trustworthiness, Polymorphism, Object versus class composition or how to avoid inheritance, Aspects of scale and granularity, Patterns, frameworks, architectures,

Business Components: The Business Component Approach: Levels of Component Granularity, Architectural Viewpoints, Development Process, Business Component, Business Component Factory, Scenario, Applicability of the Approach, Distributed Component: Concepts, Internals, Externals, Business Component: Concepts, Internals, Externals, Development Lifecycle, Address Book Scenario, Relation to SE, Business Component System: Concepts, Internals, Externals, Federation of System-Level Components: The Business Problem, Interoperability Concepts, Federation Concepts, Architected Federation, Technical Architecture: Concepts, Technical Core, Services and Facilities, Extended Integrated Development Environment, Application Architecture: Concepts, Architectural Principles and Styles, Collaboration Patterns, Project Management Architecture: Concepts, SCM and Versioning, Dependency Management, Development Model and Environment, Component-Based Business Modeling: Concepts, Identification Strategies, Modeling the Processes, Modeling the Entities, Transitioning

Section2:Topics/Contents

Component Architectures and implementations: Component Models with Objects as Components, Component Models with Architectural Units as Components, Component Models with Encapsulated Components, Programming – shades of gray, Object and component "wiring" standards, The OMG way: CORBA, CCM, OMA, and MDA, The Sun way – Java, JavaBeans, EJB, and Java 2 editions, The Microsoft way: COM, OLE/ActiveX, COM+, and .NET CLR, Strategic comparison, Efforts on domain standards, Ongoing concerns, Component architecture, Component frameworks, Component development, Component distribution and acquisition, Component assembly, Gamut of markets, New professions, A component marketing paradox

Legacy Systems: Application, Data, and Time Stovepipes, Managing the Evolution of Enterprise Applications, Legacy Portfolio Analysis. Wrappers, Connectors, Mediators, Adapters, The Landscape of Legacy Wrappers, A Roadmap for Developing Wrappers, Modularizing Legacy Systems, Constructing WSDL/SOAP-Based Wrappers, Developing Enterprise Applications: Loosely Coupled versus Tightly Coupled Networked Enterprises, Single Organizations, Networked Organizations, Toward a Methodological Framework, Methodological Framework: Overview, Forward Engineering, Reverse Engineering, Matching Phase, Adaptation Phase, Matching Phase: Structural Matching, Semantic Matching, Metamodel-Driven Matching Component Adaptation, Parametric Contracts, Adapter Generation

Software Reuse and Agile: Technical Aspects of Software Reuse, Nontechnical Aspects of Software Reuse, Installing a Reuse Program, Component Composition and Attributes, Domain Engineering, Component Engineering, Application Engineering, Software Documentation, Reuse Documentation, Literate Programming, Reuse Measurement in Literate Programs, Documentation Reuse, On Components and the Nature of Reality: Introduction to the Metaworld, The Universal Perspective: Scope of Business, On the Nature of Change and Winged Pigs , Managing Emotions Unleashed by Change, Governance of Change, Components in Product Line Architectures, Koala Component Model

List of Practicals: (For THL, TLP courses)

1. To narrate concise Requirement Definition Document and System Requirements Specification Document for target system with reference to the IEEE 610.12.1990 Standard guidelines clearly indicating the requirements considerations.
2. To decompose and organize the problem domain area into broad subject areas and identify the boundaries of problem/system along with identification of Business Processes and develop full detail Business Process diagrams.
3. To develop Component vocabulary of the target system indicating component lexicon and context-based terminologies.
4. To identify and categorize the target system component services with detailed component specifications modeled with component diagram incorporating appropriate architectural style and coupling.
5. To design the component layers and tiers modeled with deployment diagram accommodating abstraction, autonomy, statelessness and reuse.
6. To map the component service levels and primitives to appropriate Strategies for data processing using applicable application architectures.
7. To produce, invoke, compose Component Services using Component-Service Wrapping-Wiring.
8. To implement and integrate the components of the target system using sufficient platforms adhering to Component specifications.

Text Books:

1. *IvicaCrnkovic, Magnus Larsson, Building Reliable Component-Based Software Systems, 2002 Artech House, Inc., ISBN 1-58053-327-2*
2. *Johannes Sametinger, Software Engineering with Reusable Components, Springer-Verlag, 1997, ISBN 978-3-662-03345-6*

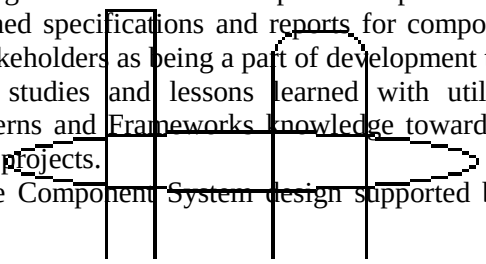
***Reference Books*:**

1. *Kung-Kiu Lau, Simone Di Cola, An Introduction To Component-Based Software Development (Series On Component-Based Software Development), World Scientific Publishing Co. Pte. Ltd., ISBN -13: 978-9813221871*
2. *Peter Herzum, Oliver -Sims, Business Component Factory: Comprehensive Overview Of Component-Based Development For The Enterprise, John Wiley & Sons, Inc, 2000, ISBN-13: 978-0471327608*
3. *Willem-Jan van den Heuvel, Aligning Modern Business Processes and Legacy Systems, 2007 MIT Press, ISBN-13: 978-0-262-22079-8*
4. *Clemens Szyperski, Component Software: Beyond Object-Oriented Programming, Pearson Education Limited 2002, ISBN 0-201-74572-0*

Course Outcomes:

Upon completion of the course, graduates will be able to -

1. Model business requirements and business processes using BPMN 2.0 standard encompassing Process Orchestrations and Choreographies.
2. Discover the set of component services with composite services creation and designing services to facilitate integration in IT infrastructure.
3. Explore the concepts, guidelines and technology for component orchestration to integrate a Component Design Solution in an Enterprise Component Systems in societal context.
4. Prepare well-formed specifications and reports for component service composition and delivery to the stakeholders as being a part of development team.
5. Understand case studies and lessons learned with utilization of Component-based development patterns and Frameworks knowledge towards planning and implementing complex business projects.
6. Create sustainable Component System design supported by reuse, documentation, and testability.



CS4006: DATA MINING AND DATA WAREHOUSING

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Introduction to DWH : Data warehouse (DWH): Need, Definition, Advantages of DWH, OLTP Vs DWH, 3-tier Architecture, DWH Design Process, ETL Process, DWH Back-end Tools and Utilities, Metadata Repository, Models of DWH: Enterprise Warehouse, Data Mart, Virtual Warehouse, Comparison.

Dimensional Modeling: Dimensional Model Vs ER Model, DWH Schemas: Star, Snowflake, Fact Constellation, their Comparison, Techniques to Handle Changing Dimensions, Aggregation, Families of Fact Tables, Fact Less Fact Tables; Data Warehouse Indexing: Factors used to select an Indexing Technique, Properties of a Good Indexing Technique for DWH, Indexing Techniques: Projection Index, Bitmap Index (Pure and Encoded), Join Index and their Comparison.

Data Mining and Functionalities : Need of Data Mining, Knowledge Discovery in Database (KDD), Architecture of Data Mining System, Data Mining on Different kind of Data, Data Mining Functionalities; Data Preprocessing: Need, Cleaning, Integration, Transformation, Reduction, Discretization, Concept Hierarchy Generation.

Section2: Topics/Contents

Cluster Analysis: Categories of Clustering methods, Partitioning methods: k-Means, k-Medoids ; Prediction: Numerical Prediction, Linear, Non-Linear Regression; Outlier Analysis: Applications, Techniques.

Classification: Decision Tree Classifier, Rule Based Classification, Bayesian Classification, Neural Network Classification: Back Propagation Algorithm, Lazy Learner: kNN Classifier, Case-Based Reasoning, Other: Fuzzy Set Approach, Classifier Accuracy Measures, Techniques for Evaluating Classifier Accuracy; Frequent Itemset Mining: Interesting Item Set Mining: Market Basket Analysis, APriori Algorithm, Generating Association Rules, Types of Association Rules, Correlation Analysis.

Data Mining on different Databases: Multimedia Data Mining, Web Mining, Text Mining, Spatial Data Mining, Mining on Social Networks, Multi-relational Data Mining.

List of Practicals: (For THL, TLP courses)

1. Design a star / snowflake schema for a data warehouse for an organization.
2. Use ETL tool for extract-transform-load operations
3. OLAP Data cube implementation and operations
4. Data mining: Implement clustering algorithms
5. Classification algorithms
6. Association rule mining
7. Prediction, Linear Regression
8. Outlier analysis

Text Books:

1. "The Data Warehouse Lifecycle Toolkit", Kimball, Reeves, Ross, Thornthwaite, John Wiley, ISBN 9971-51-415-X, 2002 Edition.
2. *"Data Mining: Concepts and Techniques", Jiawei Han and Micheline Kamber, Morgan Kaufman, ISBN 978-81-312-0535-8, 2nd Edition.*

Reference Books:

1. "Decision Support and Data Warehouse Systems", Mallach Efrem G, Tata McGraw Hill, ISBN 978-0070486843, 2009 Edition.
2. *"Mastering Data Mining: The art and science of customer relationship management", M Berry and G. Linoff, John Wiley, ISBN 9971-51-369-2, 2001 Edition.*

Course Outcomes:

The student will be able to –

1. Construct an end-to-end data warehousing solution.
2. Evaluate various data processing algorithms in their applicability to different problems
3. Display the process of converting data into a user defined format required for particular analysis
4. Utilize statistical tools in deriving insights from data
5. Describe various techniques for clustering and classification
6. Apply various techniques to solve real-world data analysis problems

CS4007: MODELING AND SIMULATION

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
: Lab 2 Hours / Week

Section 1: Topics/Contents

Process of Modeling and Simulation: What is M&S, Need for Abstraction, Relationship between modeling and simulation Process of modeling: Problem identification and formulation, Real system data collection, Model development, Validation, Experiment design, Simulation runs and Results interpretation.

Formal models and modeling techniques: Monte Carlo methods, Stochastic processes, Queuing theory: Little's Theorem and applications, M/M/1 Queuing System, Petri nets, Game theory, State spaces and transitions, Graph structures: directed graphs, trees, networks.

Discrete Event Simulation : Deterministic vs. stochastic simulation, Static vs. Dynamic Simulation, Constructing dynamic stochastic simulation models, Time keeping, Event Scheduling, State transition, Time driven and event-driven models, Pseudo-random number generation.

