Babu Banarasi Das University, Lucknow

Department of Computer Science & Engineering

School of Engineering

Master of Technology (Software Engineering)-Regular

Evaluation Scheme

SEMESTER I									
Course Category	**Course Code**	**Code Title**	**Contact Hours**			**Evaluation Scheme**			**Credits**
			L	**T**	**P**	**CIA**	**ESE**	**Course Total**	
C	MCS3111	Advance Software Engineering Principles & Practices	4	0	0	40	60	**100**	**4**
C	MCS3112	Object Oriented Software Engineering & UML	4	0	0	40	60	**100**	**4**
C	MAS3001	Probability and Statistical Analysis	4	0	0	40	60	**100**	**4**
C	MCS3103	Fundamentals of Theoretical Computer Science	4	0	0	40	60	**100**	**4**
GE		**Generic Elective I**	4	0	0	40	60	**100**	**4**
C	MCS3151	Object Oriented Modeling & Design Lab	0	0	2	40	60	**100**	**1**
C	MCS3152	Operating Systems Lab (Unix/Linux)	0	0	2	40	60	**100**	**1**
C	MCS3153	Seminar	0	0	2	100	0	**100**	**1**
Total			**20**	**0**	**6**	**380**	**420**	**800**	**23**

SEMESTER II									
Course Category	**Course Code**	**Code Title**	**Contact Hours**			**Evaluation Scheme**			**Credits**
			L	**T**	**P**	**CIA**	**ESE**	**Course Total**	
C	MCS3211	Software Design and Testing	4	0	0	40	60	**100**	**4**
C	MCS3212	Web Semantics	4	0	0	40	60	**100**	**4**
C	MCS3213	Software Project Management	4	0	0	40	60	**100**	**4**

GE		**Generic Elective II**	4	0	0	40	60	**100**	**4**
GE		**Generic Elective III**	4	0	0	40	60	**100**	**4**
C	MCS3251	Software Design and Testing Lab.	0	0	2	40	60	**100**	**1**
C	MCS3252	Web Semantic Lab	0	0	2	40	60	**100**	**1**
C	MCS3253	Seminar	0	0	2	100	0	**100**	**1**
Total			**20**	**0**	**6**	**380**	**420**	**800**	**23**

SEMESTER III									
Course Category	**Course Code**	**Code Title**	**Contact Hours**			**Evaluation Scheme**			**Credits**
			L	**T**	**P**	**CIA**	**ESE**	**Course Total**	
C	MCS3351	State of the art seminar[#]	-	-	-	200	-	**200**	**4**
C	MCS3352	Thesis I[*]	-	-	-	400	-	**400**	**16**
Total			**-**	**-**	**-**	**600**	**-**	**600**	**20**

SEMESTER IV									
Course Category	**Course Code**	**Code Title**	**Contact Hours**			**Evaluation Scheme**			**Credits**
			L	**T**	**P**	**CIA**	**ESE**	**Course Total**	
C	MCS3451	Thesis II[**]	-	-	-	200	800	**1000**	**28**
Total			**-**	**-**	**-**	**200**	**800**	**1000**	**28**

[#]**SOTA:** The student need to perform a literature survey, will give presentation on state of art topics and will submit a synopsis already mentioned in the problem statement. This will be evaluated internally within two months of the start of the semester and result will be intimated to the students, so to precede for Thesis I.

[*]The student will develop a workable model for the problem they have proposed in the synopsis

[**]This is in continuation with Thesis I. The required experimental/ mathematical verification of the proposed model will be done in this semester.

Legends:

L Number of Lecture Hours per week

T Number of Tutorial Hours per week
P Number of Practical Hours per week
CIA Continuous Internal Assessment
ESE End Semester Examination

Category of Courses:
F Foundation Course
C Core Course
GE Generic Elective
OE Open Elective

Course Code	Generic Elective I
GE34911	Fundamentals of Operating System and DBMS (Compulsory for students of Non-CSE background)
GE34912	Software Agents
GE34913	Information Architecture
GE34914	Real Time and Concurrent Systems
GE34915	IT Infrastructure Management

Course Code	Generic Elective II
GE34921	Software Reliability Engineering
GE34922	Software Architecture and Integration
GE34923	System Simulation and Modeling
GE34924	Agile Software Engineering

