M.Tech. CSE (Software Engineering) (1ST SEM.)
TOTAL CONTACT HRS. = 24, TOTAL CREDITS = 22

Course		Contact Hrs.			Marks			Credits
Code	**Name**	**L**	**T**	**P**	**Int.**	**Ext.**	**Total**	
MCSE3-101	Advanced Data Structures and Algorithm	3	1	0	40	60	100	4
MREM0-101	Research Methodology	4	0	0	40	60	100	4
MCSE3-103	Soft Computing	3	1	0	40	60	100	4
Departmental Elective-I		3	1	0	40	60	100	4
MCSE3-156	Software requirements							
MCSE3-157	Software Project Estimation and Planning							
MCSE3-158	Software Engineering Concepts and Methodologies							
MCSE3-159	Object Oriented Analysis and Design							
Departmental Elective-II		3	1	0	40	60	100	4
MCSE3-160	Software project management							
MCSE3-161	Advanced Operating System							
MCSE3-162	Software Architecture and Design Patterns							
MCSE3-163	Software Maintenance and Management							
MCSE3-104	**Practical Lab.-I**	0	0	4	60	40	100	2
Total 5 Theory & 1 Lab. Courses		**16**	**4**	**4**	**260**	**340**	**600**	**22**

M.Tech. CSE (Software Engineering) (2nd SEM.)
TOTAL CONTACT HRS. = 24, TOTAL CREDITS = 22

Course		Contact Hrs.			Marks			Credits
Code	**Name**	**L**	**T**	**P**	**Int.**	**Ext.**	**Total**	
MCSE3-205	Agile Software Development	3	1	0	40	60	100	4
MCSE3-206	Software Testing& Validations	3	1	0	40	60	100	4
Departmental Elective-III		3	1	0	40	60	100	4
MCSE3-264	Information Retrieval							
MCSE3-265	Software Quality Management							
MCSE3-266	Unified Software Configuration Management							
MCSE3-267	Software Design and Construction							
Departmental Elective-IV		3	1	0	40	60	100	4
MCSE3-268	Secure Software Engineering							
MCSE3-269	Software Project Metrics							
MCSE3-270	Software Reliability and Metrics							
MCSE3-271	Open Source Technology							
Open Elective I		3	1	0	40	60	100	4
MCSE3-207	Practical Lab.-II	0	0	4	60	40	100	2
Total 5 Theory & 1 Lab. Courses		**15**	**5**	**4**	**260**	**340**	**600**	**22**

M.Tech. CSE (Software Engineering) (3rd SEM.)
TOTAL CONTACT HRS. = 8, TOTAL CREDITS = 21

Course		Contact Hrs.			Marks			Credits
Code	Name	L	T	P	Int.	Ext.	Total	
Departmental Elective-V		3	1	0	40	60	100	4
MCSE3-372	Component Based Development							
MCSE3-373	Object Oriented Engineering Using UML							
MCSE3-374	Software Reliability & Metrics							
MCSE3-375	Usability Engineering							
Open Elective-II		3	1	0	40	60	100	4
MCSE3-308	Project	-	-	-	60	40	100	10
MCSE3-309	Seminar	-	-	-	60	40	100	4
Total 2 Theory Courses		6	2	0	200	200	400	22

M.Tech. CSE (Software Engineering) (4th SEM.)
TOTAL CREDITS = 25

Course		Contact Hrs.			Marks			Credits
Code	Name	L	T	P	Int.	Ext.	Total	
MCSE3-410	Dissertation	0	0	0	60	40	100	24
Total		0	0	0	60	40	100	24

Total Marks = 600 + 600 + 400 + 100 = 1700
Total Credits = 22 + 22 + 22 + 24 = 90

ADVANCED DATA STRUCTURES AND ALGORITHMS

Subject Code- MCSE1-101 MCSE2-101, MCSE3-101, MCSE4-101	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:
To learn the advanced concepts of data structure and algorithms and its implementation. The course has the main ingredients required for a computer science graduate and has all the necessary topics for assessment of data structures and algorithms.
LEARNING OUTCOMES:
CO1 Ability to apply and implement various data structures to algorithms and to solve problems.
CO2 Basic ability to analyze algorithms and to determine algorithm correctness and time efficiency class.
CO3 Ability to apply various traversing, finding shortest path and text pattern matching algorithm.
CO4 Know the concepts of tractable and intractable problems and the classes P, NP and NP-complete problems.

UNIT-I (12 Hrs.)

Introduction to Basics: Significance and need of various data structures and algorithms, Arrays, linked lists, Stacks, Queues, Priority queues, Heaps; Strategies for choosing the appropriate data structures.
Advanced Data Structures: Binary Search Tree**,** AVL Trees, Red-Black Trees, Splay Trees, B-trees, Fibonacci heaps, Data Structures for Disjoint Sets, Augmented Data Structures.

UNIT-II (10 Hrs.)

Algorithms Complexity and Analysis: Probabilistic Analysis, Amortized Analysis, Competitive Analysis, Internal and External Sorting algorithms: Quick Sort, Heap Sort, Merge Sort, Counting Sort, Radix Sort.

UNIT-III (12 Hrs.)

Graphs & Algorithms: Representation, Type of Graphs, Paths and Circuits: Euler Graphs, Hamiltonian Paths & Circuits; Cut-sets, Connectivity and Separability, Planar Graphs, Isomorphism, Graph Coloring, Covering and Partitioning, bridges, Depth- and breadth-first traversals, Minimum Spanning Tree: Prim's and Kruskal's algorithms, Shortest-path Algorithms: Dijkstra's and Floyd's algorithm, Topological sort, Max flow: Ford-Fulkerson algorithm, max flow – min cut.
String Matching Algorithms: Suffix arrays, Suffix trees, Brute Force, Rabin-Karp, Knuth-Morris-Pratt, Boyer-Moore algorithm.

UNIT-IV (11 Hrs.)

Approximation algorithms: Need of approximation algorithms: Introduction to P, NP, NP-Hard and NP-Complete; Deterministic, non-Deterministic Polynomial time algorithms; Knapsack, TSP, Set Cover, Open Problems.
Randomized algorithms: Introduction, Type of Randomized Algorithms, 2-SAT; Game Theoretic Techniques, Random Walks.

RECOMMENDED BOOKS:
1. E. Horowitz, S. Sahni and Dinesh Mehta, 'Fundamentals of Data structures in C++', Galgotia, **1999**.
2. Thomas H. Corman, Charles E. Leiserson, Ronald L. Rivest, 'Introduction to Algorithms', 3rd Edn., PHI, **2009**.

3. Adam Drozdex, 'Data Structures and Algorithms in C++', 2nd Edn., Thomson Learning – Vikas Publishing House, **2001**.
4. G. Brassard and P. Bratley, 'Algorithmics: Theory and Practice', Prentice –Hall, **1988**.

RESEARCH METHODOLOGY

Subject Code: MREM0-101 **L T P C** **4 0 0 4** **Duration: 45 Hrs.**

UNIT–I (11 Hrs.)

Introduction to Research: Meaning, Definition, Objective and Process
Research Design: Meaning, Types - Historical, Descriptive, Exploratory and Experimental
Research Problem: Necessity of Defined Problem, Problem Formulation, Understanding of Problem, Review of Literature
Design of Experiment: Basic Principal of Experimental Design, Randomized Block, Completely Randomized Block, Latin Square, Factorial Design.
Hypothesis: Types, Formulation of Hypothesis, Feasibility, Preparation and Presentation of Research Proposal

UNIT–II (10 Hrs.)

Sources of Data: Primary and Secondary, Validation of Data
Data Collection Methods: Questionnaire Designing, Construction
Sampling Design & Techniques – Probability Sampling and Non Probability Sampling
Scaling Techniques: Meaning & Types
Reliability: Test – Retest Reliability, Alternative Form Reliability, Internal Comparison Reliability and Scorer Reliability
Validity: Content Validity, Criterion Related Validity and Construct Validity

UNIT–III (13 Hrs.)

Data Process Operations: Editing, Sorting, Coding, Classification and Tabulation
Analysis of Data: Statistical Measure and Their Significance, Central Tendency, Dispersion, Correlation: Linear and Partial, Regression: Simple and Multiple Regression, Skewness, Time series Analysis, Index Number
Testing of Hypothesis: T-test, Z- test, Chi Square, F-test, ANOVA

UNIT – IV (11 Hrs.)

Multivariate Analysis: Factor Analysis, Discriminant Analysis, Cluster Analysis, Conjoint Analysis, Multi-Dimensional Scaling
Report Writing: Essentials of Report Writing, Report Format
Statistical Software: Application of Statistical Softwares like SPSS, MS Excel, Mini Tab or MATLAB Software in Data Analysis
**Each Student has to Prepare Mini Research Project on Topic/ Area of their Choice and Make Presentation. The Report Should Consists of Applications of Tests and Techniques Mentioned in The Above UNITs.*

RECOMMENDED BOOKS:

1. R.I. Levin and D.S. Rubin, 'Statistics for Management', 7th Edn., Pearson Education New Delhi.
2. N.K. Malhotra, 'Marketing Research–An Applied Orientation', 4th Edn., Pearson Education New Delhi.
3. Donald Cooper, 'Business Research Methods', Tata McGraw Hill, New Delhi.
4. Sadhu Singh, 'Research Methodology in Social Sciences', Himalaya Publishers.
5. Darren George & Paul Mallery, 'SPSS for Windows Step by Step', Pearson Education New Delhi.