Section2: Topics/Contents

Agent-based simulation : Modeling Complex Systems, Agents, environments, ABMS: When and Why, Agent based model design, Autonomous Agents, Agent Interaction, Topologies and Neighborhoods, Tools for ABMS: Repast, Swarm, NetLogo.

M&S Applications and Awareness : Application areas: optimization, decision making support, forecasting, safety considerations, training and education. ABMS Applications: Social networks, Organizations, Markets, Flows, Epidemiology, Diffusion.

Advanced Topics: Model scalability, Virtual Reality, Virtual Worlds, Intro to Rare Event Simulation, Intro to Parallel Discrete Event Simulation, PDES Challenges.

List of Practicals: (For THL, TLP courses)

1. Develop a simple deterministic simulation to determine the loan tenure for Rs.X principal amount when the customer pays Rs.Y per month. Assume the fixed interest rate of 10% per year.

2. Develop a Monte Carlo simulation model for profit estimation before introducing a new product in the market. Consider the uncertainty in terms of sales, production costs, competitive pricing and other market dynamics.

3. Develop a discrete event simulation of a typical fast-food restaurant. Restaurant configuration, business factors and customer behavior factors should be tunable parameters. (SimPy)

4. Agent based simulation : Marketing/Diffusion: Word of Mouth publicity, Epidemiology: spread of disease. (NetLogo)

5. Develop a parallel discrete event simulation for a network of routers using conservative event processing.

Text Books:

1. Discrete Event Simulation: A First Course, L. Leemis and S. Park, 2006, Prentice-Hall.

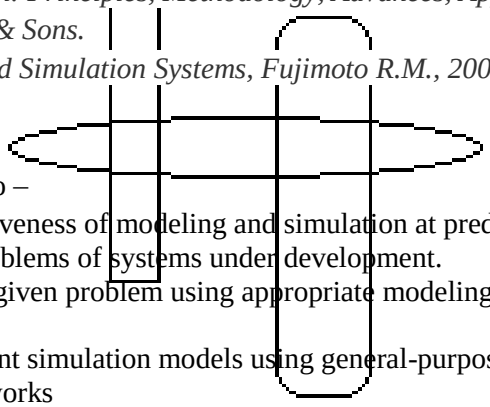
2. Agent-Based Models, Nigel Gilbert, 2008, SAGE Publications.

3. System Simulation and Modeling, Sankar Sengupta, 2013, Pearson Education.

Reference Books:

1. Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, J. Banks, 1998, John Wiley & Sons.

2. Parallel and Distributed Simulation Systems, Fujimoto R.M., 2000, John Wiley & Sons.



Course Outcomes:

The student will be able to –

1. Demonstrate the effectiveness of modeling and simulation at predicting behavior/performance/problems of systems under development.
2. Develop a model for a given problem using appropriate modeling and simulation technique/formalism.
3. Implement discrete event simulation models using general-purpose programming languages or DES frameworks
4. Design an agent-based simulation model for a complex system.
5. Contribute towards increased utilization of modeling and simulation as a problem solving approach for issues in governance and industry where it could be applied
6. Adapt to the changing needs of the organizations and individuals during the development process.

CS4008: INTERNET OF THINGS

Credits: 04 **Teaching Scheme: Theory: 3 Hours / Week**
Project Based Lab: 2 Hours/Week

Section 1: Topics/Contents

Introduction & Application to Internet of Things: The Internet of Things, Importance of IoT, Towards the IoT Universe IoT Strategic Researchnd Innovation Directions, IoT Applications, Future Internet Technologies, Infrastructure, Networks and Communication, Processes, Data Management, Iot Smart X Application: Smart Cities, Smart Energy & Smart Grid, Smart Mobility & transport, Smart Home, Smart Building & Infrastructure, Smart Factory & Manufacturing, Smart Health, Smart Logistics & Retails.

Embedded Suite for IoT: Introduction to ARM Processor, introduction to ARM Processor, ARM Processor Architecture, Programmer's model, Modes of operation, Interrupt Structure and Applications. Management of Power Supply, Introduction to Raspberry Pi, Understanding the Raspberry Pi board and its components, recognizing the Input/output, GPIO connectivity.

WIRELESS TECHNOLOGIES supporting IoT: Protocol Standardization for IoT, Machine to machine (M2M) and WSN Protocols, Basics of RFID, RFID Protocols & NFC protocols, Issues with IoT Standardization, Unified Data Standards, Protocols – IEEE 802.15.4, ZigBee, IPv6 technologies for the IoT.

Section2:Topics/Contents

IoT Networking: Networking Architectures: Star, Mesh, Tree, and Overview of networking Protocols: TCP/IP, 6LowPan, RPL: Routing protocol for lossy& low power network. IoT Devices Application Level Protocols: Introduction to MQTT, Quality of Service parameter in MQTT, CoAP, XMPP

IoT:PRIVACY, SECURITY& GOVERNANCE: Overview of Governance, Privacy and Security Issues, Contribution from FP7 Projects, Security, Privacy and Trust in IoT-Data-Platforms for Smart Cities, First Steps towards a Secure Platform, Smart Approach. Data Aggregation for the IoT in Smart Cities, Security,

Connecting IoT to Cloud: Introduction to cloud computing, Difference between Cloud Computing and IoT, Fog Computing: The Next Evolution of Cloud Computing, Role of Cloud Computing in IoT. Livingon the Edge, Connecting IoT to cloud, Cloud Storage for IoT. Cloud-to-Device Connectivity,Challenge in integration of IoT with Cloud.

List of Project areas:

1. Being Social on Twitter & update status on Twitter through Arduino.
2. IoTTemperature monitor for green house.
3. Arduino Based Home Security System.
4. Arduino Based Heart Rate Monitor:
5. A fall detection system based on Arduino, Windows and Azure
6. Voice Controlled Mini Home Automation using Android Smartphone.
7. Control Devices using Localhost Web Server for Home Automation.
8. Minimizing Electricity Theft by Internet of Things.
9. Internet-of-Things Based Ubiquitous Healthcare Systems.
10. A Design of the IOT Gateway for Agricultural Greenhouse.

Text Books:

1. Jan Holler, VlasiosTsiatsis, Catherine Mulligan, Stefan Avesand, StamatisKarnouskos, David Boyle, "From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence", 1st Edition, Academic Press, 2014.

2. Francis daCosta, "Rethinking the Internet of Things: A Scalable Approach to Connecting Everything", 1st Edition, Apress Publications, 2013

Reference Books:

1. Marco Schwartz, "Internet of Things with the Arduino Yun", Packt Publishing, 2014. 2. Daniel Minoli John Wiley &Sons ,Building the internet of things with ipv6 and mipv6, The Evolving World of M2M Communications, ISBN: 978-1-118-47347-4

3. Cassimally, Hakim, "Designing the Internet of Things", Wiley Publications, ISBN 10: 111843062X

Course Outcomes:

The student will be able to

1. Learn the terminology, technology and its applications of IoT
2. Analyze Embedded suite widely used in IoT.
3 . Describe the concept of M2M with necessary protocols
4. Uunderstand the cloud storage for IoT applications.
5. Optimize resources for different IoT applications
6. Understand Real world IoT Design constraint.

CS4009: IMAGE PROCESSING

Credits: 04 **Teaching Scheme: Theory: 3 Hours / Week**
Lab: 2 Hours/Week

Section 1: Topics/Contents

Introduction, Elements of image processing system, Scenes and Images, Vector Algebra, Human Visual System, color vision color model: RGB, HVS, YUV, CMYK, YC_bC_r and some basic relationships between pixels, linear and nonlinear operations. Image types (optical and microwave), Image file formats (BMP, tiff, jpeg, ico, ceos, GIF,png, raster image format). Image sampling and quantization.

Thresholding, Spatial domain techniques { Image Negative, Contrast stretching, gray level slicing, bit plane slicing, histogram and histogram equalization, local enhancement technique, image subtraction and image average, Image Smoothening: low-pass spatial filters, median filtering, Image Sharpening: high-pass spatial filter, derivative filters, Frequency domain techniques- Ideal low-pass filter, Butterworth low-pass filter, High-pass filter, Homo-morphic filters.

Image segmentation- Classification of image segmentation techniques: Watershed Segmentation, Edge-based Segmentation, region approach, clustering techniques, edge-based, classification of edges and edge detection, watershed transformation Feature Extraction- Boundary representation(Chain code, B-spline representation, Fourier descriptor) Region representation (Area, Euler number, Eccentricity, Shape matrix, moment based descriptor), texture based features.

Section2: Topics/Contents

Introduction to Image compression and its need, Coding redundancy, classification of compression techniques (Lossy and lossless- JPEG, RLE, Huffman, Shannon fano), scalar and vector quantization Object Recognition { Need, Automated object recognition system, pattern and pattern class, relationship between image processing and object recognition, approaches to object recognition.

Introduction to two dimensional orthogonal and unitary transforms, properties of unitary transforms. One-two dimensional discrete Fourier Transform (DFT). Cosine, Slant, KL, affine transforms. Singular Value Decomposition, Applications of transforms in Image processing.

Sub band coding, Haar Transform – it's application as a Wavelet, multi resolution expansions, Wavelet Transform in one dimensions; Wavelet transforms in two dimensions.DB4, Fast Wavelet Transform, Other Applications of Wavelet in image processing

List of Assignments: (THL course)

1. Write matlab code to display following binary images Square Triangle Circle
   Write matlab code to perform following operations on images
   Flip Image along horizontal and vertical direction.
   Enhance quality of a given image by changing brightness of image.
   Image negation operation.
   Change contrast of a given Image.
2. Write Matlab code to implement pseudo colouring operation of a given image.
   Write Matlab Code for Pseudo Colour of Image by using Gray to colour transform.
3. Study of different file formats e.g. BMP, TIFF and extraction of attributes of BMP.
4. Write matlab code to find following statistical properties of an image.
   Mean
   Median
   Variance
   Standard deviation
   Covariance.
5. Write matlab code to enhance image quality by using following techniques
   Logarithmic transformation
   Histogram Equalization
   Gray level slicing with and without background.
   Inverse transformation.
6. Read an Image and Perform singular value decomposition. Retain only k largest singular values and reconstruct the image. Also Compute the Compression ratio.
7. Write matlab code to enhance image quality by using following techniques
   Low pass and weighted low pass filter.
   Median filter. Laplacian mask.
8. Write matlab code for edge detection using Sobel, Prewitt and Roberts operators.
9. Write C-language code to find out Huffman code for the following word COMMITTEE.
10. Write matlab code to design encoder and decoder by using Arithmetic coding for the following word MUMMY. (Probabilities of symbols M-0.4, U-0.2, X-0.3,Y-0.1).
11. Write matlab code to find out Fourier spectrum, phase angle and power spectrum of binary image and gray scale image.