Course Code	Generic Elective III
GE34931	Embedded System Theory & Design
GE34932	Requirements Elicitation and Analysis
GE34933	Software Reusability
GE34934	Formal Software Specifications

Credit Summary Chart						
	Semester				Total Credits	%age
Course Category	I	II	III	IV		
F	0	0	0	0	0	0
C	19	15	20	28	82	87.23
GE	4	8	0	0	12	12.77
Total	**23**	**23**	**20**	**28**	**94**	**100.00**

Discipline Wise Credit Summary Chart						
	Semester				Total Credits	%age
Course Category	I	II	III	IV		
Basic Sciences	0	0	0	0	0	0
Professional Subject Core	18	14	16	28	76	80.85
Professional Subject -General Elective	4	8	0	0	12	12.77
Project Work, Seminar and / or internship in Industry or elsewhere.	1	1	4	0	6	6.38
Total	**23**	**23**	**20**	**28**	**94**	**100**

SYLLABUS

MCS3111 Advance Software Engineering Principles and Practices

Course Objective:

1. Familiarize knowledge and practical skills in the development of software systems of high quality, which is invaluable for software architects, project managers and technical specialists.
2. Master General Principles and techniques for dealing with quality attributes for various types of software systems.
3. Analytically apply general principles of software development in the development of complex software and real time systems.

Learning Outcome:

At the end of the course, the student should be able to:

1. Understand and adhere to professional ethical standards in the system development and modification process, especially by accepting responsibility for the consequences of design decisions and design implementations.
2. Analyze and implement solutions to complex problems including human computer interface design.
3. Have demand for knowledgeable experts in software engineering is steadily increasing, which makes students very competitive nationally as well as internationally, both in industry and in academic research.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**System Engineering** Computer based systems, system engineering hierarchy, Information engineering, Information strategy planning, Business area analysis, Product engineering, Modeling the system architecture, System modeling and simulation, system specification. Computer Based System Engineering: Emergent system properties.	30 Hours	1
II	**Modern Design Concept and Principles** Design Concepts: Mapping of analysis model to design model, Design process, design principles, Design concepts, effective modular design, Design model, design specification. Architectural Design Process: Transform mapping and transaction mapping, Design post processing, interface design, Human computer interface design, and Interface design guidelines.	30 Hours	1
III	**Real Time Software Design** Real–time systems, definition, System consideration, Real time system analysis, stimulation / Response systems, Real –time System model, system elements, Real –time programming, system design, Real-time system modeling, RTOS, process priority, process	30 Hours	1

	management, Scheduling strategy, Monitoring and control system Generic architecture, Data acquisitions systems.		
IV	**Component Based Development** CBSE: Component based software engineering, Components and component models; Component based software engineering process, Component Composition. Software Reuse: Management issues, Reuse process, Domain engineering, Building Reusable Components, classification and retrieving components.	30 Hours	1

Text/Reference Books:

1. Roger S. Pressman, "Software Engineering – A Practitioner's Approach", McGraw Hill Publications.
1. 2. Ian Sommerville, "Software Engineering", Pearson Education Publications.
2. Shari Lawarence Pfleeger, "Software Engineering Theory and Practices",
3. John W. Satzinger, Robert B Jackson, Stephen D Burd, "System Analysis and Design in Changing World", Thomson Course Technology.
4. Richard Murch, Tony Johnson, "Intelligent Software agents", Prentice Hall

MCS3112 Object Oriented Software Engineering & UML

Course Objective:

1. Study the basic concepts and functions of UML
2. Learn about Modeling, Architecture and SDLC.
3. Learn various Diagrams like Class, Object diagram
4. Understand the Modeling Techniques.
5. Learn various Architectural modeling likes Activity, component diagrams

Learning Outcome:

At the end of the course, the student should be able to:

1. Draw the various Classes and Objects
2. Apply the Modeling Techniques.
3. Uses Of various Relationships.
4. Compare various interactions Diagrams
5. Design and Implement of Activity, Component and Deployment diagram

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction to UML** Introduction to UML Model, Importance of Model, Principles of Modeling, Object oriented Modeling, Conceptual Model of UML, Architecture, Software Development Life Cycle.	30 Hours	1
II	**Basic and Advanced Structural Modeling** Classes and Advanced Classes, Relationship and	30 Hours	1