6. C.R. Kothari, 'Research Methodology Methods & Techniques', 2nd Edn., New Age International Publishers.

SOFT COMPUTING

Subject Code: MCSE1-103	**L T P C**	**Duration: 45 Hrs.**
MCSE2-103,	**3 1 0 4**	
MCSE3-103,		
MCSE4-103		

LEARNING OBJECTIVES:
The objective of this course is to teach basic neural networks, fuzzy systems, Genetic Algorithms and optimization algorithms concepts and their relations**.**
LEARNING OUTCOMES:
CO1 Able to comprehend techniques and applications of Soft Computing in real world problems.
CO2 Able to follow fuzzy logic methodology and design fuzzy systems for various applications.
CO3 Able to design feed forward Artificial Neural Networks (ANN) and implement various methods of supervised learning.
CO4 Able to design feedback Artificial Neural Networks (ANN) and implement various methods of unsupervised learning
CO5 Able to appreciate the methodology of GA and its implementation in various applications.

UNIT-I (11 Hrs.)

Soft Computing: Introduction of soft computing, soft computing vs. hard computing, various types of soft computing techniques, applications of soft computing.
Fuzzy Logic: Fuzzy set versus crisp set, basic concepts of fuzzy sets, membership functions, basic operations on fuzzy sets and its properties. Fuzzy relations versus Crisp relation,
Fuzzy Rule Base System: Fuzzy propositions, formation, decomposition & aggregation of fuzzy rules, fuzzy reasoning, Fuzzy Inference Systems (FIS) – Mamdani Fuzzy Models – Sugeno Fuzzy Models – Tsukamoto Fuzzy Models, Fuzzification and Defuzzification, fuzzy decision making & Applications of fuzzy logic.

UNIT-II (12 Hrs.)

Structure and Function of a Single Neuron: Biological neuron, artificial neuron, definition of ANN and its applications. Neural Network architecture: Single layer and multilayer feed forward networks and recurrent networks. Learning rules and equations: Perceptron, Hebb's, Delta, winner take all and out-star learning rules. Supervised Learning Network: Perceptron Networks, Adaptive Linear Neuron, Multiple Adaptive Linear Neuron, Back Propagation Network, Associative memory networks, Unsupervised Learning Networks: Competitive networks, Adaptive Resonance Theory, Kohnen Self Organizing Map.

UNIT-III (11 Hrs.)

Genetic Algorithm: Fundamentals, basic concepts, working principle, encoding, fitness function, reproduction, Genetic modeling: selection operator, cross over, mutation operator, Stopping Condition and GA flow, Constraints in GA, Applications of GA, Classification of GA.

UNIT-IV (11 Hrs.)

Hybrid Soft Computing Techniques: An Introduction, Neuro-Fuzzy Hybrid Systems, Genetic Neuro-Hybrid systems, Genetic fuzzy Hybrid and fuzzy genetic hybrid systems

RECOMMENDED BOOKS:

1. S. Rajasekaran & G.A. Vijayalakshmi Pai, 'Neural Networks, Fuzzy Logic & Genetic Algorithms, Synthesis & Applications', 1st Edn., PHI Publication, **2003**.
2. S.N. Sivanandam & S.N. Deepa, 'Principles of Soft Computing', 2nd Edn., Wiley Publications, **2008**.
3. Michael Negnevitsky, 'Artificial Intelligence', 2nd Edn., Pearson Education, New Delhi, **2008.**
4. Timothy J. Ross, 'Fuzzy Logic with Engineering Applications', 3rd Edn, Wiley, **2011.**
5. Bose, 'Neural Network Fundamental with Graph, Algorithm. & Application', TMH, **2004**.
6. Kosko, 'Neural Network & Fuzzy System', 1st Edn., PHI Publication, **2009.**
7. Klir & Yuan, 'Fuzzy Sets & Fuzzy Logic: Theory & Application', PHI, **1995**.
8. Hagen, 'Neural Network Design', 2nd Edn., Cengage Learning, **2008**.

SOFTWARE REQUIREMENTS

Subject Code: MCSE3-156	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:
It gives knowledge about various techniques used for acquisitioning the software requirements.

LEARNING OUTCOMES:
CO1: To understand various software requirements and techniques for how to manage these requirements.
CO2: To analyze the problems that comes under designing the software requirements.
CO3: To measure the various qualities like performance, availability, safety etc.
CO4: To understand the modeling of various software requirements and building the right system.

UNIT-I (11 Hrs.)

Introduction: Requirements Problem, Requirements management, Requirements and software life cycle-software team, Techniques for eliciting requirements, Languages and models for representing requirements.

UNIT-II (12 Hrs.)

Analyzing the Problem and Requirements in the Context of System Engineering: Business modeling, Systems engineering of software intensive systems, understanding user and stakeholders needs, features of a product or system, Interviewing, Requirements workshops, Brain storming and Idea reduction, Specifying and measuring external qualities: performance, reliability, availability, safety, security. Specifying and analyzing requirements for various types of systems: embedded systems, consumer systems, web based systems, business systems.

UNIT-III (11 Hrs.)

Requirements Modeling and Evaluation- Class diagrams, Agents, Operations, Behaviors, integrating multiple views, Goal diagrams, Risk analysis in goal diagrams- Requirements Evaluation, Requirements Specification and Documentation, Diagrammatic Notations- Use Case Example, Refining the use cases-developing the supplementary specification-Ambiguity and specificity.

UNIT-IV (11 Hrs.)

Building the Right System- From use cases to Implementation- From Use Cases to Test Cases-Tracing Requirements-Managing Change-Assessing Requirements Quality in Iterative Development-Agile Requirement methods.

RECOMMENDED BOOKS:

1. D. Leffingwell, D. Widrig, 'Managing Software Requirements: A Use Case Approach', 2nd Edn., Pearson Education, **2007**.
2. Karl E. Wiegers, 'Software Requirements', 2nd Edn.
3. Van Lamsweerde, 'A. Requirements Engineering', John Wiley & Sons, Ltd., UK, **2009**.
4. R.N. Taylor, et al., 'Softaware Achitecture: Foundations, Theory and Practice', John Wiley & Sons, Ltd. USA, **2009**.
5. Ian K. Bray, 'An Introduction to Requirements Engineering', Addison Wesley, **2002**.
6. Elizabeth Hull, Ken Jackson, Jeremy Dick,' Requirements Engineering', Springer-Verlag, **2004**.
7. Gerald Kotonya, Ian Sommerville, 'Requirements Engineering: Processes and Techniques', Wiley, **1998**.

SOFTWARE PROJECT ESTIMATION AND PLANNING

Subject Code: MCSE3-157	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:

To learn the advanced concepts of Software Engineering, Project Estimation, Project Planning and Project Management and its implementation for assessment of understanding the course by the students

LEARNING OUTCOMES:

CO1: Able to know about Introduction, history and development of software and its various process models.

CO2: Able to explain about project planning, estimation and Scheduling.

CO3: Able to study about software quality, its planning and assurance techniques.

CO4: Able to know how to manage the project, risks and case study of different software quality models

UNIT-I (11 Hrs.)

Principles and Motivations: History; definitions; why engineered approach to software development; Software development process models from the points of view of technical development and project management: waterfall, rapid prototyping, incremental development, spiral models, Agile Software Development, Emphasis on computer-assisted environments. Selection of appropriate development process.

UNIT-II (12 Hrs.)

Project Planning and Estimation: Characteristics of a software project, Software scope and feasibility, resources, the SPM plan, Size/scope estimation, Decomposition techniques, WBS. Effort estimation: Sizing, Function point, LOC, FP vs LOC. Schedule estimation: GANTT Charts, Activity networks, PERT/CPM networks. Cost estimation: Models: COCOMO I, COCOMO II.

UNIT-III (11 Hrs.)

Software Quality and its Assurance and Planning: Concepts of software quality, software quality control and software quality assurance, evolution of SQA, major SQA activities and issues, zero defect software, SQA techniques; Management review process, technical review process, walkthrough, software inspection process, configuration audits, and document verification, Error Reporting , Trend Analysis and Corrective Action: Identification ,Analysis and Correction of defect, implementation of correction, regression testing; Categorization of defect w.r.t development phases; Error quantity, error frequency, program unit complexity, compilation frequency; Corrective action and documenting the corrective action, periodic review of actions taken, Quality control, Quality assurance, Formal Technical Reviews, The SQA Plan, ISO and CMM standards.

UNIT-IV (11 Hrs.)

Software Project Management and Case Studies: Reactive vs. proactive Risk strategies, Risk projection, Risk Refinement, Risk Monitoring, Monitoring and management, RMMM plan., Earned Value Analysis., Team structures: hierarchical, Egoless, chief programmer, mixed; Team software Process; Resource leveling, Building a team: Skill sets, CASE tools, Quality management standards, Quality standards with emphasis on ISO approach, Capability Maturity Models-CMM and CMMI, TQM Models, Bootstrap methodology, The SPICE project, ISO/IEC 15504, Six Sigma Concept for Software Quality.

RECOMMENDED BOOKS:

1. Bob Hughes and Mike Cotterell, 'Software Project Management', 5th Edn., Tata McGraw Hill, **2009**.
2. Roger Pressman, 'A practitioner's Guide to Software Engineering', 8th Edn., Tata McGraw Hill, **2014**.
3. 'Head First PMP: A Brain Friendly Guide to Passing the Project Management Professional Exam', **2013**.