Text Books:

1. *Rafael Gonzalez & Richard Woods, "Digital Image Processing," 3rd Edition, Pearson publications, ISBN 0132345633.*
2. *Anil K. Jain, "Fundamental of Digital Image Processing," 5th Edition, PHI publication, ISBN 13: 9780133361650.*

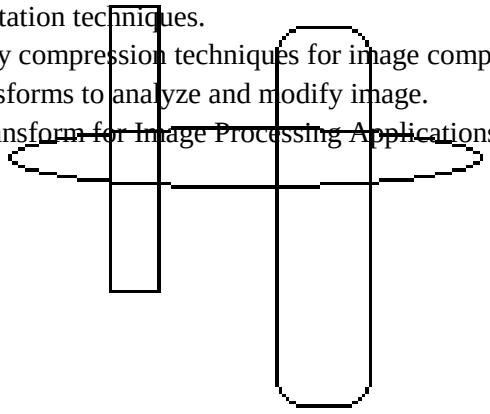
Reference Books:

1. *Pratt, "Digital Image Processing," Wiley Publication, 3rd Edition* ***, ISBN*** *0-471-37407-5.*
2. *K.R. Castleman, "Digital Image Processing," 3rd Edition, Prentice Hall: Upper Saddle River, NJ, 3, ISBN 0-13-211467 -4.*
3. *K. D. Soman and K. I. Ramchandran, "Insight into wavelets - From theory to practice," 2nd Edition PHI, 2005.*

Course Outcomes:

The student will be able to

1. Describe image model.
2. Perform spatial filtering on image.
3. Identify Image Segmentation techniques.
4. Apply lossless and lossy compression techniques for image compression.
5. Use various image transforms to analyze and modify image.
6. Understand Wavelet transform for Image Processing Applications.



CS 4010: MODEL-BASED SYSTEMS ENGINEERING

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
Lab 2 Hours / Week

Section 1: Topics/Contents

System Specifications: Static Behavior specification using Use Case diagrams, Dynamic Behavior specification using Sequence diagrams, Collaboration diagrams, State Chart diagrams, Activity Diagrams, Communication Diagram, Interaction Overview Diagrams, Timing Diagrams, Metamodel discussion of behavioral diagrams, System Design Specification with Class diagrams, Component, Deployment, Package, Profile diagrams, Interface and architectural specifications, Metamodel discussion of Structural diagrams, CMOF-EMOF Infrastructure Specifications, OMG's profiles for object-oriented and service-oriented technologies

MDA and Enterprise Computing: Challenges Facing the Software Industry, Machine-Centric Computing, Application-Centric Computing, Enterprise-Centric Computing, Pressures on Enterprise-Centric Computing, Pressure on Production Costs, Bringing Model-Centrism to Intermediate Tiers, EAI, and B2Bi, Syntactic Abstraction versus Semantic Abstraction, B2Bi and MDA, Flexibility in Choosing the Abstraction Level, EAI and MDA, The Limits of Declarative Specification, Metadata Integration, MDA and Component-Based Development, Model Driven Enterprise Architecture, Standardized MDA-Based Modeling Languages, Synchronizing among Multiple Tiers, Middleware and the Abstraction Gap, Design by Contract, The Role of UML in MDA, The Meta Object Facility (MOF), Extending and Creating Modeling Languages, Building Compilation of Class Models, Abstraction Levels, Modeling Transformations with CWM, Sample Transaction Metamodel

Introduction to Graph Transformation Systems: A Short Introduction to Category Theory, A Short Introduction to Signatures and Algebras, General Overview of Graph Grammars and Graph, The Main Ideas of the Algebraic Graph Transformation, Approach, Graphs, Typed Graphs, and the Gluing Construction, Graph Transformation Systems, Adhesive High-Level Replacement Categories, Adhesive High-Level Replacement Systems, Embedding and Local Confluence, Constraints and Application Conditions

Section2: Topics/Contents

Systems Engineering Overview, Introduction to model-based systems engineering, MBSE concepts, Process modeling with MBSE, Expanded Process modeling, Introduction to SysML and systems modeling, The SysML Basic Feature Set, Viewing SysML Models with Diagrams, Organizing The Model With Packages, Modeling Structure With Blocks, Modeling Constraints with Parametrics, Modeling Flow-Based Behavior with Activities. Modeling Message-Based Behavior with Interactions, Modeling Text-Based Requirements, Modeling Cross-Cutting Relationships with Allocations, Customizing SysML for Specific Domains

Managing the Quality of UML Models: Quality in Model Driven Engineering, Examples and Evidences, Integrating Quality Criteria and Methods of Evaluation for Software Models,

Evaluating Performance of Software Architecture Models with the Palladio Component Model, Integrating Measures and Redesigns in the Definition of Domain Specific Visual Languages, Measuring Models, Model-Driven Software Refactoring, A Pattern Approach to Increasing the Maturity Level of Class Models, Transitioning from Code-Centric to Model-Driven Industrial Projects: Empirical Studies in Industry and Academia, From Requirements to Java Code: An Architecture-Centric Approach for Producing Quality Systems, Quality-Driven Model Transformations: From Requirements to UML Class Diagrams, A Framework for Understanding and Addressing the Semiotic Quality of Use Case Models, Quality-Aware Model-Driven Service Engineering, Model-Driven Integration in Complex Information Systems: Experiences from Two Scenarios

Models of Reactive Systems: Communication, Concurrency, and Causality, Model-Based Integration, Metamodelling: State of the Art and Research Challenges, Semantics of UML Models for Dynamic Behavior, A Survey of Different Approaches, Modeling Languages for Real-Time and Embedded Systems, Requirements and Standards-Based Solutions, Requirements Modeling for Embedded Real-time Systems, UML for Software Safety and Certification: Model-Based Development of Safety-Critical Software-Intensive Systems, Model Evolution and Management, Model-Based Analysis and Development of Dependable, Tools and Languages for Real-time Systems

---

List of Practical's: (For THL, TLP courses)

1. To narrate System Requirements Specification Document for target system with reference to the IEEE 610.12.1990 Std guidelines.
2. To classify and structure functional and non-functional requirements into broad subject areas. The requirements structures shall be indicated by Requirement Structure diagrams using SysMLprofiles for requirement management.
3. To decompose and organize the problem domain area into broad subject areas and identify the boundaries of problem/systemusing Use Case diagrams. The target system representation shall be specified using Use Case Metamodel and OCL.
4. To depict the dynamic behavior of the target system using sequence diagram. The Sequence diagram should be based on the Scenarios generated by the inter-object Communication. The target system representation shall be specified using Interaction diagram Metamodel and OCL.
5. To depict the state transition with the life history of objects of a given class model. The model should depict:
    a. Possible ways the object can respond to events from other objects.
    b. Determine of start, end, and transition states.
    c. The target system representation shall be specified using State diagram Metamodel and OCL.
6. To depict the dynamic behavior using detailed Activity diagram. Activity is a parameterized behavior represented as coordinated flow of actions. The target system representation shall be specified using Activity diagram Metamodel and OCL.
7. To develop logical static structure of target system with Software Class diagram. To prepare Class Collaboration-Responsibility (CRC) cards for the Conceptual classes traced from System analysis phase. The target system representation shall be specified using Class diagram Metamodel and OCL.

8. To represent physical module that provides occurrence of classes or other logical elements identified during analysis and design of system using Component diagram. The model should depict allocation of classes to modules. The component specifications shall be narrated using precise Program Design Language constructs separating computation from interface.
9. To represent deployment view of the system through Architecture Diagram encompassing both hardware and software profiles. The target system representation shall be specified using Component and Deployment Metamodel.

Text Books:

1. *David S. Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing, 2003, Wiley Publishing, Inc, ISBN: 0-471-31920-1*
2. *Prof. Dr. HartmutEhrig, Dr. KarstenEhrig, Ulrike Prange, Dr. Gabriele Taentzer, Fundamentals of Algebraic Graph Transformation, Springer-Verlag Berlin Heidelberg 2006, ISBN-13 978-3-540-31187-4*

***Reference Books* :**

1. *Holger Giese, Gabor Karsai, Edward Lee, Bernhard Rumpe, Bernhard Schätz, Model-Based Engineering of Embedded Real-Time Systems, Springer-Verlag Berlin Heidelberg 2010, ISBN-13 978-3-642-16276-3*
2. *JörgRech, Christian Bunse, Model-Driven Software Development: Integrating Quality Assurance, 2009 IGI Global, ISBN 978-1-60566-007-3*
3. *Jon Holt, Simon Perry, SysML for Systems Engineering 2nd Edition: A model-based approach, The Institution of Engineering and Technology 2008, ISBN 978-1-84919-651-2*
4. *Sanford Friedenthal, Alan Moore, Rick Steiner, A Practical Guide to SysML: The Systems Modeling Language, Third edition, Elsevier Inc, ISBN: 978-0-12-800202-5*
5. *Sami Beydeda, Matthias Book, Volker Gruhn, Model-Driven Software Development, Springer-Verlag Berlin Heidelberg 2005, ISBN-13 978-3-540-25613-7*
6. *Tom Pender, "UML Bible", John Wiley & sons, ISBN – 0764526049*
7. *Jim Arlow, IlaNeustadt, "UML 2 and Unified Process: Practical Object Oriented Analysis and Design.", 2nd Edition, Addison-Wesley, ISBN – 0321321278.*

Course Outcomes:

Upon completion of the course, graduates will be able to -

1. Model requirements using SysML specifications and profiles encompassing requirement structures and catalogue.
2. Discover the set of behavioral and structural properties of complex systems by making use of UML Superstructure specifications and OCL.
3. Explore the concepts, guidelines and technology for System Designelements orchestration to integrate in Enterprise Component Systems in societal context.
4. Prepare well-formed specifications and reports for component service composition and delivery to the stakeholders as being a part of development team.
5. Understand case studies and lessons learned with utilization of Model-based development concepts and specification knowledge towards planning and implementing complex systems.
6. Create sustainable System design supported by reuse, documentation, and testability.