	Advanced Relationship- Dependency, Generalization and Realization, Association and Aggregation, Association Class .Interfaces, Packages. Class and Object Diagrams- Terms and Concepts And Modeling Techniques		
III	**Basic Behavioral Modeling-I:** Interactions - Terms and Concepts, Interaction diagrams and its Type- Sequential diagram, Collaboration diagram with Modeling Techniques. **Basic Behavioral Modeling-II** Use cases- Terms and Concepts, Use case diagrams- Use case and Collaboration and Modeling Techniques, Activity diagrams- Forking and Joining, Swimlanes and Object flow.	30 Hours	1
IV	**Architectural Modeling** Events and signals, Processes and Threads, State Machines and State Chart diagrams. Component and Component diagrams- Terms and Concepts with Modeling Techniques. Deployment and Deployment diagrams-Terms and Concepts with Modeling Techniques.	30 Hours	1

Text/ Reference books:

1. Grady Booch, James Rumbaugh, Ivar Jacobson: "The Unified Modeling Language User Guide", Pearson Education.
2. Hans-Erik Eriksson, Magnus Penker, Brian Lyons, David Fado: "UML 2 Toolkit", WILEY -Dreamtech India Pvt. Ltd.
3. Meilir Page-Jones: "Fundamentals of Object Oriented Design in UML", Pearson Education.
4. Pascal Roques: "Modeling Software Systems Using UML2", WILEY Dreamtech India Pvt. Ltd.
5. AtulKahate: "Object Oriented Analysis & Design", The McGraw-Hill Companies.
6. Mark Priestley: "Practical Object-Oriented Design with UML" , TATA McGraw Hill

MCS3103 Fundamentals of Theoretical Computer Science

Course Objective:

1. Automata are mathematical machines, that is, abstract computing devices. Their purpose is to capture study and compare different models and views of the abstract notion of computation and its various aspects.
2. The computational power of automata can be characterized through the classes of languages (that is, sets of strings over a finite alphabet of symbols) they can accept/recognize.
3. Important notions in computer science like state, non determinism and minimization are captured in the simple model of finite automata, which recognize the class of regular languages.

4. Automata provide the basis for the implementation of many programming languages, with parsing being a typical example for the application of pushdown automata, which recognize the more powerful class of context-free languages.
5. Another important reason for studying automata is to capture the notion of effective computability, that is, to characterize the notion of computation as a process which can be physically implemented. This allows the important question to be posed: what problems can be decided algorithmically, and where are the limits to this? The most influential model for studying these issues is Turing machines.

Learning Outcome:

1. The overall aim of the course is to provide students with a profound understanding of computation and effective computability through the abstract notion of automata and the language classes they recognize.
2. Along with this, the students will get acquainted with the important notions of state, non-determinism and minimization.
3. After the course, the successful student will be able to perform the following constructions:
    a. Determining and minimizing automata.
    b. Construct an automaton for a given regular expression.
    c. Construct a pushdown automaton for a given context-free language.
    d. Construct a Turing machine deciding a given problem.
    e. Prove undecidability of a given problem by reducing from a known undecidable problem

Course Contents:

Module	Course Topics	Total Hours	Credits
I	Mathematical Preliminaries: Sets, Relations, Partial Orders, Well-ordered sets, Structural induction, Cardinality of Sets. Logic : Basics of propositional and first order logic, completeness and compactness results	30 Hours	1
II	Models of computation: Classification, Properties and equivalences, Regular languages models, Finite state machines (deterministic and nondeterministic). Grammars - Production systems - Chomsky hierarchy, Regular grammars, Regular expressions, Equivalence of deterministic and non-deterministic machines. Properties: closure decidability, minimality of automata, iteration theorems.	30 Hours	1
III	Recursive and recursively enumerable sets models: Turing machines, grammars, recursive functions, their equivalence. Context-free languages models: grammars (including different normal forms). Pushdown automata and their equivalence. Properties: closure, iteration theorems, parsing.	30 Hours	1
IV	Church's thesis, undecidability, Post machines, and Computational complexity: time & tape bounds, time	30 Hours	1

	& tape bounded simulations, notion of complexity classes, classes P & NP, NP-completeness, some natural NPcomplete problems.		