SOFTWARE ENGINEERING CONCEPTS AND METHODOLOGIES

Subject Code: MCSE1-158, MCSE3-158	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:

To impart knowledge on software engineering concepts and methodologies. To develop skills that will help the students to construct software using different methodologies and advanced techniques.

LEARNING OUTCOMES:

CO1: To study project management concepts
CO2: To understand the role of formal methods and reengineering
CO3: To understand the use of advanced techniques to develop the software.
CO4: To study the special requirements and development of different types of systems

UNIT-I (11 Hrs.)

Project Management: The management spectrum, The People; stakeholders, software team, Agile teams, coordination and communication issues, the product; problem decomposition, the process; modeling the product and process, process decomposition, The W^5 HH principle, RAD model, Metrics for process and projects, software measurements. Agile Methodology-Scrum and XP.

Cleanroom Software Engineering: The cleanroom approach, Functional specification, Cleanroom design and testing.

UNIT-II (12 Hrs.)

Formal Methods: Basic concepts, mathematical preliminaries, Applying mathematical notions for formal specification, Formal specification languages, Z specification Language, Formal methods- the road ahead.

Reengineering: Business process reengineering, Software reengineering, Reverse reengineering, Restructuring, Forward reengineering, economics of reengineering.

UNIT-III (11 Hrs.)

Component-Based Software Engineering: Engineering of component -based systems, CBSE process, Domain engineering, Component-based development, Classifying and retrieving components and economics of CBSE.

Computer-Aided Software Engineering: Building Blocks for CASE, taxonomy Of CASE tools, integrated CASE environments, Integration architecture, and CASE repository

UNIT-IV (11 Hrs.)

Web Engineering: Attributes of web-based applications, the Web E process, a framework for Web E. Formulating, analyzing web-based systems, design and testing for web-based applications, Management issues.

Client/Server Software Engineering: Structure of client/server systems, Software engineering for Client/Server systems, Analysis modeling issues, Design for Client/Server systems, Testing issues

RECOMMENDED BOOKS:

1. Roger S. Pressman, 'Software Engineering a Practitioners Approach', 5th Edn., McGraw-Hill, **2014**.
2. Sommerville, 'Software Engineering', 7th Edn., Pearson**, 2005**.
3. J. Bowan, 'Formal Specification and Documentation testing - A Case Study Approach', International Thomson Computer Press, **2003**.
4. James S. Peters, Witold Pedrycz, 'Software engineering an engineering approach', Wiley India, **2011**.

OBJECT ORIENTED ANALYSIS AND DESIGN

Subject Code: MCSE3-159	**L T P C**	**Duration: 45 Hrs.**
	3 1 0 4	

LEARNING OBJECTIVES:

To give students the detailed knowledge about Objects, Classes, types of modeling and detailed system design students will also come across the comparison of different methodologies.

LEARNING OUTCOMES:

CO1 Understanding Objects and classes and concept of generalization and inheritance
CO2 Learning Dynamic modeling and various functional models
CO3 Understanding system design and Object design
CO4 Comparing various methodologies and their implementation

UNIT-I (11 Hrs.)

Introduction to Object: Object Orientation, Development, Modeling, Object Modeling technique. Objects and classes, Links and Association, Generalization and inheritance, Grouping constructs, Aggregation, Abstract Classes, Generalization as extension and restriction, multiple inheritance, Meta data, Candidate keys, Constraints.

UNIT-II (12 Hrs.)

Dynamic modeling: Events and states, Nesting, Concurrency, Advanced Dynamic Modeling concepts, Functional modeling: Functional Models, Data flow diagrams, Specifying operations, Constraints, Relation of Functional model to Object and Dynamic Models.

Design Methodology, Analysis: Object modeling, Dynamic modeling, Functional modeling, adding operations, Iterating Analysis.

UNIT-III (11 Hrs.)

System Design: Subsystems Concurrency, Allocation to processor and tasks, Management of data stores, Handling Global Resources, Handling boundary Conditions, Setting Trade-off priorities.

Object Design: Overview, Combining the three models, Designing Algorithms, Design Optimization, Implementation of Control, Adjustment of Inheritance, Design of Associations, Object Representation, Physical Packaging, and Document Design Decision.

UNIT-IV (11 Hrs.)

Comparison of methodologies: Structured Analysis/Structured Design, Jackson Structured **Development. Implementation**: Using Programming Language, Database System, outside Computer.

Programming Style: Object Oriented Style, Reusability, Extensibility, Robustness, and Programming-in-the-large.

RECOMMENDED BOOKS:

1. Rambough, 'Object Oriented Modeling and Design', Pearson Education.
2. BOOCH, 'Object Oriented Analysis and Design', Addison Wesley.
3. Rebecca Wirfs-Brock, 'Design Object Oriented Software', PHI.

SOFTWARE PROJECT MANAGEMENT

Subject Code: MCSE3-160	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:

It gives an in depth knowledge of software project management and project planning. It also covers the Step Wise framework in project planning

LEARNING OUTCOMES:

CO1 Apply the basics of Software Project Management in order to manage and deliver qualified product and plan the activities within time schedules with CPM and PERT Analysis

CO2 For managing the quality of product and managing the risk involved

CO3 Managing team and measuring and tracking the planning

CO4 Configuration management and project monitoring and control

UNIT-I (11 Hrs.)

Project Planning: Characteristics of a software project, Software scope and feasibility, resources, the SPM plan.

Software Project Estimation: Size/scope estimation, Decomposition techniques, WBS. Effort estimation: Sizing, Function point, LOC, FP vs LOC. Schedule estimation: GANTT Charts, Activity networks, PERT/CPM networks. Cost estimation: Models: COCOMO I, COCOMO II.

UNIT-II (12 Hrs.)

Quality Planning: Quality control, Quality assurance, Formal Technical Reviews, The SQA Plan, ISO and CMM standards.

Risk Management: Reactive vs proactive Risk strategies, Risk projection, Risk Refinement, Risk Monitoring, Monitoring and management, RMMM plan.

UNIT-III (11 Hrs.)

Measurement and Tracking Planning: Earned Value Analysis.

Team Management: Team structures: hierarchical, Egoless, chief programmer, mixed; Team software Process; Resource levelling, Building a team: Skill sets.

UNIT-IV (11 Hrs.)

Configuration Management: Baselines, Configurable items, SCM repository, SCM process, version control change control, configuration audit**.**

Project Monitoring and Control: Audits and Reviews.

RECOMMENDED BOOKS:

1. Bob Hughes and Mike Cotterell, 'Software Project Management', 5th Edn., Tata McGraw Hill, **2009**.

2. Roger Pressman, ‘A practitioner’s Guide to Software Engineering’, 8th Edn., Tata McGraw Hill, **2014**.
3. ‘Head First PMP: A Brain Friendly Guide to Passing the Project Management Professional Exam’, **2013**.

ADVANCED OPERATING SYSTEM

Subject Code-MCSE1-161, MCSE4-162, MCSE2-161, MCSE3-161	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:
To learn the fundamentals of Operating Systems and gain knowledge on Distributed operating system concepts that includes architecture, Mutual exclusion algorithms, Deadlock detection algorithms and agreement protocols

LEARNING OUTCOMES:
CO1 Discuss the various synchronization, scheduling and memory management issues
CO2 Demonstrate the Mutual exclusion, Deadlock detection and agreement protocols of Distributed operating system
CO3 Discuss the various resource management techniques for distributed systems
CO4 Identify the different features of real time and mobile operating systems

UNIT-I (12 Hrs.)

Fundamentals of Operating Systems: Strategies of operating system, Structures of operating system, overview, Synchronization Mechanisms, Processes and Threads, Process Scheduling, Deadlocks: Detection, Prevention and Recovery, Models of Resources, Memory Management Techniques.
Distributed Operating Systems: Issues in Distributed Operating System, Architecture, Communication Primitives, Lamport’s Logical clocks, Causal Ordering of Messages – Distributed Mutual Exclusion Algorithms – Centralized and Distributed Deadlock Detection Algorithms, Agreement Protocols.

UNIT-II (11 Hrs.)

Distributed Resource Management: Distributed File Systems – Design Issues - Distributed Shared Memory – Algorithms for Implementing Distributed Shared memory–Issues in Load Distributing – Scheduling Algorithms, Synchronous and Asynchronous Check Pointing and Recovery, Fault Tolerance, Two-Phase Commit Protocol, Non-blocking Commit Protocol, Security and Protection.

UNIT-III (11 Hrs.)

Real Time and Mobile Operating Systems: Basic Model of Real Time Systems, Characteristics, Applications of Real Time Systems, Real Time Task Scheduling, Handling Resource Sharing, Mobile Operating Systems, Micro Kernel Design, Client Server Resource Access, Processes and Threads, Memory Management, File system, Networked file system.

UNIT-IV (11 Hrs.)

CASE STUDIES: Linux System: Design Principles, Kernel Modules, Process Management Scheduling, Memory Management, Input-Output Management, File System, Interprocess Communication. iOS and Android: Architecture and SDK Framework, Media Layer, Services Layer, Core OS Layer, File System.