CS 4012: GENETIC ALGORITHMS

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

A GENTLE INTRODUCTION TO GENETIC ALGORITHMS : What Are Genetic Algorithms? ,Robustness of Traditional Optimization and Search Methods. The Goals of Optimization, How Are Genetic Algorithms Different from Traditional Methods?
A Simple Genetic Algorithm. Genetic Algorithms at Work—a Simulation by hand. Grist for the Search Mill—Important Similarities 18 Similarity Templates (Schemata) 19 Learning the Lingo.
GENETIC ALGORITHMS REVISITED: MATHEMATICAL FOUNDATIONS: Who Shall Live and Who Shall Die? The Fundamental Theorem, Schema Processing at Work: An Example by Hand Revisited. The Two-armed and й-armed Bandit Problem, How Many Schemata Are Processed Usefully. The Building Block Hypothesis, Another Perspective: The Minimal Deceptive Problem, Schemata Revisited: Similarity Templates as Hyper planes.
COMPUTER IMPLEMENTATION OF A GENETIC ALGORITHM: Data Structures, Reproduction, Crossover, and Mutation, A Time to Reproduce, a Time to Cross, Get with the Main Program, How Well Does it Work?, Mapping Objective Functions to Fitness Form, Fitness Scaling, Coding, A Multiparameter, Mapped, Fixed-Point Coding, Discretization , Constraints.

Section2: Topics/Contents

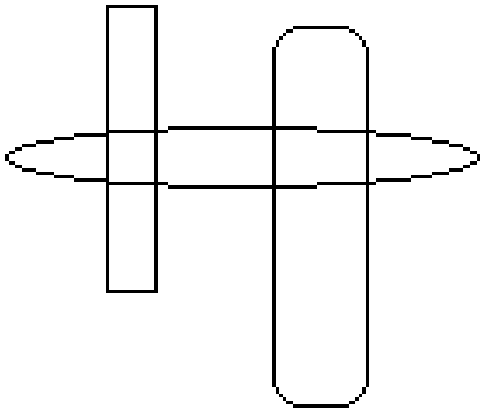
SOME APPLICATIONS OF GENETIC ALGORITHMS: The Rise of Genetic Algorithms , Genetic Algorithm Applications of Historical Interest , De Jong and Function Optimization, Improvements in Basic Technique , Current Applications of Genetic Algorithms.
ADVANCED OPERATORS AND TECHNIQUES IN GENETIC SEARCH : Dominance, Diploidy, and Abeyance . Inversion and Other Reordering Operators .Other Micro-operators, Niche and Speciation, Multiobjective Optimization, Knowledge-Based Techniques, Genetic Algorithms and Parallel Processors .
INTRODUCTION TO GENETICS-BASED MACHINE LEARNING : Genetics-Based Machine Learning: Whence It Came 218 What is a Classifier System? Rule and Message System. Apportionment of Credit: The Bucket Brigade , Genetic Algorithm . A Simple Classifier System in Pascal, Results Using the Simple Classifier System .
APPLICATIONS OF GENETICS-BASED MACHINE LEARNING: The Rise of GBML, Development of CS-1, the First Classifier System , Smith's Poker Player , Other Early GBML Efforts , A Potpourri of Current Applications .

List of Practical's: (For THL, TLP courses)

Text Books:

Reference Books:

Course Outcomes:
Upon completion of the course, graduates will be able to -



CS 4014: SOFTWARE TESTING AND QUALITY ASSURANCE

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
: Lab 2 Hours / Week

Section 1: Topics/Contents

Software Measurement: Measurement Theory, Software Measurement and Models, Measurement Scales, Classification of Software Measures, Measurement Framework, Theory of Program Testing, Graph Theory for Testers, Software Complexity, Measuring Internal Product Attributes: Size, Measuring Internal Product Attributes : Structure, Halstead's Software Science, Product Quality Metrics, In-Process Quality Metrics, Software Reliability: Measurement and Prediction, The Rayleigh Model, SRE process, Reliability Concepts: Hardware and Software, Deploying SRE

Principles of Testing: Testing Concepts: Purpose of Software Testing, Testing Principles, Goals of Testing, Testing aspects: Requirements, Test Scenarios, Test cases, Test scripts/procedures, Strategies for Software Testing, Testing Activities, Mistakes, Faults & Failures, Planning for Verification and Validation, Software Inspections, Automated Static Analysis, Verification and Formal Methods, Levels of Testing

White-Box Testing: Test Adequacy Criteria, Static Testing, Structural Testing, Code Complexity Testing, Mutation Testing, Data Flow Testing

Black-Box Testing: Test Case Design Criteria, Requirement Based Testing, Positive and Negative Testing, Boundary Value Analysis, Equivalence Partitioning State Based Testing, Domain Testing

Functional Testing: Test Plan, Test Management, Test Execution and Reporting, Test Specialist Skills, Tester's Workbench and Tool Categories, Test Maturity Model and Test Process Assessment, Debugging & Root Cause Analysis, Software Items, Component & Units, Test Bed, Traceability and Testability, Attributes of Testable Requirements, Test Matrix, Types of Testing Documentation, Verification Testing, Validation Testing, Integration Testing, System and Acceptance Testing, GUI Testing,Regression Testing, Selection, Minimization and Prioritization of Test Cases for Regression Testing, Creating Test Cases from Requirements and Use cases, Software Defects: Origins of Defects, Defect Classes, Defect Repository / Test Design, Defect Repository

Section2: Topics/Contents

Higher Order Testing: Object Oriented Testing, Specification Based Testing, Performance Testing, Ad-hoc Testing, Usability and Accessibility Testing, Risk-based Testing, Exploratory Testing, Scenario-based Testing, Random Testing Compatibility Testing, User Documentation Testing, Client–Server System Testing, RAD Testing, Configuration Testing, Testing internal Controls, Multiplatform Environment Testing, Security Testing, Web-based System Testing, Reliability Testing, Efficiency Testing, Maintainability Testing, Portability Testing, Introduction to Performance Testing, Application Performance Testing, Process of Performance Testing, Effective Root-Cause analysis, Testing VS Test Automation, Tool evaluation and selection, Automation team roles, Architectures, Planning and implementing test automation process

Introduction to Software Quality Assurance: The software quality challenge, Meaning of software quality, Software quality factors , Software Quality Lessons Learned, The components of the software quality assurance system, Pre-project software quality components: Contract Review, Development and quality plans, SQA components in the project life cycle: Integrating quality activities in the project life cycle, Assuring the quality of software maintenance components, Assuring the quality of external participants' contributions, CASE tools, Software quality infrastructure components, Pareto Principles, Total Quality Management, Ishikawa's Seven Basic Tools

Software Quality Assurance Management: Management components of software quality: Project progress control, Software quality metrics, Costs of software quality, Standards, certification and assessment: Quality management standards, SQA project process standards – IEEE software engineering standards, Management and its role in software quality assurance, The SQA unit and other actors in the SQA system, Inspection as an Up-Front Quality Technique, Software Audit Methods, Software Safety and Its Relation to Software Quality Assurance, SQA for Small Projects, Development Quality Assurance, Quality Management in IT, Introduction to ITIL, Software Quality Assurance Metrics, Software Benchmarks and Baselines

List of Practicals: (For THL, TLP courses)

1. To Prepare Test Plan for the implemented system under test. The Test Plan shall be based on System Requirement Specification. The Test plan consists of following issues.
    a. Purpose of the test. /Location and schedule of the test.
    b. Test descriptions. /Pass and Fail Criteria.
2. To identify and narrate Test cases, Test scripts/procedures and Test incident Report identifier for the system under test. Refer Use case analysis document to prepare mentioned/ identified test documents. The expected specifications/ behaviors can be stated with the help of Test Oracle.
3. To perform Unit testing especially indicating the traced Independent data paths, Control paths and Error handling paths. Prepare control flow graphs for the unit under test. Compute the Cyclomatic complexity of the unit.
4. To perform Data Flow testing for the Program Segments by identifying the Definition-Use chain and type of data flow anomaly.
5. To perform Mutation Analysis of the Program Segments along with mutant history, mutation score and type of mutation by using any Code analysis Tool / Mutation Testing Tool (JUNIT, MuJava).
6. To perform Black-Box Testing for all the units contained in the architectural segments using Equivalence Partitioning, Boundary Value Analysis and Orthogonal Array testing methods. To study exploratory Testing for the Module under Test and merits/demerits of this technique.
7. To perform Regression Testing / GUI Testing of the System under construction with Unit and Integration profiles by using any Functional Testing Tool.
8. To perform Automated Testing using suitable CASE tool addressing Higher-Order testing strategies.
9. To perform Web Based Testing for Web Application incorporating any Open Source Tool. To study Performance Testing, Load Testing, Security Testing, Stress Testing, Demonstrate on link Test expectation.

10. To perform Software Audit (Checklist and Template-based) for the software developed and improve the Code Quality.

Text Books:

1. *Burnstein, "Practical Software Testing", Springer International Edition, ISBN 81-8128-089-X*
2. *William E. Perry, " Effective Methods for Software Testing", John Wiley and Sons, ISBN 9971-51-345-5*
3. *Daniel Galin, Software Quality Assurance: From theory to implementation, Pearson Education Limited, 2004, ISBN 0201 70945 7*

Reference Books:

1. *KshirasagarNaik, PriyadarshiTripathy, Software Testing and Quality Assurance-Theory and Practice, John Wiley & Sons, Inc., 2008, ISBN 978-0-471-78911-6*
2. *Fenton, Pfleeger, "Software Metrics: A Rigourous and practical Approach", Thomson Brooks/Cole, ISBN 981-240-385-X.*
3. *Desikan, Ramesh, "Software Testing: principles and Practices", Pearson Education, ISBN 81-7758-121-X.*
4. *Anne MetteJonassen Hass, Guide to Advanced Software Testing, ARTECH HOUSE, INC., 2008, ISBN-13: 978-1-59693-285-2*
5. *Ian Molyneaux,The Art of Application Performance Testing, O'Reilly Media, Inc., 2009, ISBN: 978-0-596-52066-3*
6. *Jamie L. Mitchell, Rex Black, Advanced Software Testing—Vol. 3, 2nd Edition, Rocky Nook, 2015, ISBN: 978-1-937538-64-4*
7. *G. Gordon Schulmeyer, Handbook of Software Quality Assurance Fourth Edition, ARTECH HOUSE, INC., 2008, ISBN-13: 978-1-59693-186-2*

Course Outcomes:

Upon completion of the course, graduates will be able to –

1. Select and classify measurement scales and models, software metrics and measures addressing software quality and reliability.
2. Conduct unit and integration tests by determining test design, test automation, test coverage criteria using testing frameworks and test adequacy assessment using control flow, data flow, and program mutations.
3. Apply suitable higher order testing techniques and methods in order to achieve verified and validated software by following testing best practices.
4. Demonstrate the skillset as a tester to neutralize the consequences of wicked problems by narrating effective test cases and test procedures.
5. Adapt to various test processes, types of errors and fault models and methods of test generation from requirements for continuous quality improvement of the software system along with Software Quality best practices usage.
6. Apply software testing cycle in relation to software development and project management focusing incidents and risks management within a project towards efficient delivery of software solutions and implement improvements in the software development processes by making use of standards and baselines.