Text/ Reference Books:

1. Raynold M. Smullyan.First-Order Logic, Springer-Verlag, 1968.
2. Introduction to Antomata Theory, Languages of Computations, J. E. Hopcroftand J. Ullman. Addison Wesley.
3. W. Thomas, Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, Handbook of Formal Languages, volume III. Springer, New York, 1997.

Generic Elective I

GE34912 Software Agents

Course Objective:

1. The aim of this course is to introduce the concepts, techniques and applications of software agents.
2. By the end of the course the students are expected to be able to understand the nature, concepts and techniques of the agent technology and its standards and to evaluate current software agent systems.

Learning Outcome:

By the end of this course, students should be able to:

1. Understand the basic concepts techniques and applications of software agents.
2. Learn how to do a research.
3. Understand software agents design tools.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Agents and Multi-Agent Systems** Introduction to Intelligent Agent: Situated Agents: Actions and Percepts, Proactive and Reactive Agents: Goals and Events, Challenging Agent Environments: Plans and Beliefs, Social Agents. Agent Execution Cycle. Prometheus: A Brief Overview: System Specification, Architectural Design, Detailed Design Guidelines for Using Prometheus. Agent-Oriented Methodologies	30 Hours	1

II	**Goal Specification** System Specification. Goal Specification: Identify Initial Goals, Goal Refinement. Functionalities Scenario Development: Goal Step Details, Capturing Alternative Scenarios, Interface Description Percepts and Actions .Data Checking for Completeness and Consistency.	30 Hours	1
III	**Architectural Design** Agent types. Grouping Functionalities. Agent Coupling–Acquaintance Diagrams: Agent Descriptors, architectural Design, Overall System Structure, Boundaries of the Agent System, Agent Percepts and Actions, Shared Data Objects. System Overview Diagram. Completeness and Consistency, Consistency between Agents and Functionalities. Consistency between Interaction Diagrams Scenarios and Protocols .Consistency of Communication Specifications. Consistency between Descriptors and the System Overview Diagram.	30 Hours	1
IV	**Detailed Design: Agents, Capabilities and Processes, Capabilities** Agent Overview Diagrams, Process Specifications, Capability and Process Descriptors. Detailed Design: Capabilities, Plans and Events. Capability Overview Diagrams: Subtasks and Alternative Plans, Identifying Context Conditions Coverage and Overlap, Events and Messages, Action and Percept Detailed Design, Data Develop and Refine, Descriptors. Checking for Completeness and Consistency: Agent Completeness, Missing or Redundant Items, Consistency between Artifacts, Important Scenarios. Implementing Agent Systems: Agent Platforms, Example of Agent Capabilities, Data Messages/Events, Plans Automatic Generation of Skeleton Code.	30 Hours	1

Text/ Reference Books:

1. “Beyond 3G: Bringing Networks, Terminals and the web together”, Martin Sauter, Wiley Publications.
2. “ Mobile Computing Handbook: Mohammad Illayas, CRC Press
3. “Wireless Internet and Mobile Computing: Interoperability and Performance (Information and Communication Technology Series,), John Wiley & Sons.

GE34913 Information Architecture

Objectives:

1. Acquaint students with the theory and practice of information architecture.
2. To help students to leverage their skills and knowledge in the field of software engineering within the broad context and practice of information architecture.

Learning Outcome:

By the end of this course, students should be able to:

Understand the basics of optimizing websites for search engines

1. Gain a solid comprehension of the definition and topics of information architecture
2. Learn the roles for information architecture in web design / redesign teams
3. Learn the basic tools used in by practicing information architects
4. Gain experience with organization structure, labeling and taxonomy design
5. Develop a working knowledge of user centered design and usability testing
6. Understand the issues currently being researched in information architecture

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction** Information architecture. IA, IAin project. IA for non-web: Understanding content, Understanding Content needed, Communicating about content, Content Planning.	30 Hours	1
II	**Classification schemes and analysis** Classification schemes, Understanding people Learning about their users, Analyzing user research, Communicating about users .Perspective of people about their users, people searching for information, people thinking about categories	30 Hours	1
III	**Designing information architecture** Understanding content: Content you have, Content you need, Communicating about content, Content planning, Classification schemes. Designing information architecture. IA patterns: Labels and language, creating IA, Testing IA, Communicating IA.	30 Hours	1
IV	**Navigation designing** Designing navigation, Navigation core, Navigation extras, Designing navigation, Testing navigation, Communicating navigation.	30 Hours	1

Text/Reference books:

1. A Practical Guide to Information Architecture (Practical Guide Series), Five Simple
2. Steps LLP, 2010
3. Information Architecture: Blueprints for the Web (2nd Edition) (Voices That
4. Matter), New Riders Publishing, 2009
5. Information Architecture with XML: A Management Strategy, John Wiley & Sons, 2003.