RECOMMENDED BOOKS:
1. Andrew S. Tanenbaum and Maarten van Steen, ‘Distributed Systems: Principles and Paradigms’, 2nd Edn., Prentice Hall, **2007**.

2. Mukesh Singhal and Niranjan G. Shivaratri, ‘Advanced Concepts in Operating Systems – Distributed, ‘Database, and Multiprocessor Operating Systems’, Tata McGraw-Hill, **2001**.
3. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, ‘Operating System Concepts’, 7th Edn., John Wiley & Sons, **2004**.
4. Daniel P. Bovet and Marco Cesati, ‘Understanding the Linux Kernel’, 3rd Edn., O’Reilly, **2005**.
5. Rajib Mall, ‘Real-Time Systems: Theory and Practice’, Pearson Education India, **2006**.
6. Neil Smyth, ‘iPhone iOS 4 Development Essentials – Xcode’, 4th Edn., Payload Media, **2011**.

SOFTWARE ARCHITECTURE AND DESIGN PATTERNS

Subject Code: MCSE3-162	L T P C	Duration: 45 Hrs.
	3 1 0 4	

LEARNING OBJECTIVES:
This course deals with problem of identifying the concept of architecture and intends to reach the student architecture centric software development methodology. Other topics covered are Architecture Documentation, Architecture Evaluation, Product Lines Enterprise Architecture.
LEARNING OUTCOMES:
CO1: To understand the variety of implemented bad practices related to the Business for quality enhancements.
CO2: To understand the evolution of patterns for decision making.
CO3: To understand the concept of patterns and the Catalog.
CO4: To learn about specific behavioral patterns, idioms.

UNIT-I (11 Hrs.)

Envisioning Architecture: The Architecture Business Cycle, what is Software Architecture, Architectural patterns, reference models, reference architectures, architectural structures and views. Creating an Architecture Quality Attributes, achieving qualities, Architectural styles and patterns, designing the Architecture, Documenting software architectures, Reconstructing Software Architecture.

UNIT-II (12 Hrs.)

Software Architecture: Patterns in Software Architecture, Enabling Techniques for Software Architecture, Non-functional Properties of Software Architecture.
Analysing Architectures: Architecture Evaluation, Architecture design decision making, ATAM, CBAM. Moving from one system to many Software Product Lines, Building systems from off the shelf components, Software architecture in future.

UNIT-III (11 Hrs.)

Patterns: Pattern Description, organizing catalogues, role in solving design problems, Selection and usage. Creational and Structural patterns, Abstract factory, builder, factory method, prototype, singleton, adapter, bridge, composite, decorator, façade, flyweight, proxy.

UNIT-IV (11 Hrs.)

Behavioural Patterns: Chain of responsibility, command, Interpreter, iterator, mediator, memento, observer, state, strategy, template method, visitor.
Idioms: Introduction, What Can Idioms Provide, Idioms and Style, Where Can You Find Idioms, Counted Pointer.

RECOMMENDED BOOKS:
1. Len Bass, Paul Clements & Rick Kazman, ‘Software Architecture in Practice’, 2nd Edn., Pearson Education, **2003**.

2. Erich Gamma, 'Design Patterns', Pearson Education, **1995**.
3. Luke Hohmann, 'Beyond Software Architecture', Addison Wesley, **2003**.
4. David M. Dikel, David Kane and James R. Wilson, 'Software Architecture', Prentice Hall PTR, **2001**.
5. David Budgen, 'Software Design', 2nd Edn., Pearson Education, **2003**.
6. Eric Freeman and Elisabeth Freeman, 'Head First Design patterns, Eric Freeman & Elisabeth Freeman', O'reilly, **2007**.
7. Steven John Metsker and William C. Wake, 'Design Patterns in Java', Pearson Education, **2006**.
8. Deepak Alur, John Crupi and Dan Malks, 'J2EE Patterns', Pearson Education, **2003**.
9. Steven John metsker, 'Design Patterns in C#', Pearson Education, **2004.**
10. F. Buschmann and others, 'Pattern Oriented Software Architecture', John Wiley & Sons.

SOFTWARE MAINTENANCE AND MANAGEMENT

Subject Code: MCSE3-163	**L T P C**	**Duration: 45 Hrs.**
	3 1 0 4	

LEARNING OBJECTIVES:
To expertise in the SQA maintenance tools, gain knowledge of the overall project activities, To analyses the various issues in each phase of project management and people management.
LEARNING OUTCOMES:
CO1 Learn about measurement, benchmarking
CO2 Know about SQA maintenance tools and process models
CO3 Knowledge gained in usage and application of umbrella activities for project management. Execute the project development in a systematic manner using tools and techniques.
CO4 Issues are analysed in various phases of project management and people management

UNIT-I (11 Hrs.)
Maintenance Process: Software Maintenance- Customer's Viewpoint- Economics of Maintenance- Issues in Maintenance- Software Maintenance Standard, Process, Activities and Categories – Maintenance Measurement – Service Measurement and Benchmarking – Problem Resolution Reporting – Fix Distribution.

UNIT-II (12 Hrs.)
Activities for Maintenance and process Models: Role of SQA for Support and Maintenance, SQA tools for Maintenance, Configuration Management and Maintenance, Maintenance of Mission Critical Systems, Global Maintenance Teams, Foundation of S3m Process Model, Exemplary Practices. Product, Process and Project, Definition, Product Life Cycle, Project Life Cycle Models. Format model for a process, The ISO 9001 and CMM models and their relevance to project Management-other emerging models like People CMM.

UNIT-III (11 Hrs.)
Umbrella activities in projects and in stream activities in projects: Software Project Management -Formal Technical Reviews-Software Quality Assurance, Software Configuration Management, Re-usability Management-Risk analysis and Management - Measurement and Metrics-Document Preparation and Production. Project Initiation - Project Planning- feasibility study estimation- resource allocation- execution and tracking, root cause analysis- Project Wind-Up-Concept of process/project database.

UNIT-IV (11 Hrs.)
Engineering and People Issues in Project Management: Phases (Requirements, Design, Development, Testing, maintenance, deployment) – engineering activities and management

issues in each Phase-Difficulties in people management- Role of Project manager, Special considerations in project management for India and geographic distribution issues.

RECOMMENDED BOOKS:

1. Alian April and Alain Abran, 'Software Maintenance Management: Evaluation and Continuous Improvement', John Wiley & Sons Inc., **2008**.
2. Watts Humphrey, 'Managing the Software Process', Pearson Education, New Delhi, **2000**.
3. Gopalaswamy Ramesh and Ramesh Bhattiprolu, 'Software Maintenance: Effective Practices for Geographically Distributed Environments', 2nd Reprint, Tata McGraw Hill, **2009**.
4. Humphrey, Watts, Managing the software process, Addison Wesley, **1986**.
5. Pressman, Roger, 'Software Engineering: A Practitioner's Approach', McGraw Hill, **1997**.
6. Pankaj Jalote, 'Software Project Management in Practice', Pearson Education, New Delhi, **2002**.

PRACTICAL LAB.-I

Subject Code: MCSE3-104 **L T P C**
0 0 4 2

- Practical's should be related to the core subjects of the same semester.

AGILE SOFTWARE DEVELOPMENT APPROACHES

Subject Code: MCSE1-156 **L T P C** **Duration: 45 Hrs.**
MCSE2-156, **3 1 0 4**
MCSE4-156,
MCSE3-205

LEARNING OBJECTIVES:
This course makes student learn the fundamental principles and practices associated with each of the agile development methods. To apply the principles and practices of agile software development on a project of interest and relevance to the student.

LEARNING OUTCOMES:
CO1: To learn the basics concepts of Agile software and their principles design
CO2: To explain different agile development method, project tools requirement, risk and measurements related with different development methods.
CO3: To understand the overview of Agile methods, strategies, requirements and testing.
CO4: Describe and explain agile measurement, configuration and risk management. Principles of Astern and tools.

UNIT-I (11 Hrs.)

Introduction: Basics and Fundamentals of Agile Process Methods, Values of Agile, Principles of Agile, stakeholders, Challenges.

Agile and its Significance: Agile development, Classification of methods, the agile manifesto and principles, Practices of XP, Scrum Practices, working and need of Scrum, advanced Scrum Applications, Scrum and the Organization.

UNIT-II (12 Hrs.)

Agile Project Management: Embrace communication and feedback, Simple practices and project tools, Empirical Vs defined and prescriptive process – Principle-based versus Rule-Based – Sustainable discipline: The human touch – Team as a complex adaptive system – Agile hype – Specific agile methods. Quality, Risk, Metrics and Measurements, the facts of change on software projects – Key motivations for iterative development – Meeting the requirements challenge iteratively – Problems with the waterfall. Research evidence – Early historical project evidence – Standards-Body evidence, Expert and thought leader evidence – A Business case for iterative development – The historical accident of waterfall validity.

UNIT-III (11 Hrs.)

Agile Methodology: Method overview – Lifecycle – Work products, Roles and Practices values – Common mistakes and misunderstandings – Sample projects – Process mixtures – Adoption strategies – Fact versus fantasy – Strengths versus "Other" history.

Agile Requirements: User Stories, Backlog Management. Agile Architecture: Feature-Driven Development. Agile Risk Management: Risk and Quality Assurance, Agile Tools.

UNIT-IV (11 Hrs.)

Agile Testing: Agile Testing Techniques, Test-Driven Development, User Acceptance Test.