CS 4015: SOFTWARE ARCHIECTURE AND DESIGN

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
: Lab 2 Hours / Week

Section 1: Topics/Contents

Introduction to Software Architecture: Architecture Orientation Framework: Motivation, Overview of the Framework, Architectures and Architecture Disciplines (What), Architecture Perspectives (Where), Architecture Requirements (Why), Architecture Means (With What), Organizations and Individuals (Who), Architecture Method (How), Architectures and Architecture Disciplines (What): Classic Architecture as Starting Point, From Classic Architecture to Software Architecture, Architecture and the System Concept, Architecture and the Building Blocks of a System, Architecture Perspectives (Where): Architecture Levels, Organizational Level, System Level, Building Block Level, Architecture Views, Architecture Requirements (Why): Requirements Characteristics and Types, Organizational Requirements, System Requirements, Building Block Requirements, Qualities and Constraints, Requirements in the Context of Architecture, Architecture Means (With What): Architecture Principles, Basic Architecture Concepts, Architecture Tactics, Styles, and Patterns, Organizations and Individuals (Who), Architecture Method (How)

Software Architecture Styles: Three Categories of Styles, Style Guides: A Standard Organization for Explaining a Style, Choosing Which Element and Relation Properties to Document, Notations for Architecture Views, Module Views, Module Styles: Decomposition Style, Uses Style, Generalization Style, Layered Style, Aspects Style, Data Model
Component-and-Connector Views: Overview, Elements, Relations, and Properties of C&C Views, C&C Views applicability, Perspectives: Choosing Connector Abstractions, Notations for C&C Views, Relation to Other Kinds of Views, Component-and-Connector Styles: Data Flow Styles, Pipe-and-Filter Style, Call-Return Styles, Client-Server Style, Peer-to-Peer Style, Service-Oriented Architecture Style, Event-Based Styles, Publish-Subscribe Style, Repository Styles, Shared-Data Style, Crosscutting Issues for C&C Styles, Allocation Views and Allocation Styles: Deployment Style, Install Style, Work Assignment Style, Other Allocation Styles

Architectural Decisions: Refinement, Descriptive Completeness, Documenting Context Diagrams, Documenting Variation Points, Documenting Architectural Decisions, Combining Views, Documenting Software Interfaces: Interface Documentation, Showing the Existence of Interfaces in Diagrams, A Standard Organization for Interface Documentation, Coming to Terms: Error Handling, Stakeholders of Interface Documentation, Conveying Syntactic Information, Conveying Semantic Information, Coming to Terms: Signature, Interface, API, Architectural Documentation Relevance: Documenting Behavior, Building the Architecture Documentation, Building the Documentation Package, Reviewing an Architecture Document

Section2: Topics/Contents

Collaborative Architecture Determination: The Impact of People and Leadership on Scalability, Roles for the Scalable Technology Organization, Designing Organizations, Leadership 101, Management 101, Making the Business Case, Understanding Why Processes Are Critical to Scale, Managing Incidents and Problems, Managing Crisis and Escalations, Controlling Change in Production Environments, Exploring Architectural Principles, Joint Architecture Design, Architecture Review Board

Architecting Scalable Solutions: Focus on Core Competencies: Build Versus Buy, Performance and Stress Testing, Barrier Conditions and Rollback, Designing for Any Technology, Creating Fault Isolative Architectural Structures, Introduction to the AKF Scale Cube, Splitting Applications for Scale, Splitting Databases for Scale, Caching for Performance and Scale, Asynchronous Design for Scale, Large Scale Data, Clouds and Grids, Soaring in the Clouds, Plugging in the Grid, Monitoring Applications, Planning Data Centers, Introduction to Middleware Technologies, Reaching for the Clouds, Defining the Clouds for the Enterprise, Making the Business Case for Clouds, Working from Data, Services, Processes to the Clouds

Software Architecture Practices: SOA and Channels, Service-Oriented Architecture Enterprise Architecture Framework and Methodology, Incorporating Existing Enterprise Architecture Documents and Artifacts into the SOA~EAF, Dealing with Purchased or Leased Business Applications, Transforming Governance to Support SOA, SOA System Development Life Cycle, Capacity Planning under SOA, People Involved in the SOA Process, Implementing an Effective SOA Strategy under a Decentralized Business or IT Model, SOA Business Strategy and Roadmap

List of Practical's: (For THL, TLP courses)

1. To narrate Requirement Definition Document for the target system with following three areas: Problem Identification, Problem Definition, and Problem Statement
2. To narrate System Requirements Specification Document for target system with reference to the IEEE 610.12.1990 Std guidelines.
3. To narrate System Architecture Requirement Specification Document for target system with stakeholder and roles description.
4. To select appropriate Architectural View and Style and prepare Architecture Diagram for the target system.
5. To prepare Architecture Decision document describing Architectural Decisions, Software Interfaces, and behaviors along with Architectural Review.
6. To implement the target system using the Technical Architecture conforming to technology availability and scalability.
7. To create Test Plan, Test Cases and apply them to test the performance adequacy of the system implemented.

Text Books:

1. *Paul Clements, Felix Bachmann, Len Bass, David Garlan, Documenting Software Architectures: Views and Beyond Addison-Wesley Professional 2003, ISBN-10:0201703726, ISBN-13: 9780201703726*
2. *Martin L. Abbott, Michael T. Fisher, The Art of Scalability, Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise, Pearson Education, Inc., 2010, ISBN-13: 978-0-13-703042-2*

Reference Books:

1. *Rick Sweeney, Achieving Service-Oriented Architecture, Applying an Enterprise Architecture Approach, John Wiley Publications, 2010, ISBN 978-0-470-60451-9*
2. *David S. Linthicum, Cloud Computing and SOA Convergence in Your Enterprise, 2010 Pearson Education, Inc., ISBN-13: 978-0-13-600922-1*
3. *Arno Puder, Kay Römer, Frank Pilhofer, Distributed Systems Architecture, A Middleware Approach, 2006 by Elsevier Inc., ISBN 13: 978-1-55860-648-7*
4. *ParthaKuchana, Software Architecture Design Patterns in Java, 2004 by CRC Press LLC, ISBN 0-8493-2142-5*
5. *C.E. Dickerson, D.N. Mavris, Architecture And Principles Of Systems Engineering, 2010 by Taylor and Francis Group, LLC, ISBN 978-1-4200-7253-2*

Course Outcomes:

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

1. Examine and breakdown real-world problem scenarios into structured partitions depicting static and dynamic behavior of the system using Software Architecture Requirements Capture practices.
2. Identify and formulate software requirements and behavioral models using Architectural Assessment of behavioral views by selecting appropriate architectural views.
3. Compose system design specifications indicating logical, physical, deployment, and concurrency viewpoints using software interface explorations, error handling, and architectural reviews.
4. Construct and justify the evolutionary system description models expressing high-level technical architecture accommodating applicable architectural styles compatible to requirements using CASE tools.
5. Comprehend the nature of architectural styles implementation by understanding numerous examples from different technology categories and publish guidance, applicability, reasonableness, and relation to other design criteria resulting in well-documented system profiles to the engineering and social community.
6. Propose multi-faceted defendable solutions demonstrating team-skills accommodating architectural views and styles reducing the potential cost and performance impedance in order to realize system artifacts with the help of Architecture Derivation and Development practices.

CS4017: MOBILE COMPUTING

Credits: 4 **Teaching Scheme: Theory: 3 Hours / Week**
Lab: 2 Hours / Week

Section 1: Topics/Content

Introduction to Cellular Networks: Personal Communication System (PCS), PCS Architecture, Cell phone generation-1G to 5G Mobile Station (UE), SIM, Base Station (enodeB), Base Station Controller, Mobile Switching Center, MSC Gateways, HLR and VLR, AuC/EIR/OSS, Radio Spectrum, Free Space Path Loss, S/N Ratio, Line of sight transmission, Length of Antenna, Fading in Mobile Environment. **Cellular Network Design:** Performance Criterion, Frequency Reuse, Co-channel Interference and System Capacity, Channel Planning, Cell Splitting, Mobility Management in GSM and CDMA. **Medium Access Control:** Specialized MAC, SDMA, FDMA, TDMA, CDMA, Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS), GMSK Modulation, 8PSK, 64 QAM, 128 QAM and OFDM

Section2: Topics/Content

GSM and CDMA : D-AMPS, GSM – Architecture, GSM Identifiers, Spectrum allocation, Physical and Logical Traffic and Control channels, GSM Bursts, GSM Frame, GSM Speech Encoding and decoding, Location Update, Incoming and Outgoing Call setup GPRS, EDGE, cdmaOne (IS-95: CDMA95), **3G and 4G Technologies for GSM and CDMA:**, W-CDMA, UMTS, HSPA (High Speed Packet Access), HSDPA, HSUPA, HSPA+, TD-SCDMA, LTE (E-UTRA) 3GPP2 family CDMA2000 1x, 1xRTT, EV-DO (Evolution-Data Optimized), Long Term Evolution (LTE) in 4G

List of Practical's: (For THP)

Operating System recommended: - 64-bit Open source Linux or its derivative

Programming tools recommended: - Android Studio 3.x, JAVA

Project 1: Design a Mobile App for (Learning How To Develop App)

Phone Call.
Media Player
Simple Calculator
Sms Manager
Google Map To Trace The Location Of Device
Frame Animation

Project 2. : Design mobile app to perform the task of creating the splash screen for the application using timer, camera options and integrate google map api on the first page of the application. Make sure map has following features:

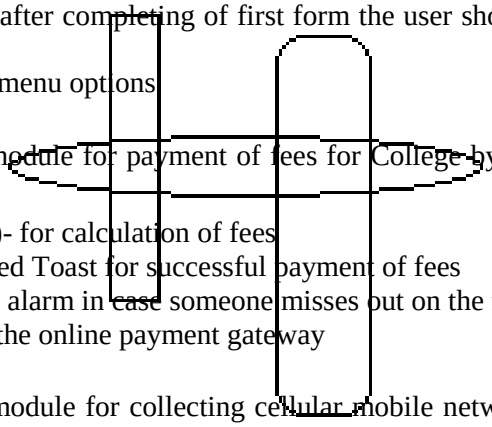
i) Zoom & View change
ii) Navigation to specific locations
iii) Marker & getting location with touch
iv) Monitoring of location

Project 3. : Create an app to add of a product to SQLite database and make sure to add following features

i) SMS messaging and email provision
ii) Bluetooth options
iii) Accessing Web services
iv) Asynchronous remote method call
v) Use Alert box for user notification