GE34914 Real Time and Concurrent systems

Course Objective:

1. To study the basic of tasks and scheduling
2. To understand programming languages and databases
3. To analyze real time communication
4. To analyze evaluation techniques and reliability models for Hardware Redundancy
5. To understand clock synchronization

Learning Outcome:

At the end of the course, the student should be able to:

1. Know the definition and characteristics of Real Time systems
2. Know the various task assignment and scheduling methods. For example RM and EDF scheduling.
3. Know the important characteristics of Real Time Operating System.
4. Know about the RT System requirement, design, and performance analysis.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Real Time Systems** Introduction**:** Definition Issues in Real Time Computing, Structure of a Real Time System, Task Classes. Characterizing Real Time Systems and Tasks: Introduction, Performance measures for real time systems-Traditional performance measures, Performability; Cost functions and hard Deadlines.	30 Hours	1
II	**Task Assignment and Scheduling:** Introduction to Classical Uniprocessor scheduling algorithms: Rate Monotonic, EDF algorithm, Task assignment, Fault tolerant Scheduling.	30 Hours	1
III	**Real Time Database** Basic definitions, Real time Vs General Purpose databases, Main Memory databases, concurrency control issues, databases for hardreal time Systems.	30 Hours	1
IV	**Real Time Communication:** Introduction: Architectural Issues; Protocols-Contention based protocols, Token based protocols, Deadlines based protocols, Stop and Go Multi-hop protocol, polled bus protocol, Hierarchical round robin protocol. .	30 Hours	1

Text/Reference Books:

1. A Practitioner's Handbook for Real-Time Analysis: Mark H. Klein, Thomas Ralya, Bill Pollak, Ray Obenza, Michael Gonzalez Harbour, Springer.
2. Software Design for Real-time Systems: J.E. Cooling, Prentice Hall.
3. Real-time systems design and analysis, an engineer's handbook, Phillip Laplante, *IEEE* Computer Society Press

GE34915 IT Infrastructure Management

Course Objective:

1. The course provides the student comprehensive knowledge, technical expertise and hands-on experience in Infrastructure Management.
2. Students would be capable of implementing IT infrastructure solutions and handling the Industry challenges
3. The course will provide an overview of the management of infrastructure both at the individual and network level.
4. It will provide background knowledge on the inspection, durability and maintenance of structures.
5. It will also provide details of asset management for structures and networks of structures, from design though operation to repair and replacement.

Learning Outcome:

At the end of the course, the student should be able to:

1. Apply a systems approach in managing infrastructure
2. Appreciate of the whole of life cycle of infrastructure systems covering the service (customer expectations), performance (technical and functional capabilities), and life expectancy.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**UNIT:1** Definitions, Infrastructure management activities ,Evolutions of Systems and their management, Current business demands and their management, Current business demands and IT systems issues, Complexity of today's computing environment, Total cost of complexity issues, Value of Systems management for business, Growth of internet,	30 Hours	1
II	**UNIT:2** Preparing for Infrastructure Management Factors to consider in designing IT organizations and IT infrastructure ,Determining customer's Requirements, Identifying System Components to manage Exist Processes, Data, applications, Tools and their integration, Patterns for IT systems management, Introduction to the design process for information systems, Infrastructure Library (ITIL)	30 Hours	1
III	**UNIT:3** Service Delivery Processes, Service-level management, financial management and costing, IT services continuity management, Capacity management, Availability management, Service Support Processes ,Configuration Management, Service desk, Incident management, Problem management, Change management, Release management.	30 Hours	1

IV	**UNIT:4** Storage and Security Management Introduction Security, Identity management, Single sign-on, Access Management, Basics of network security ,LDAP fundamentals, Intrusion detection, firewall, Security information management, Introduction to Storage, Backup & Restore, Archive & Retrieve, Space Management, AN & NAS, Disaster Recovery, Hierarchical space management, Database & Application, protection, Bare machine recovery, Data retention.	35 Hours	1