Agile Review: Agile Metrics and Measurements, The Agile approach to estimating and project variables, Agile Measurement, Agile Control: the 7 control parameters. Agile approach to Risk, The Agile approach to Configuration Management, The Atern Principles, Atern Philosophy, the rationale for using Atern, Refactoring, Continuous integration, Automated Build Tools.

RECOMMENDED BOOKS:

1. Elisabeth Hendrickson, 'Agile Testing', Quality Tree Software Inc., **2008**.
2. Craig Larman, 'Agile and Iterative Development – A Manager's Guide', 1st Edn., Pearson Education, **2004**.
3. Robert C. Martin, 'Agile Software Development, Principles, Patterns, and Practices (Alan Apt Series)', 2nd Edn., Pearson Education, **2003**.
4. Alistair Cockburn, 'Agile Software Development series', 1st Edn., Addison-Wesley Professional, **2001**.
5. Mike Cohn, 'Succeeding with Agile: Software Development Using Scrum', 1st Edn., Pearson, **2010**.

SOFTWARE TESTING & VALIDATIONS

Subject Code: MCSE3-206	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:

This course is designed to enable a clear understanding and knowledge of the foundations, techniques, and tools in the area of software testing and its practice in the industry. The course will prepare students to be leaders in software testing.

LEARNING OUTCOMES:

CO1: able to apply software testing knowledge, verification & validation and engineering methods.

CO2: Have an ability to design and conduct a software test process for a quality software test project.

CO3: Have an ability understand and identify various software testing problems, and solve these problems by designing and selecting software test models, criteria, strategies, and methods.

CO4: Have an ability to use software testing methods and modern software testing tools for their testing projects.

UNIT-I (11 Hrs.)

Review of Software Engineering: Overview of software evolution, design models, development life cycle, unit and system testing, project management, maintenance, Concept of Software verification, validation and testing.

V & V and their Limitations: Theoretical Foundations: Impracticality of Testing All data; Impracticality of testing All Paths; No Absolute Proof of Correctness.

UNIT-II (12 Hrs.)

The Role of V & V in Software Evolution: Types of Products, Requirements; Specifications, Designs, Implementation, Charges, V & V Objectives, Correctness, Consistency, Necessity Sufficiency, Performance.

Software Reliability and Quality Assurance: Software reliability, validation, safety and hazards analysis; features affecting quality of software. Concepts and importance of quality assurance, Software quality assurance strategies, FTR, structured walk through techniques.

UNIT-III (11 Hrs.)

Software V & V Approaches and their Applicability: Software Technical Reviews, Software Testing : Levels of testing, Module, Integration, System, Regression, Testing techniques and their Applicability, Functional testing and Analysis Structural testing and Analysis, Error Oriented testing and Analysis, Hybrid Approaches, Integration Strategies, Transaction Flow Analysis, Stress Analysis, Failure Analysis, Concurrency Analysis, Performance Analysis Proof of Correctness, Simulation and Prototyping, Requirements Tracing.

UNIT-IV (11 Hrs.)

Software V & V Planning, Identification and Selection techniques: requirements, Specifications, Designs, Implementations, Changes, Organizations Responsibilities, Development Organization Independent Test Organization, Software Quality Assurance, Independent V &V contractor, V & V Standards, Integrating V & V Approaches, Problem Tracking Test Activities, Assessment.

RECOMMENDED BOOKS:

1. William Perry, 'Effective Methods for Software Testing', John Wiley & Sons, **1995**.
2. Mare Roper, 'Software Testing', McGraw Hill Book Co., **1994**.
3. Cem Kaner, Jack Falk, Nguyen Quoc, 'Testing Computer Software', 2nd Edn., Van Nostrand Reinhold, **1993**.
4. Ron Patton, 'Software Testing', 2nd Edn., **2009.**
5. K.K. & Yogesh Singh, 'Software Engineering; Agricultural', New Age International, **2001**.
6. James Mc Manus I & Gordon Schulmeyer Van Nostrand Reinhold, 'Handbook of Software Quality Assurance', **1992.**
7. Ronald Owston, Van Nostrand Reinhold, 'Software System Testing and Quality Assurance', **1984**.
8. Michael Deutch Prentice Hall, 'Software Verification and Validation: Realistic Project Approach', **1982**.

INFORMATION RETRIEVAL

Subject Code: MCSE3-264 **L T P C** **Duration: 45 Hrs.**
3 1 0 4

LEARNING OBJECTIVES:
To learn the underlying technologies of modern information retrieval system.
LEARNING OUTCOMES:
CO1 Able to understand the basic concepts of modern information retrieval system.
CO2 Able to understand the search engine architecture.
CO3Able to learn the retrieval models and apply the algorithms of retrieval algorithms.
CO4 Able to evaluate the quality of retrieval system.

UNIT-I (12 Hrs.)

Introduction: The nature of unstructured and semi-structured text, Boolean queries, World Wide Web, History of Hypertext, Hypertext systems, Problems due to Uniform accessibility, types of Hypertext data, Text and multimedia data indexing, PageRank, HITS, XML and Semantic web.

UNIT-II (11 Hrs.)

Search Engine Architecture: the basic building blocks of a modern search engine system, including web crawler, basic text analysis techniques, inverted index, query processing, search result interface.

UNIT-III (11 Hrs.)

Retrieval Models: Boolean, vector space, probabilistic and language models, latent semantic indexing, ranking algorithm, Introduction to the most recent development of learning-based ranking algorithms, i.e., learning-to-rank, Relevance feedback, query expansion, link analysis and search applications.

UNIT-IV (11 Hrs.)

Performance Evaluation: Evaluating search engines, User happiness, precision, recall, F-measure.

RECOMMENDED BOOKS:
1. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze, 'Introduction to Information Retrieval', 1st Edn., Cambridge University Press, **2008**.
2. Bruce Croft, Donald Metzler, and Trevor Strohman, 'Search Engines: Information Retrieval in Practice', 1st Edn., Pearson Education, **2009**.
3. Yates Ricardo and Berthier Ribeiro-Neto, 'Modern Information Retrieval', 2nd Edn., Addison-Wesley, **2011**.
4. Soumen Chakrabarti, 'Mining the Web', 1st Edn., Morgan-Kaufmann, **2002**.

SOFTWARE QULAITY MANAGEMENT

Subject Code: MCSE3-265 **L T P C** **Duration: 45 Hrs.**
3 1 0 4

LEARNING OBJECTIVES:
The objective of the course is to understand the concepts of software quality, various standards, SQA, detailed design and testing as well as various CASE tools.
LEARNING OUTCOMES:
CO1: To understand the software quality and learning the concept of SQA.
CO2: Understanding the Quality assurance and management
CO3: To learn about software design, testing, defects and error frequency.

CO4: Understanding different CASE tools and quality management standards.

UNIT-I (11 Hrs.)

Introduction to software quality, different quality attributes software quality control and software quality assurance, concept of SQA, SQA activities and SQA issues, zero defect software, Functions of software QA organization in a project.

UNIT-II (11 Hrs.)

Quality assurance and software life cycle, SQA techniques, Tailoring the Software Quality Assurance Program: Management review process, technical review process, walkthrough, software inspection process, configuration audits and document verification.

UNIT-III (12 Hrs.)

Software requirements, preliminary design, detailed design, coding and unit test, integration and testing, system testing, Identification of defect, analysis and correction of defect, implementation of correction, regression testing, Categorization of defect, Error quantity, error frequency, program unit complexity, compilation frequency

UNIT-IV (11 Hrs.)

CASE tools and their effect on Software Quality, Software Quality Metrics, Standards, certification and assessment, Quality management standards, Quality standards with emphasis on ISO approach, Capability Maturity Models-CMM and CMMI, TQM Models, Bootstrap methodology, The SPICE project, ISO/IEC 15504, Six Sigma Concept for Software Quality.

RECOMMENDED BOOKS:

1. Roger Pressman, 'Software Engineering - A Practitioners Approach', McGraw Hill, **2009**.
2. Ian Sommerville, 'Software Engineering', Addison-Wesley Publishing Company, **2006**.
3. James F. Peter, 'Software Engineering - An Engineering Approach', John Willey
4. Pankaj Jalote, 'An Integrated Approach to Software Engineering', Narosa Publishing House.

UNIFIED SOFTWARE CONFIGURATION MANAGEMENT

Subject Code: MCSE3-266	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:

LEARNING OUTCOMES:

CO1: Understanding basic SCM concepts and the evolution of SCM tools. Provides an overview of the Unified Change Management mode including today's most effective usage models, strategies, and policy configurations

CO2 Illustrates Clear Case objects, including the repository, workspaces, and component management and how to create SCM environment.

CO3 Learn how to deal with a variety of real-world development scenario such as multiple projects, and how to coordinate parallel work both within a project and between multiple projects.

CO4 Analyze the process of integration, building and releasing of software.

UNIT-I (11 Hrs.)

Software Configuration Management: SCM best practices, SCM tools and process, Dyeing with changing project requirements.

Overview of the Unified Change Management Model: UCM, Clear Case, UCM process overview, defining the Implementation Model, The UCM baseline and Change Model.

UNIT-II (12 Hrs.)

Functional Overview of Objects: The Repository, Versioned Object Base, Workspaces, Component Management, Process, Building, Clear make, Derived Objects, Configuration records.