Project 4. : Create an application for Bank using spinner, intent

a) Form 1: Create a new account for customer
b) Form 2: Deposit money in customer account.
c) Link both forms, after completing of first form the user should be directed to the second form
d) Provide different menu options



Project 5. : Create the module for payment of fees for College by demonstrating the following methods.

i) FeesMethod()- for calculation of fees
ii) Use customized Toast for successful payment of fees
iii) Implement an alarm in case someone misses out on the fee submission deadline
iv) Demonstrate the online payment gateway

Project 5. : Create the module for collecting cellular mobile network performance parameters using telephony API Manager

i) Nearest Base Station
ii) Signal Strengths
iii) SIM Module Details
iv) Mobility Management Information

Text Books:

1. *Jochen Schiller, "Mobile Communications ",Pearson Education,Second Edition, 2004, ISBN: 13: 978-8131724262*
2. *Smith Collins, "3G Wireless Networks" McGraw Hill Communications, Second Edition , Indian Print, 2016, ISBN-13 978-0-07-063692-7*
3. *Martin Sauter, "3G, 4G and Beyond: Bringing Networks, Devices and the Web Together", 2012, ISBN-13: 978-1118341483*

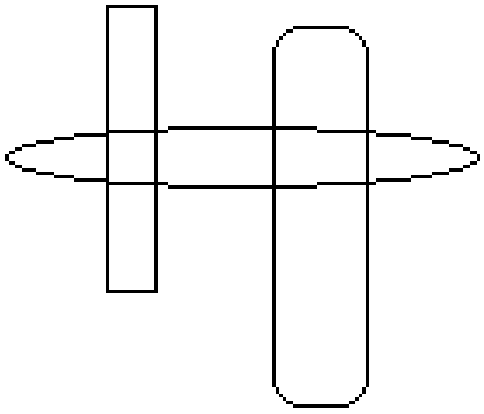
Reference Books:

10. *Wireless Communications – Principles and Practice by Theodore S Rappaport, Pearson Education.*
11. *Wireless Communication and Networks by William Stallings, Second Edition, Prentice Hall*
12. *https://developer.android.com*

Course Outcomes:

The student will be able to –

6. Select components and radio spectrum for PCS based on bandwidth requirement.
7. Justify the Mobile Network performance parameters and design decisions.
8. Choose the modulation technique for setting up mobile network.
9. Formulate GSM/CDMA mobile network layout considering futuristic requirements which conforms to the technology.
10. Deploy the 3G/4G technology based network with bandwidth capacity planning.
11. Adapt to the requirements of next generation mobile network and mobile applications.



CS4018: DATA SCIENCE

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
: Lab 2 Hours / Week

Section 1: Topics/Contents

Introduction: Big Data Analytics, Data Scientist Role, characteristics of Big Data, Data Analytics Lifecycle
Descriptive Statistics: Data Objects and Attribute Types, Basic Statistical Descriptions of Data, Measuring Data Similarity and Dissimilarity, Data Visualization
Data Preprocessing: An Overview, Data Cleaning, Data Integration, Data Reduction, Data Transformation and Data Discretization.
Predictive Analytics: Linear Regression, Logistic Regression.
Market Basket Analysis: Association Rules, Optimization Techniques.

Section2: Topics/Contents

Classification: Supervised learning, Bayesian Classification, k-Nearest Neighbors (k-NN), Decision tree, Support Vector Machines, Neural Networks, Overfitting/ Underfitting, bagging/boosting and ensemble methods, Classifier performance measures, confusion matrix, Cross validation
Clustering and Outlier Detection: Unsupervised learning, K-means algorithm, hierarchical clustering, Interpretation of clusters and validation, Introduction to outlier mining, Applications, Detection Techniques
Text mining: Text preprocessing, text mining operations, Categorization, Text mining applications.

List of Practical: (For THL, TLP courses):
Implement following assignments Using R/Python

1. Getting Started with R/Python installation, R/Python objects and basic statistics.
2. Data preprocessing, exploratory analysis, visualization.
3. Correlation and regression analysis
4. Linear Regression and Logistic Regression
5. Association Rules
6. Decision tree classifier
7. Naive Bays classifier
8. K-means and hierarchical clustering

Social media analytics: Twitter/Facebook/Whats app sentiment analysis

Text Books:

1. *Cathy O'Neil and Rachel Schutt. Doing Data Science, Straight Talk From The Frontline. O'Reilly. 2014.*
2. *"Data Mining: Concepts and Techniques", Jiawei Han and Micheline Kamber, Morgan Kaufman, ISBN 978-81-312-0535-8, 2nd Edition.*

Reference Books:

1. Peng, Roger D and Elizabeth Matsui, "The Art of Data Science." A Guide for Who Works with Data. Skybrude Consulting 200 (2015): 162.

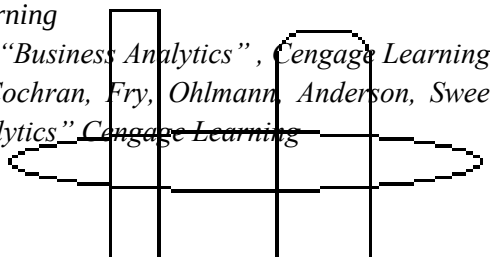
2. Evans, James R., and Carl H. Lindner, "Business analytics: the next frontier for decision sciences." Decision Line 43.2 (2012): 4-6.

3. James, G., Witten, D., Hastie, T., Tibshirani, R. An introduction to statistical learning with applications in R. Springer, 2013.

4. Albright, Winston "Business Analytics: Data Analysis and decision making" Cengage Learning

5. Sahil Raj, "Business Analytics" , Cengage Learning

6. Camm, Cochran, Fry, Ohlmann, Anderson, Sweeney, Williams, "Essentials of Business Analytics" Cengage Learning



Course Outcomes:

The student will be able to –

1. Perform exploratory data analysis.
2. Apply regression to real world examples.
3. Build and use classification model for given task.
4. Identify appropriate clustering algorithm based on data set.
5. Detect outliers in data set.
6. Analyze social media sentiments.

CS4030: BUSINESS INTILEGENCE

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
: Lab 2 Hours / Week

Section 1: Topics/Contents

Introduction: What is business intelligence and analytics? Need for BI&A, how businesses use BI&A. Evolution of BI&A. Interplay among Business Intelligence, Business Analytics, Data Science, Data Mining, Data Analytics, Data Warehousing, Statistics and Machine Learning. Drawing insights from data: DIKW pyramid, Categorization of Analytics: DIPP. Concept of a model and its characterization.

Data Preprocessing: Typical preprocessing operations: combining values into one, handling incomplete or incorrect data, handling missing values, recoding values, subsetting, sorting, transforming scale, determining percentiles, data manipulation, removing noise, removing inconsistencies, transformations, standardizing, normalizing - min-max normalization, z-score standardization, rules of standardizing data.

Descriptive Statistics: role of statistics in analytics, types of data (scales of measurement - NOIR), data distributions, measures of variability (range, quartile, five number summary, variance, std dev, coeff of variation), analyzing distributions, Chebychev's Inequality, measures of shape (skewness, kurtosis), measures of association (covariance,correlation), outliers.

Inferential Statistics: Role of probability in analytics. Need for sampling, generating samples, sampling and non-sampling error. Sampling Distribution of Mean, Central LimitmTheorem, Standard Error. **Estimation:** Point and Interval Estimates, Confidence Intervals, level of confidence, sample size.

Section2:Topics/Contents

Hypothesis Testing: basic concepts, Errors in hypothesis testing, Power of test, Level of significance, p-value, general procedure for hypothesis testing. Parametric tests – z test, ttest, chi-square test. Hypothesis testing of means: two tailed and one-tailed tests. Chi-square test for independence and goodness of fit. Hypothesis testing for comparing two related samples. Limitations of hypothesis testing. Picking up the right test for a given scenario.**Similarity Measures:** Design of recommender systems - user based and item based collaborative filtering. **Regression Analysis:** Correlation and regression, Simple Linear Regression Model,Least Squares Method. Making Data Models more flexible, making data models more selective, dealing with Categorical variables.
Introduction to Time Series Analysis and Forecasting: Time series patterns, forecast accuracy, moving averages and exponential smoothing, casual models, using regression

analysis for forecasting.

List of Practicals: (For THL, TLP courses)

1. Getting started: understand what business do with their data with a scenario based application using QlikView tool

2. Designing an end to end warehousing solution for a real world scenario involving multi-dimensional modeling, designing data cube, doing ETL, OLAP and reporting

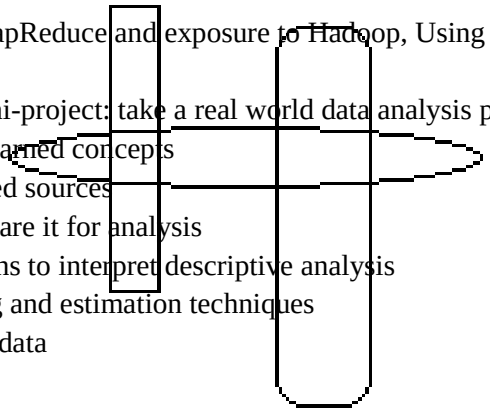
3. Getting started with R

4. Using R for data preprocessing, exploratory analysis, visualization, correlation and regression analysis, hypothesis testing, chi square test

5. Data analysis case study using R for a readily available data set

6. BigData Analytics - MapReduce and exposure to Hadoop, Using R over Hadoop

7. [Optional] A group mini-project: take a real world data analysis problem and solve it using the above learned concepts
a. Getting Data from varied sources
b. Data massaging to prepare it for analysis
c. Generating visualizations to interpret descriptive analysis
d. Implementing sampling and estimation techniques
e. Regression analysis on data
f. Hypothesis testing



Text Books:

1. "Business Analytics" by James R Evans, Pearson
2. "Fundamentals of Business Analytics", by R. N. Prasad, Seema Acharya, ISBN: 978-81-256-3203-2, Wiley-India

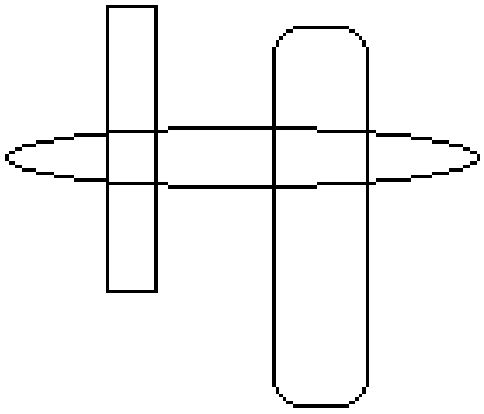
Reference Books:

1. "Data Mining: Concepts and Techniques", Jiawei Han and Micheline Kamber, Morgan Kaufman, ISBN 978-81-312-0535-8, 2nd Edition
2. "Essentials of Business Analytics" by Camm, Cochran, Fry, Ohlmann, Anderson, Sweeney, Williams, Cengage Learning
3. "The Kimball Group Reader: Relentlessly Practical Tools for Data Warehousing and Business Intelligence" 2010, Ralph Kimball, Margy Ross, Wiley Publications

Course Outcomes:

The student will be able to –

1. Construct an end-to-end data warehousing solution for business intelligence involving various data sources, ETL, multi-dimensional modeling, OLAP, reporting and analytics
2. Evaluate various data processing algorithms in their applicability to different problems
3. Display the process of converting data into a user defined format required for particular analysis
4. Utilize statistical tools in deriving insights from data
5. Describe various techniques for descriptive, predictive and prescriptive analytics
6. Apply various techniques to solve real-world data analysis problems



CS 4014: SOFTWARE TESTING AND QUALITY ASSURANCE

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
: Lab 2 Hours / Week

Section 1: Topics/Contents

Software Measurement: Measurement Theory, Software Measurement and Models, Measurement Scales, Classification of Software Measures, Measurement Framework, Theory of Program Testing, Graph Theory for Testers, Software Complexity, Measuring Internal Product Attributes: Size, Measuring Internal Product Attributes : Structure, Halstead's Software Science, Product Quality Metrics, In-Process Quality Metrics, Software Reliability: Measurement and Prediction, The Rayleigh Model, SRE process, Reliability Concepts: Hardware and Software, Deploying SRE

Principles of Testing: Testing Concepts: Purpose of Software Testing, Testing Principles, Goals of Testing, Testing aspects: Requirements, Test Scenarios, Test cases, Test scripts/procedures, Strategies for Software Testing, Testing Activities, Mistakes, Faults & Failures, Planning for Verification and Validation, Software Inspections, Automated Static Analysis, Verification and Formal Methods, Levels of Testing

White-Box Testing: Test Adequacy Criteria, Static Testing, Structural Testing, Code Complexity Testing, Mutation Testing, Data Flow Testing

Black-Box Testing: Test Case Design Criteria, Requirement Based Testing, Positive and Negative Testing, Boundary Value Analysis, Equivalence Partitioning State Based Testing, Domain Testing

Functional Testing: Test Plan, Test Management, Test Execution and Reporting, Test Specialist Skills, Tester's Workbench and Tool Categories, Test Maturity Model and Test Process Assessment, Debugging & Root Cause Analysis, Software Items, Component & Units, Test Bed, Traceability and Testability, Attributes of Testable Requirements, Test Matrix, Types of Testing Documentation, Verification Testing, Validation Testing, Integration Testing, System and Acceptance Testing, GUI Testing,Regression Testing, Selection, Minimization and Prioritization of Test Cases for Regression Testing, Creating Test Cases from Requirements and Use cases, Software Defects: Origins of Defects, Defect Classes, Defect Repository / Test Design, Defect Repository

Section2: Topics/Contents

Higher Order Testing: Object Oriented Testing, Specification Based Testing, Performance Testing, Ad-hoc Testing, Usability and Accessibility Testing, Risk-based Testing, Exploratory Testing, Scenario-based Testing, Random Testing Compatibility Testing, User Documentation Testing, Client–Server System Testing, RAD Testing, Configuration Testing, Testing internal Controls, Multiplatform Environment Testing, Security Testing, Web-based System Testing, Reliability Testing, Efficiency Testing, Maintainability Testing, Portability Testing, Introduction

to Performance Testing, Application Performance Testing, Process of Performance Testing, Effective Root-Cause analysis, Testing VS Test Automation, Tool evaluation and selection, Automation team roles, Architectures, Planning and implementing test automation process

Introduction to Software Quality Assurance: The software quality challenge, Meaning of software quality, Software quality factors , Software Quality Lessons Learned, The components of the software quality assurance system, Pre-project software quality components: Contract Review, Development and quality plans, SQA components in the project life cycle: Integrating quality activities in the project life cycle, Assuring the quality of software maintenance components, Assuring the quality of external participants' contributions, CASE tools, Software quality infrastructure components, Pareto Principles, Total Quality Management, Ishikawa's Seven Basic Tools

Software Quality Assurance Management: Management components of software quality: Project progress control, Software quality metrics, Costs of software quality, Standards, certification and assessment: Quality management standards, SQA project process standards – IEEE software engineering standards, Management and its role in software quality assurance, The SQA unit and other actors in the SQA system, Inspection as an Up-Front Quality Technique, Software Audit Methods, Software Safety and Its Relation to Software Quality Assurance, SQA for Small Projects, Development Quality Assurance, Quality Management in IT, Introduction to ITIL, Software Quality Assurance Metrics, Software Benchmarks and Baselines

List of Practicals: (For THL, TLP courses)

11. To Prepare Test Plan for the implemented system under test. The Test Plan shall be based on System Requirement Specification. The Test plan consists of following issues.
    a. Purpose of the test. /Location and schedule of the test.
    b. Test descriptions. /Pass and Fail Criteria.
12. To identify and narrate Test cases, Test scripts/procedures and Test incident Report identifier for the system under test. Refer Use case analysis document to prepare mentioned/ identified test documents. The expected specifications/ behaviors can be stated with the help of Test Oracle.
13. To perform Unit testing especially indicating the traced Independent data paths, Control paths and Error handling paths. Prepare control flow graphs for the unit under test. Compute the Cyclomatic complexity of the unit.
14. To perform Data Flow testing for the Program Segments by identifying the Definition-Use chain and type of data flow anomaly.
15. To perform Mutation Analysis of the Program Segments along with mutant history, mutation score and type of mutation by using any Code analysis Tool / Mutation Testing Tool (JUNIT, MuJava).
16. To perform Black-Box Testing for all the units contained in the architectural segments using Equivalence Partitioning, Boundary Value Analysis and Orthogonal Array testing methods. To study exploratory Testing for the Module under Test and merits/demerits of this technique.
17. To perform Regression Testing / GUI Testing of the System under construction with Unit and Integration profiles by using any Functional Testing Tool.
18. To perform Automated Testing using suitable CASE tool addressing Higher-Order testing strategies.

19. To perform Web Based Testing for Web Application incorporating any Open Source Tool. To study Performance Testing, Load Testing, Security Testing, Stress Testing, Demonstrate on link Test expectation.
20. To perform Software Audit (Checklist and Template-based) for the software developed and improve the Code Quality.

Text Books:

4. *Burnstein, "Practical Software Testing", Springer International Edition, ISBN 81-8128-089-X*
5. *William E. Perry, " Effective Methods for Software Testing", John Wiley and Sons, ISBN 9971-51-345-5*
6. *Daniel Galin, Software Quality Assurance: From theory to implementation, Pearson Education Limited, 2004, ISBN 0201 70945 7*

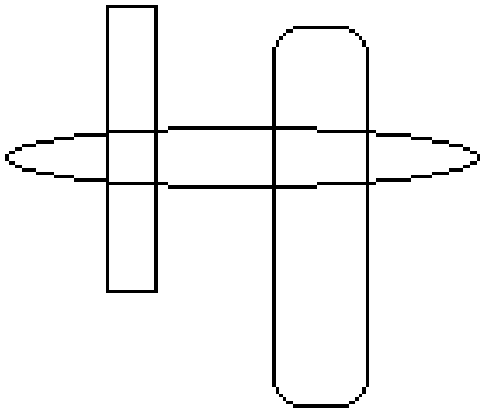
Reference Books:

8. *KshirasagarNaik, PriyadarshiTripathy, Software Testing and Quality Assurance-Theory and Practice, John Wiley & Sons, Inc., 2008, ISBN 978-0-471-78911-6*
9. *Fenton, Pfleeger, "Software Metrics: A Rigourous and practical Approach", Thomson Brooks/Cole, ISBN 981-240-385-X.*
10. *Desikan, Ramesh, "Software Testing: principles and Practices", Pearson Education, ISBN 81-7758-121-X.*
11. *Anne MetteJonassen Hass, Guide to Advanced Software Testing, ARTECH HOUSE, INC., 2008, ISBN-13: 978-1-59693-285-2*
12. *Ian Molyneaux,The Art of Application Performance Testing, O'Reilly Media, Inc., 2009, ISBN: 978-0-596-52066-3*
13. *Jamie L. Mitchell, Rex Black, Advanced Software Testing—Vol. 3, 2nd Edition, Rocky Nook, 2015, ISBN: 978-1-937538-64-4*
14. *G. Gordon Schulmeyer, Handbook of Software Quality Assurance Fourth Edition, ARTECH HOUSE, INC., 2008, ISBN-13: 978-1-59693-186-2*

Course Outcomes:

Upon completion of the course, graduates will be able to –

7. Select and classify measurement scales and models, software metrics and measures addressing software quality and reliability.
8. Conduct unit and integration tests by determining test design, test automation, test coverage criteria using testing frameworks and test adequacy assessment using control flow, data flow, and program mutations.
9. Apply suitable higher order testing techniques and methods in order to achieve verified and validated software by following testing best practices.
10. Demonstrate the skillset as a tester to neutralize the consequences of wicked problems by narrating effective test cases and test procedures.
11. Adapt to various test processes, types of errors and fault models and methods of test generation from requirements for continuous quality improvement of the software system along with Software Quality best practices usage.
12. Apply software testing cycle in relation to software development and project management focusing incidents and risks management within a project towards efficient delivery of software solutions and implement improvements in the software development processes by making use of standards and baselines.