Text/ Reference Books:

1. Foundations of IT Service Management: based on ITIL, by Jan Van Bon, Van Haren Publishing, 2nd edition 2005
2. Floyd Piedad, Michael Hawkins, "High Availability: Design, Techniques, and Processes", Prentice Hall, 2000
3. Harris Kern, Stuart Galup, Guy Nemiro, "IT Organization: Building a Worldclass Infrastructure", Prentice Hall, 2000
4. Rich Schiesser, "IT Systems Management: Designing, Implementing

GE34911 Fundamentals of Operating System and DBMS

Course Objective:

1. Study the basic concepts and functions of operating systems.
2. Study the basic concepts of Database management system.
3. Understand the structure and functions of OS.
4. Learn about Processes, Threads and Scheduling algorithms.
5. Understand the principles of concurrency and Deadlocks.
6. Learn various memory management schemes.
7. Study I/O management and File systems.

Learning Outcome:

At the end of the course, the student should be able to:

1. Design various Scheduling algorithms.
2. Apply the principles of concurrency.
3. Design deadlock, prevention and avoidance algorithms.
4. Compare and contrast various memory management schemes.
5. Design and Implement a prototype file systems.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**INTRODUCTION TO OPERATING SYSTEM** Introduction to operating systems; Process management: Process synchronization, Mutual exclusion, Two process solution, Dekker's algorithm, semaphores, examples (producer-consumer, readers-writer, dining philosophers, etc.); CPU scheduling:	30 Hours	1

	Multiprogramming, Time sharing, Scheduling approaches (SJF, FIFO, round robin, etc.); Input/ Output: Device controllers, Device drivers, Disks, other devices. Memory management: With and without swapping, Virtual memory; Paging and segmentation: Page replacement algorithms, implementation; File systems: FS services, Disk space management, directory and data structure; Deadlocks: modeling, Detection and recovery, prevention and avoidance.		
II	**INTRODUCTION TO DATABASE SYSTEM** The entity-relationship model: Beyond the ER Model, Entities, Attributes, and Entity Sets, Relationships and Relationship Sets, Additional Features of the ER Model; Key Constraints: Participation Constraints, Weak Entities; Class Hierarchies Aggregation; Conceptual Database Design With the ER Model: Entity versus Attribute, Entity versus Relationship, Binary versus Ternary Relationships, Aggregation versus Ternary Relationships; Conceptual Design for Large Enterprises. Introduction to the Relational Model; Integrity Constraints over Relations: Key Constraints, Foreign Key Constraints, General Constraints, Enforcing, Integrity Constraints, Querying Relational Data; Logical Database Design; ER to Relational, Introduction to Views, Destroying/Altering Tables and Views.	30 Hours	1
III	**Relational algebra and calculus.** Relational Algebra: Selection, Projection, Set Operations, Renaming, Joins, Division, Relational Calculus, Tuple Relational Calculus ,Domain Relational Calculus, Expressive Power of Algebra and Calculus; Structure query language: Queries, Programming, Triggers. The Form of a Basic SQL Query: Expressions and Strings in the SELECT Command, UNION, INTERSECT., EXCEPT, Nested Queries, Aggregate Operators, Null Values, Embedded SQL, Cursors, ODBC, JDBC; Complex Integrity Constraints in SQL: Triggers and Active Databases, Constraints versus Triggers; The Memory Hierarchy; RAID; Data Striping; Redundancy; Disk Space Management; Files and Indexes: File organizations and indexes: Cost Model; Heap Files, Sorted Files, Hashed Files, Choosing a File Organization, Indexes, Properties of Indexes; Tree-structured indexing: Indexed Sequential Access Method (ISAM), B+ Trees, Format of a Node,	30 Hours	1