Establishing the Initial SCM Environment: Clear Case Architecture Basics, Defining the Implementation Model, Creating the VOBs, Baseline promotion levels Project Management in Clear Case.

UNIT-III (11 Hrs.)

Coordinating Multiple Project Teams and Other Scenarios: Organizing large Multi project development efforts, coordinating cooperating projects, Independent components, shared components, Multiple Parallel release, Using UCM without Activity-based SCM.

Development Using the UCM Model: A Developer's perspective of UCM, joining a project, making changes, delivering changes to the project, rebasing your development stream, Dealing with conflicting changes.

UNIT-IV (11 Hrs.)

Integration, Build and Release: Software Integration, Isolation and integration, Building and Base lining, Staging and release.

RECOMMMENDED BOOKS:

1. G. Booch, I. Jacobson, J. Rumbaugh, 'The Unified Software Development Process', 1st Edn., Addison Wesley, **1999**.
2. Brian A. White**, '**Software Configuration Management Strategies and Rational Clear Case', 4th Edn., Addison Wesley, **2001**.
3. Roger S. Pressman, 'Software Engineering a Practitioners Approach', 5th Edn., McGraw-Hill, **2014**.
4. James Rumbaugh, Ivar Jacobson and Grady Booch, 'The Unified Modelling Language Reference Manual', 2nd Edn., Addison Wesley, **2004**.

SOFTWARE DESIGN AND CONSTRUCTION

Subject Code: MCSE3-267	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:

This course offers a good understanding of the concepts, methods and techniques of software designing and construction and prepares students to be in a position to develop software.

LEARNING OUTCOMES:

CO1: To explain the general fundamental design fundamentals. It explains the nature of Design process, Design qualities, object Model.

CO2: To understand the structured system analysis and design.

CO3: To Know the object oriented and design.

CO4: To Explain the software Architecture and design methods.

UNIT-I (11 Hrs.)

General Design Fundamentals: The nature of Design process, Objectives, Building Models, Constructs, Design qualities, Assessing the design, Design viewpoints for software, The object Model, Classes and Objects, Complexity, Classification, Notation, Process, Pragmatics.

UNIT-II (12 Hrs.)

Structured System Analysis and Design: Structured Design, Design Principle, Problem Partitioning and Hierarchy, Abstraction, Modularity, Top-down and Bottom-up Strategies, Transformation of a DFD to a Structure Chart, Transform Analysis, Transaction Analysis, Coupling, Cohesion, Multiple types of Cohesion in a module, Data Design, Normalization, Denormalization, Procedural Design.

UNIT-III (11 Hrs.)

Object Oriented Analysis and Design: Overview of Object Oriented Analysis, Shaler/Mellor, Coad/ Yourdon, Rumbaugh, Booch UML Use case, Conceptual model, Behaviour, Class Analysis Patterns, Overview, Diagrams, Aggregation, UML, Diagrams, Collaboration, Sequence, Class, Design patterns and Frameworks, Comparison with other design methods, Managing analysis and design, Evaluation testing, Coding, Maintenance, Metrics.

UNIT-IV (11 Hrs.)

Software Design: The Architecture Concepts, Design Methods Design Patterns, Rationale for Methods, Design Processes and Strategies, Design by Template, Designing with Patterns, Stepwise Refinement, Incremental Design, Prototyping, DSDM, Structured Systems Analysis and Structured Design, JSP, JSD. Domain Name System, Email, World Wide Web (HTTP), Simple Network Management Protocol, File Transfer Protocol, Security, Multimedia applications.

RECOMMENDED BOOKS:

1. David Budgen, ‘Software Design’, 2nd Edn., Pearson Education, **2004**.
2. R.S. Pressman, ‘Software Engineering’, 5th Edn., McGraw Hill Inc., **2001**.
3. Ed Downs, Peter Clare, Jan Coe, ‘Structured System Analysis and Design Methods Application and Context’, 2nd Edn., Prentice Hall, **1992**.
4. G. Suteliffe, ‘Human Computer Interface Design’, 2nd Edn., Macmillan, **1995**.

SECURE SOFTWARE ENGINEERING

Subject Code: MCSE3-268 **L T P C 3 1 0 4** **Duration: 45 Hrs.**

LEARNING OBJECTIVES:

To help students to know the significance, architecture and design, requirement engineering, quality, management and techniques of security for software systems

LEARNING OUTCOMES:

CO1: To know the basics of secure software engineering
CO2: To study various issues related to requirement specification, architecture and design of secure systems
CO3: To understand the relation of quality with security and its implementation
CO4: To study various problems and their management for secure systems

UNIT-I (11 Hrs.)

Security Concepts: Introduction, the need for software security, system security, difference between software and system security, software assurance, and its relation to software security.

Trust and threat for Security: Threats to software security, sources of software insecurity, Benefits of Detecting Software Security, Properties of Secure Software, Influencing the security properties of software, trust and threat model foe software security.

UNIT-II (12 Hrs.)

Security Requirements for Secure Software: Introduction, Security requirements, Requirement Gathering, Requirement specification, Identify Assets, Risk Management, Requirements elicitation and prioritization, the SQUARE process Model

Secure Software Architecture and Design: Introduction, architecture for software security, architectural risk analysis, software security knowledge for architecture and design, security principles, security guidelines.

UNIT-III (11 Hrs.)

Quality: Code analysis, Software Security testing, Security testing considerations throughput the SDLC, relation between quality and security.

UNIT-IV (11 Hrs.)

Security and Complexity: System Assembly Challenges: introduction, security failures, functional and attacker perspectives for security analysis, system complexity drivers and security.

Managing for More Secure Software: Governance and security, adopting an enterprise software security framework, how much security is enough?

RECOMMENDED BOOKS:

1. Julia H. Allen, 'Software Security Engineering', 1st Edn., Addison Wesley, **2008**.
2. Jason Grembi, 'Developing Secure Software', 1st Edn., Cengage Learning, **2009**.
3. Richard Sinn, 'Software Security', 1st Edn., Cengage Learning, **2009**.

SOFTWARE PROJECT METRICS

Subject Code: MCSE3-269 **L T P C** **Duration: 45 Hrs.**
3 1 0 4

LEARNING OBJECTIVES:

To teach students software metrics and about issues related to software metrices.

To provide students with skills needed to do independent research in the project metrices.

LEARNING OUTCOMES:

CO1 To understand the basics of Software Measurement, its underlying objectivity using the quantifiable approach in order to control, manage and Improvise quality.

CO2 Measuring internal and external product attributes.

CO3 To analyse characteristics of object oriented metrics and dynamic metrics.

CO4 Be able to define metrices for component based system and measuring productivity of resources.

UNIT-I (11 Hrs.)

Basics of Measurement: Measurement in everyday life, measurement in software engineering, scope of software metrics, representational theory of measurement, measurement and models, measurement scales, meaningfulness in measurement, goal-based framework for software measurement, classifying software measures, determining what to measure, software measurement validation, empirical investigation, types of investigation, planning and conducting investigations.

Software-Metrics Data Collection and Analysis: What is good data, how to define the data, how to collect the data, how to store and extract data, analyzing software-measurement data, frequency distributions, various statistical techniques.

UNIT-II (12 Hrs.)

Measuring Internal Product Attributes: Measuring size, aspects of software size, length, functionality and complexity, measuring structure, types of structural measures, control-flow structure, modularity and information flow attributes, data structures.

Measuring External Product Attributes: Modeling software quality, measuring aspects of software quality, software reliability, basics of software reliability, software reliability problem, parametric reliability growth models, predictive accuracy, recalibration of software-reliability growth predictions, importance of operational environment, and wider aspects of software reliability.

UNIT-III (11 Hrs.)

Metrics for Object-Oriented Systems: Intent and characteristics of object-oriented metrics, various object-oriented metric suites LK suite, CK suite and MOOD metrics.

Dynamic Metrics: Runtime Software Metrics, Extent of Class Usage, Dynamic Coupling, Dynamic Cohesion, and Data Structure Metrics.

UNIT-IV (11 Hrs.)

Metrics for Component-based Systems: The intent of component-based metrics, distinguishing characteristics of component-based metrics, various component-based metrics.

Resource Measurement: Measuring productivity, teams, tools, and methods.

RECOMMENDED BOOKS:

1. Norman E-Fenton and Shari Lawrence Pfleeger, Software Metrics, 2nd Edn., International Thomson Computer Press, **1997**.
2. Norman Fenton and James Bieman, Software metrics: a rigorous and practical approach, 3rd Edn., CRC Press, **2014**.
3. Stephen H. Kan, Metric and Models in software Quality Engineering, 2nd Edn., Addison Wesley, **1995**.
4. C. Ravindranath Pandian, Software metrics: A guide to planning, analysis and application, 1st Edn., CRC Press, **2003**.

SOFTWARE RELIABILITY AND METRICES

Subject Code: MCSE3-270	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:

This course offers a good understanding of the concepts, methods and techniques for testing software reliability and provide software metrics and prepares students to be in a position to develop error free and good quality software.

LEARNING OUTCOMES:

CO1: Able to explain the need and basic concepts of software reliability. Able to explain the different software reliability models.

CO2: Able to explain the software prediction analysis. It explains the short and long term predictions, Analyzing Predictive Accuracy, Errors and Inaccuracy.

CO3: Able to explain the software reliability testing techniques.