CS 4015: SOFTWARE ARCHIECTURE AND DESIGN

Credits: 04 **Teaching Scheme Theory :3 Hours / Week**
Lab :2 Hours / Week

Section 1: Topics/Contents

Introduction to Software Architecture: Architecture Orientation Framework: Motivation, Overview of the Framework, Architectures and Architecture Disciplines (What), Architecture Perspectives (Where), Architecture Requirements (Why), Architecture Means (With What), Organizations and Individuals (Who), Architecture Method (How), Architectures and Architecture Disciplines (What): Classic Architecture as Starting Point, From Classic Architecture to Software Architecture, Architecture and the System Concept, Architecture and the Building Blocks of a System, Architecture Perspectives (Where): Architecture Levels, Organizational Level, System Level, Building Block Level, Architecture Views, Architecture Requirements (Why): Requirements Characteristics and Types, Organizational Requirements, System Requirements, Building Block Requirements, Qualities and Constraints, Requirements in the Context of Architecture, Architecture Means (With What): Architecture Principles, Basic Architecture Concepts, Architecture Tactics, Styles, and Patterns, Organizations and Individuals (Who), Architecture Method (How)

Software Architecture Styles: Three Categories of Styles, Style Guides: A Standard Organization for Explaining a Style, Choosing Which Element and Relation Properties to Document, Notations for Architecture Views, Module Views, Module Styles: Decomposition Style, Uses Style, Generalization Style, Layered Style, Aspects Style, Data Model Component-and-Connector Views: Overview, Elements, Relations, and Properties of C&C Views, C&C Views applicability, Perspectives: Choosing Connector Abstractions, Notations for C&C Views, Relation to Other Kinds of Views, Component-and-Connector Styles: Data Flow Styles, Pipe-and-Filter Style, Call-Return Styles, Client-Server Style, Peer-to-Peer Style, Service-Oriented Architecture Style, Event-Based Styles, Publish-Subscribe Style, Repository Styles, Shared-Data Style, Crosscutting Issues for C&C Styles, Allocation Views and Allocation Styles: Deployment Style, Install Style, Work Assignment Style, Other Allocation Styles

Architectural Decisions: Refinement, Descriptive Completeness, Documenting Context Diagrams, Documenting Variation Points, Documenting Architectural Decisions, Combining Views, Documenting Software Interfaces: Interface Documentation, Showing the Existence of Interfaces in Diagrams, A Standard Organization for Interface Documentation, Coming to Terms: Error Handling, Stakeholders of Interface Documentation, Conveying Syntactic Information, Conveying Semantic Information, Coming to Terms: Signature, Interface, API, Architectural Documentation Relevance: Documenting Behavior, Building the Architecture Documentation, Building the Documentation Package, Reviewing an Architecture Document

Section2: Topics/Contents

Collaborative Architecture Determination: The Impact of People and Leadership on Scalability, Roles for the Scalable Technology Organization, Designing Organizations, Leadership 101, Management 101, Making the Business Case, Understanding Why Processes Are Critical to Scale, Managing Incidents and Problems, Managing Crisis and Escalations, Controlling Change in Production Environments, Exploring Architectural Principles, Joint Architecture Design, Architecture Review Board

Architecting Scalable Solutions: Focus on Core Competencies: Build Versus Buy, Performance and Stress Testing, Barrier Conditions and Rollback, Designing for Any Technology, Creating Fault Isolative Architectural Structures, Introduction to the AKF Scale Cube, Splitting Applications for Scale, Splitting Databases for Scale, Caching for Performance and Scale, Asynchronous Design for Scale, Large Scale Data, Clouds and Grids, Soaring in the Clouds, Plugging in the Grid, Monitoring Applications, Planning Data Centers, Introduction to Middleware Technologies, Reaching for the Clouds, Defining the Clouds for the Enterprise, Making the Business Case for Clouds, Working from Data, Services, Processes to the Clouds

Software Architecture Practices: SOA and Channels, Service-Oriented Architecture Enterprise Architecture Framework and Methodology, Incorporating Existing Enterprise Architecture Documents and Artifacts into the SOA~EAF, Dealing with Purchased or Leased Business Applications, Transforming Governance to Support SOA, SOA System Development Life Cycle, Capacity Planning under SOA, People Involved in the SOA Process, Implementing an Effective SOA Strategy under a Decentralized Business or IT Model, SOA Business Strategy and Roadmap

List of Practical's: (For THL, TLP courses)

8. To narrate Requirement Definition Document for the target system with following three areas: Problem Identification, Problem Definition, and Problem Statement
9. To narrate System Requirements Specification Document for target system with reference to the IEEE 610.12.1990 Std guidelines.
10. To narrate System Architecture Requirement Specification Document for target system with stakeholder and roles description.
11. To select appropriate Architectural View and Style and prepare Architecture Diagram for the target system.
12. To prepare Architecture Decision document describing Architectural Decisions, Software Interfaces, and behaviors along with Architectural Review.
13. To implement the target system using the Technical Architecture conforming to technology availability and scalability.
14. To create Test Plan, Test Cases and apply them to test the performance adequacy of the system implemented.

Text Books:

3. Paul Clements, Felix Bachmann, Len Bass, David Garlan, Documenting Software Architectures: Views and Beyond Addison-Wesley Professional 2003, ISBN-10:0201703726, ISBN-13: 9780201703726

4. Martin L. Abbott, Michael T. Fisher, The Art of Scalability, Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise, Pearson Education, Inc., 2010, ISBN-13: 978-0-13-703042-2

Reference Books:

6. Rick Sweeney, Achieving Service-Oriented Architecture, Applying an Enterprise Architecture Approach, John Wiley Publications, 2010, ISBN 978-0-470-60451-9

7. David S. Linthicum, Cloud Computing and SOA Convergence in Your Enterprise, 2010 Pearson Education, Inc., ISBN-13: 978-0-13-600922-1

8. Arno Puder, Kay Römer, Frank Pilhofer, Distributed Systems Architecture, A Middleware Approach, 2006 by Elsevier Inc., ISBN 13: 978-1-55860-648-7

9. ParthaKuchana, Software Architecture Design Patterns in Java, 2004 by CRC Press LLC, ISBN 0-8493-2142-5

10. C.E. Dickerson, D.N. Mavris, Architecture And Principles Of Systems Engineering, 2010 by Taylor and Francis Group, LLC, ISBN 978-1-4200-7253-2

Course Outcomes:

Upon completion of the course, graduates will be able to –

7. Examine and breakdown real-world problem scenarios into structured partitions depicting static and dynamic behavior of the system using Software Architecture Requirements Capture practices.
8. Identify and formulate software requirements and behavioral models using Architectural Assessment of behavioral views by selecting appropriate architectural views.
9. Compose system design specifications indicating logical, physical, deployment, and concurrency viewpoints using software interface explorations, error handling, and architectural reviews.
10. Construct and justify the evolutionary system description models expressing high-level technical architecture accommodating applicable architectural styles compatible to requirements using CASE tools.
11. Comprehend the nature of architectural styles implementation by understanding numerous examples from different technology categories and publish guidance, applicability, reasonableness, and relation to other design criteria resulting in well-documented system profiles to the engineering and social community.
12. Propose multi-faceted defendable solutions demonstrating team-skills accommodating architectural views and styles reducing the potential cost and performance impedance in order to realize system artifacts with the help of Architecture Derivation and Development practices.

CS4022: SOFTWARE DESIGN METHODOLOGIES

Credits: 04 **Teaching Scheme: Theory 3 Hours / Week**
Lab 2 Hours / Week

Section 1: Topics/Contents

Agile - High level overview: Concept of iterative development: Waterfall vs Agile, Challenges of Waterfall project, Iron triangle of project management upside down, Development in Agile vs waterfall Agile in nutshell: Principles behind the Agile Manifesto, Extreme programming, Agile Ceremonies, Daily standups, Retrospective, Important agile activities, Backlog Grooming, Spring Planning, Agile methodologies, TDD(Test Driven development), BDD(Behavior Driven development), ATDD(Acceptance Test Driven development), Tools used for Agile Rally Agile Team roles: Scrum Master, Product Owner, Team Member, Agile Metrics: Do/Say, Velocity.

AngularJs Overview : Single Page application architecture, MVVM framework - Binding in AngularJs, Services and dependency injection, Data Binding and Directives, Custom Directives Debugging in AngularJs.

Nodejs Overview : Nodejs overview, Asynchronous Nodejs, Deploying nodejs app, Testing, Security.

Junit : Junit Overview, Test Driven Development with Junit

Spring Boot - Creating micro-services: Spring REST, Spring Boot – overview, JPA (Java Persistence API) – Overview, Writing micro-services with Spring Boot.

Git / SVN: Committing, Browsing, Tagging and Branching, Merging.

Section2: Topics/Contents

AWS - Overview: AWS Overview: Different services available, Advantages of AWS, Multi-AZ, Region. IAM: Roles, Policy, EC2, Terminology, EC2 Setup. Overview of S3: Terminology, Lifecycle management.

Search Technologies: Apache Solr: Text centric search engine, Inverted Index, How to setup first apache solr instance?, Tokenizer, Filters, How to integrate Apache Solr with your spring application?, Contextual Search, Solr Cloud – Zookeeper ensemble, Tuning Solr. Elastic Search: Mapping in Elastic Search, Integrating Elastic Search with spring application, How is Elastic search different than Solr?

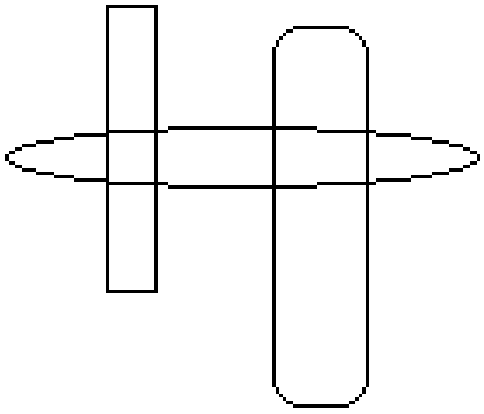
DevOps – Jenkins: What is DevOps?, Tools in DevOps, Jenkins, DeployIT, How to setup Jenkins Jobs?, How to create CI/CD pipeline?.

Design pattern & clean code practices: Core Design Patterns: Singleton, Abstract Factory, Proxy, Strategy, Template, Chain of responsibility.J2EE Design Patterns: DAO design pattern, Front Controller, Inversion of Control, Intercepting Filter Pattern. Design Principles: SOLID-single responsibility, open-closed, Liskov substitution, interface segregation and dependency inversion. Clean Code Practices: SRP(Single Responsibility Principle), DRY (Do not repeat yourself), KISS (Keep it short and simple).

Text Books:

Reference Books:

Course Outcomes:
The student will be able to –



MODULE VII/ VIII

Issue 1, Rev. 1 , Dt. 01/07/2018

Title: Course Structure **FF No. 653**

Branch: Computer **Year:** B.Tech. **A.Y.:** 2019-20 **Module:** VII/VIII **Pattern:** B-19.

1.

Subject No.	Subject Code	Subject Name	Teaching Scheme (Hrs/Week)			Examination Scheme						Credits
			Theory	LAB	Tut.	CA		MSE (%)	ESA		Total	
						HA (%)	LAB (%)		ESE (%)	VIVA (%)		
INT.		Semester Internship.	-	-	-	-	-	30	-	70	100	16
		TOTAL	**-**	**-**	**-**	**-**	**-**	**30**	**-**	**70**	**100**	**16**