CO4: Able to understand the measuring internal product attributes like software size, length, functionality and complexity etc. and the measuring external product attributes measuring aspects of software quality, parametric reliability growth models etc.

UNIT-I (12 Hrs.)

Introduction: Need and Concepts of Software Reliability, Failure and Faults – Prevention, Removal, Tolerance, Forecast, Dependability Concept– Failure Behavior, Characteristics, Maintenance Policy, Reliability and Availability Modeling. Software reliability models: Historical Perspective and Implementation, classification, limitations and issues, Exponential Failure Models – Jelinski moranda model, Poisson, Musa, Exponential models, Weibull Model, Musa okumoto Model, Bayseian Model – Littlewood verral Model.

UNIT-II (11 Hrs.)

Prediction Analysis: Model Disagreement and Inaccuracy – Short & Long Term Prediction, Model Accuracy, Analyzing Predictive Accuracy – Outcomes, PLR, U & Y Plot, Errors and

Inaccuracy, Recalibration – Detecting Bias, Techniques, Power of Recalibration, Limitations in Present Techniques, Improvements.

UNIT-III (11 Hrs.)

Testing for Reliability Measurement: Software Testing – Types, White and Black Box, Operational Profiles – Difficulties, Estimating Reliability, Time/Structure based software reliability – Assumptions, Testing methods, Limits, Starvation, Coverage, Filtering, Microscopic Model of Software Risk.

UNIT-IV (11 Hrs.)

Measuring Internal and External Product Attributes: Measuring size, aspects of software size, length, functionality and complexity, measuring structure, types of structural measures, control-flow structure, modularity and information flow attributes, data structures. Modeling software quality, measuring aspects of software quality, software reliability, basics of software reliability, software reliability problem, parametric reliability growth models, predictive accuracy, importance of operational environment.

RECOMMENDED BOOKS:

1. Patric D.T.O Connor, 'Practical Reliability Engineering', 4th Edn., John Wesley & Sons, **2003**.
2. John D. Musa, 'Software Reliability Engineering', 1st Edn., Tata McGraw Hill, **1999**.
3. Michael R. Lyu, 'Handbook of Software Reliability Engineering', 1st Edn., IEEE Computer Society Press, **1996**.
4. Norman E-Fenton and Shari Lawrence Pfleeger, 'Software Metrics', 2nd Edn., International Thomson Computer Press, **1997**.
5. Norman Fenton and James Bieman, 'Software Metrics: A Rigorous and Practical Approach', 3rd Edn., CRC Press, **2014**.

OPEN SOURCE TECHNOLOGY

Subject Code: MCSE3-271	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES:

To give a brief introduction to the open source technology. Through interactive sessions enabling students to enhance their skills in contributing and implementing their technical knowledge.

LEARNING OUTCOMES:

CO1: Open source software history, initiatives and principles. Open standards, Licenses and FOSS.

CO2: Learn about the Open Source Operating system and its distributions like Fedora, Google chrome OS, Ubuntu.

CO3: Study of Web technologies based on open Software's LAMP (Linux Apache MySqland PHP/Python)

CO4: To Learn HTML, XHTML, PHP and JavaScript

UNIT-I (11 Hrs.)

Introduction: Open Source Definition, Free Software vs. Open Source Software, Public Domain Software, Open Source History, Initiatives, Principle and Methodologies. Open Standards.

Open Source Development Model Licenses and Patents: What Is a License, Important FOSS Licenses (Apache, BSD, GPL, LGPL), copyrights and copy lefts, Patents Economics of FOSS: Zero Marginal Cost, Income-generation opportunities, Problems with traditional commercial software, Internationalization.

UNIT-II (12 Hrs.)

Open Source Operating Systems: Different open source operating systems. Google Chrome OS, BSD, Linux Distributions – Fedora and Ubuntu, Installation, Disk Partitioning, Boot loader. Using Linux – Shell, File system familiarity, Linux Administration – Managing users, services and software, Network Connectivity, Configurations and Security.

Open Source Web Technologies: Two Tier and Three Tier Web based Application Architecture. LAMP Terminologies, Advantages. Apache, Web server conceptual working, Web browser, HTTP, Installation and Configuration, httpd.conf file, Logging, Security, Running a website, MySQL, Database management system, ER diagram, Relational database, Installation, Configuration, Administration, Common SQL queries.

UNIT-III (11 Hrs.)

Programming on XHTML and XML: Editing XHTML, W3C XHTML validation services, designing XHTML by using XHTML tables, frames, forms and other elements. CSS and its types. XML, XML namespaces, DTD, XML schema, XML vocabularies, DOM and its methods, SOAP.

UNIT-IV (11 Hrs.)

Programming on PHP and JavaScript: JavaScript: JavaScript variables, control structures, functions, arrays and objects. Cascading Style Sheets, Client Side Scripting - Java Script, PHP: Form processing and business logic, stream processing and regular expressions, viewing client/server environment variables, connecting to database and handling of cookies. SQL, Accessing databases with PHP.

Open Source Ethics: Open source vs. closed source Open source government, Open source ethics. Social and Financial impacts of open source technology, shared software, Shared source.

Case Studies: Mozilla (Firefox), Wikipedia, Joomla, Open Office, GCC.

RECOMMENDED BOOKS:

1. B. Ware, B Lee J., 'Open Source Development with Lamp: Using Linux, Apache, MySQL, Perl, and PHP', Addison-Wesley Professional, **2002**.
2. Deitel, 'Internet and World wide web: How to program', 4th Edn., Prentice Hall, **2008**.
3. P. DuBois, 'MySQL', 5th Edn., Addison-Wesley Professional, **2013**.
4. M. Zandstra, 'Teach Yourself PHP in 24 Hours', 3rd Edn., Sams Publishing, **2003**.

PRACTICAL LAB.-II

Subject Code: MCSE3-207 **L T P C**
0 0 4 2

- Practical's should be related to the core subjects of the same semester.

COMPONENT BASED DEVELOPMENT

Subject Code: MCSE3-372 **L T P C** **Duration: 45 Hrs.**
3 1 0 4

LEARNING OBJECTIVES:

The objective of this course is to gain the knowledge of current component models in terms of their design, management and related issues. The students will be able to assess that how these models measure up to the goals of CBD.

LEARNING OUTCOMES:

CO1: Able to explain the Fundamentals of Component based Software Development, their Software Engineering Model, Success Factors and Common High Risk Mistakes.

CO2: Acquire the knowledge about various Software Engineering Practices and techniques for defining component Infrastructure.

CO3: Acquire the knowledge about management of Component based Software Systems, their various measurement metrics, processes and phases.

CO4: Acquire the knowledge about various Component Technologies and their comparisons.

UNIT-I (11 Hrs.)

Component Definition: Definition of Software Component and its Elements. Component Models and Component Services: Concepts and Principles, COTS Myths in Component-Based Software Development, Roles for Component-Based Development, Objectives of CBSE, Component based Software Engineering Processes, Component based software life cycle model, Common High Risk Mistakes in Component-Based Software Engineering, CBSE Success Factors: Integrating Architecture, Process, and Organization.

UNIT-II (12 Hrs.)

Software Engineering Practices: The Practice of Software Engineering, From Subroutines to Subsystems: Component-Based Software Development.

The Design of Software Component Infrastructures: Software Components and the UML, Component Infrastructures: Placing Software Components in Context, Business Components, Components and Connectors: Catalysis Techniques for Defining Component Infrastructures, An Open Process for Component-Based Development, Designing Models of Modularity and Integration.

UNIT-III (11 Hrs.)

The Management of Component-Based Software Systems: Measurement and Metrics for Software Components, The Practical Reuse of Software Components, Selecting the Right COTS Software: Why Requirements are Important, Software Component Project Management Processes, The Trouble with Testing Software Components, configuration Management and Component Libraries, The Evolution, Maintenance and Management of Component-Based Systems.

UNIT-IV (11 Hrs.)

Component Technologies: Overview of the CORBA Component Model, Transactional COM+: Designing Scalable Applications, The Enterprise JavaBeans Component Model, Choosing Between COM+, EJB, and CCM, Software Agents as Next Generation Software Components, Future of CBSE.

RECOMMENDED BOOKS:

1. Katharine Whitehead, 'Component-Based Development: Principles and Planning for Business Systems', Addison Wilsey, **2010.**
2. Don Box, Dorling Kingsley, 'Essential COM', 1st Edn., Addison-Wesley Professional, **2006**.

OBJECT ORIENTED ENGINEERING USING UML

Subject Code: MCSE3-373	**L T P C** **3 1 0 4**	**Duration: 45 Hrs.**

LEARNING OBJECTIVES: Object-Oriented Software Development is an approach/paradigm of developing software by identifying and implementing a set of objects and their interactions to meet the desired objectives. The first step towards this kind of software development is to learn and master the various concepts, tools and techniques that are to be used design and implementation of such systems.

LEARNING OUTCOMES:

CO1: Understanding the history and goals of UML.
CO2: Use of functional, non-functional requirements along with Use Case Modeling.
CO3: Knowledge of Modeling Classes and Dependencies.
CO4: Ability to know interfaces, components and Sequence Diagrams.

UNIT-I (11 Hrs.)

UML: History of UML, Goals of UML, nature & purpose of models, UML views & diagrams – static, design, use case, state machine, activity, interaction deployment, model management, profile; relationships in UML – association, dependency, generalization, realization; UML extensibility mechanisms – constraints, stereotypes, tagged values. Unified Process (UP): UP structure, phases of UP.

UNIT–II (12 Hrs.)

Requirements: Meta Model, Workflow, Functional and Non-functional Requirements; Requirement Attributes, Finding Requirements.

Use Case Modeling: Finding Actors and Use Cases, Use Case Scenario – main flow, branching within a flow, repletion within a flow, modeling alternative flows; relationships among actors and use cases; use case diagrams.

UNIT–III (11 Hrs.)

Analysis: Meta Model, Workflows, Finding Analysis Classes – using noun/verb analysis, CRC analysis, using RUP stereotypes - entity, boundary and control; Modeling Classes – Association (role name, multiplicity, navigability, association classes, qualified association) dependencies (usage, abstraction, permission), class generalization, generalization sets, power types; Analysis Package – nested packages, dependencies, transitivity, package generalization, architectural analysis, finding analysis packages; Concepts of Patterns & Frameworks.

Use Case Realization – interaction diagram, sequence diagram; Activity Diagrams.

UNIT–IV (11 Hrs.)

Design: Meta Model, Workflow, design classes – well-formed design classes, inheritance, templates, nested classes, design relationships, aggregation and composition, refining analysis relationships; interfaces and components – provided and required interfaces, interface realization v/s interface, components, finding interfaces, designing with interfaces; interaction diagram in design, modelling concurrency, active classes, concurrency in sequence diagram, concurrency in communication diagram; state machine - state machine diagrams, Implementation: Meta model, workflow, deployment diagram.

RECOMMENDED BOOKS:

1. Jim Arlow, Ila Neustadt, 'UML 2 and the Unified Process – Practical Object Oriented Analysis and Design', 2nd Edn., Pearson Education, **2005**.
2. Bernd Bruegge, Allen H. Dutoit, 'Object Oriented Software Engineering using UML', 3rd Edn., Pearson Education, **2009**.
3. Rumbaugh J., Jacobson I., Booch G., 'The Unifed Modeling Language Reference Manual', 2nd Edn., Pearson Education, **2005**.
4. Blaha M., Rumbaugh J., 'Object-Oriented Modeling and Design with UML', 2nd Edn., Pearson Education, **2005**.
5. Timothy C. Lethbridge, Robert Laganiere, 'Object Oriented Software Engineering', 1st Edn., Tata McGraw-Hill, **2008.**
6. Satzinger, Jackson, Burd, 'Object-Oriented Analysis & Design with the Unified Process', PKD Edn., Course Technology Inc., **2004.**

SOFTWARE RELIABILTY & METRICS

Subject Code MCSE3-374 **L T P C** **Duration: 45 Hrs.**
3 1 0 4

LEARNING OBJECTIVES: This course is a step by step description of the software metrics. It includes introduction to foundations of measurement theory, models of software engineering measurement, software products metrics, software process metrics and measuring management

LEARNING OUTCOMES:

CO1: To appreciate and understand scientific concepts of Software and Hardware Reliability.
CO2: To apply Software Reliability Growth Models in Software Development.
CO3: To emphasize the Application of Software Reliability Models.
CO4: Knowledge of Different Software Testing Models.

UNIT-I (11 Hrs.)

Need and Concepts of Software Reliability, Failure and Faults – Prevention, Removal, Tolerance, Forecast, Dependability Concept – Failure Behavior, Characteristics, Maintenance Policy, Reliability and Availability Modeling, Reliability Evaluation

UNIT-II (12 Hrs.)

Introduction - Historical Perspective and Implementation, classification, limitations and issues, Exponential Failure Models – Jelinski-moranda model, Poisson, Musa, Exponential models, Weibull Model, Musa-okumoto Model, Bayseian Model – Littlewood verral Model, Phase Based Model.

UNIT-III (11 Hrs.)

Model Disagreement and Inaccuracy – Short & Long Term Prediction, Model Accuracy, Analyzing Predictive Accuracy – Outcomes, PLR, U & Y Plot, Errors and Inaccuracy, Recalibration – Detecting Bias, Techniques, Power of Recalibration, Limitations in Present Techniques, Improvements.

UNIT-IV (11 Hrs.)

Concepts and Development Procedures – Customer Type, User Type, System Mode, Functional and Operational Profile, Test Selection - Selecting Operations, Regression Test, Special Issues – Indirect Input Variables, Updating, Distributed system, CASE STUDY - Application of DEFINITY & FASTAR, Power Quality Resource System.

Software Metrics: Introduction to Metrics, static and dynamic metrics, Design metrics, coding metrics, testing metrics and reliability metrics

RECOMMNEDED BOOKS:

1. Patric D.T.O Connor, 'Practical Reliability Engineering', 4th Edn., John Wesley & sons, **2003**.
2. John D. Musa, 'Software Reliability Engineering', 1st Edn., Tata McGraw Hill, **1999**.
3. Michael Lyu, 'Handbook of Software Reliability Engineering', IEEE Computer Society Press, **1996**.

USABILITY ENGINEERING

Subject Code: MCSE3-375 **L T P C** **Duration: 45 Hrs.**
3 1 0 4

LEARNING OBJECTIVES:
To introduce the need for human-computer-interaction study or human-centered software design, usability engineering lifecycle for designing a user-friendly software, information, interaction and GUI design process for enhancing user-experience and to develop usability evaluation skills for software testing.

LEARNING OUTCOMES:
CO1: Justify the need to study human-computer-interaction or human-factors while designing software.
CO2: Discuss the process of designing user-friendly software based on usability engineering guidelines.
CO3: Apply interaction design and UI design process in enhancing user-experience of an application.
CO4: Conduct usability evaluation of user-interfaces or software applications.

UNIT-I (11 Hrs.)

HCI AND USABILITY: What is HCI design? Disciplines contributing to HCI, Psychology of everyday things, Importance of human factors in design, Need Satisfaction curve of technology, Levels of human computer interaction What is Usability? benefits and cost savings, usability slogans, attributes of system acceptability, definition of usability, usability trade-Offs , categories of users and individual user differences, generations of user interfaces, scenario-based usability engineering case study - A Virtual Science Fair.

UNIT – II (12 Hrs.)

THE USABILITY ENGINEERING LIFECYCLE: User research and requirements analysis, know the user, user-profile questionnaire, field-study methods, contextual inquiry and analysis, hierarchical task analysis, ethnography, cultural probe, affinity diagramming, persona, scenarios of use, use cases. Iterative Design, setting usability criteria or goals, participatory design (getting users involved), guidelines and heuristic evaluation, prototyping and scenarios, examples of problem scenarios, iterative design, interface evaluation, meta methods. Usability Heuristics, simple and natural dialogue, speak the users' language, minimize user memory load, consistency, feedback, clearly marked exits, shortcuts, good error messages, prevent errors, help and documentation, heuristic evaluation.

UNIT–III (11 Hrs.)

INFORMATION DESIGN AND INTERACTION DESIGN: Information design, Information architecture concepts, stages of action in human-computer interaction, perceiving information, interpreting information, making sense of information. Interaction Design, selecting system goal, planning action sequence, executing action sequence, case study of information and interaction design User Interface Design, Goals of UID, User Interface Models, conceptual model and mock-ups of GUI, choosing prototyping alternatives - paper prototyping, rapid prototyping, storyboarding, wireframes, Cost/benefit of good interface design, Case Study.

UNIT–IV (11 Hrs.)

USABILITY EVALUATION: Developing usability specifications for evaluation - case study, criteria for user feedback techniques, formative and summative techniques of evaluation Usability Inspections (testing without users), heuristic evaluation, user-interface guideline reviews, cognitive walkthrough, model-based analysis Usability Testing (testing with users), developing usability or test specifications with case study , test goals and test plans , getting test users, choosing experimenters, ethical aspects of tests with human subjects, test tasks, stages of a test, performance measurement, thinking-aloud testing, usability laboratories, remote evaluation, Methods beyond testing, observation, user satisfaction questionnaire (rating scale), interviews, system usability scale (SUS), focus groups, logging actual use, user feedback, choosing a methods.

RECOMMENDED BOOKS:

1. J. Nielsen, 'Usability Engineering', 1st Edn., Elsevier, **1994**.
2. M.B. Rosson, & J.M. Carroll, 'Usability Engineering: Scenario-Based Development of Human-Computer Interaction', 1st Edn., Elsevier, **2001**.
3. D. Mayhew, 'The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design', 1st Edn., Morgan Kaufmann, **1999**.
4. A. Cooper et. al., 'The Essentials of Interaction Design',4th Edn., Wiley, **2007.**
5. B. Schneiderman, 'Designing the User Interface', 6th Edn., Pearson Education, **2005**.
6. A. Dix et. al., 'Human - Computer Interaction',3rd Edn., Prentice Hall, **1993**.
7. T. Mandel, 'Elements of User Interface Design', 2nd Edn., John Wiley & Sons, **2007.**
8. Rogers et. Al., 'Interaction Design', 3rd Edn., John Wiley & Sons, **2011**.
9. D. Norman, 'The Design of Everyday Things', Revised and expanded Edition, Basic Books, **2013**.
10. Donna Spencer, 'A Practical Guide to Information Architecture', Five Simple Steps, **2011**.
11. W. Galitz, 'The Essential Guide to User Interface Design', 3rd revised Edn., Wiley, **2002.**