	Search Insert, Delete; Hash-based indexing: Static Hashing, Extendible Hashing, Linear Hashing, Extendible Hashing versus Linear Hashing; Evaluation of relational operators: Introduction to Query Processing, The Selection Operation, General Selection Conditions, The Projection Operation, The Join Operation, The Set Operations, Aggregate Operations; Introduction to query optimization: Overview of Relational Query Optimization, System Catalogue in a Relational DBMS, A typical relational query optimizer, Translating SQL Queries into Algebra, Estimating the Cost of a Plan, Relational Algebra Equivalences, Enumeration of Alternative Plans, Nested Sub queries, Other Approaches to Query Optimization.		
IV	**Schema refinement and normal forms.** Schema refinement and normal forms: Functional Dependencies, Normal Forms, Decompositions, Normalization, Other Kinds of Dependencies; Introduction to Database Security: Access Control, Discretionary Access Control, Mandatory Access Control; Transaction management overview: The Concept of a Transaction, Transactions and Schedules, Concurrent Execution of Transactions, Lock-Based Concurrency Control; Introduction to Crash Recovery: Lock-Based Concurrency Control Revisited, Lock Management, Specialized Locking Techniques, Transaction Support in SQL, Concurrency Control without Locking; Crash recovery: Recovering from a System Crash, Media Recovery, Other Algorithms and Interaction with Concurrency Control.	30 Hours	1

Text/Reference Books:

1. Operating System Concepts, J. Peterson, A. Silberschatz, and P. Galvin. Addison Wesley.
2. Compilers: Principles, Techniques and Tools, A. V. Aho, R. Sethi, and J. D. Ullman. Addison-Wesley.
3. Fundamentals of Data Base Systems, R. El. Masri and S. B. Navathe. Benjamin Cummings.
4. Database System Concepts, Abraham Silberschatz, Henry F. Korth, S. Sudarshan.

MCS3151 Object Oriented Modeling & Design Lab

A) Object Oriented Programming Lab (Using C++)

Experiments should include but not limited to:
A complete C++ program Assignments corresponding to fundamental C++ features like objects, classes, flexible declaration, dynamic initialization, reference variable,

inline, friend function, static member function Program introducing array, pointer to member, pointer to function. Program illustrating fundamental OOP concept: abstraction, encapsulation, inheritance–single, multiple, multilevel, hierarchical
Program on operator and function overloading, virtual function Program on files and exception handling.

B) Object Oriented Programming Lab (USING JAVA)

1. Assignments on class, constructor, overloading, inheritance, overriding
2. Assignments on wrapper class, vectors, arrays
3. Assignments on developing interfaces- multiple inheritance, extending interfaces
4. Assignments on creating and accessing packages
5. Assignments on multithreaded programming, handling errors and exceptions,
6. applet programming and graphics programming

MCS3152 Operating System Lab (Unix/Linux)

UNIX commands.

Basic File IO File Descriptors, open, creat, close, lseek, read and write Functions, Atomic Operations, dup Function, fcntl, ioctl Function, dev/fd

Files and Directories: stat, fstat, and lstat Functions, File Types, File Access Permissions, Ownership of New Files and Directories, umask Function chmod and fchmod Functions, Sticky Bit, chown, fchown, and lchown Functions, File Size, File Truncation, File systems, link, unlink, remove, and rename Functions, Symbolic Links, symlink and readlink Functions, File Times, utime Function, mkdir and rmdir Functions, Reading Directories, chdir, fchdir, and getcwd Functions, Special Device Files * sync and fsync Functions.

Standard I/O Library: Streams and FILE Objects, Standard Input, Standard Output, and Standard Error, Buffering, Opening a Stream, Reading and Writing a Stream, Formatted I/O.

Processes: main Function, Environment Variables, setjmp and longjmp Functions, getrlimit and setrlimit Functions. Fork Function, vfork Function, exit Functions, wait and waitpid, exec Functions, Threads.
Basic Interprocess Communication: Pipes, popen and pclose Functions, Coprocesses, FIFOs, IPC, Identifiers and Keys, Message Queues, Semaphores, Shared Memory.

Text/Reference Books:

1. Advanced Programming in the UNIX Environment. W. Richard Stevens, Stephen A. Rago, Addison-Wesley.

MCS3211 Software Design and Testing

Course Objective:

1. Study the basic concepts of testing.
2. Understand the static and dynamic testing.
3. Learn about software quality model.
4. Understand white box and black box testing.

5. Understand the development of test cases.

Learning Outcome:

At the end of this course the student will be able to:

1. Appreciate the fundamentals of software testing and its application through the software life cycle.
2. Develop skills in designing and executing software tests suitable for different stages in the software life cycle.
3. Understand and appreciate the role of software testing in systems development, deployment and maintenance.
4. Develop a continuing interest in software testing, and obtain satisfaction from its study and practice.
5. Appreciate the responsibilities of software testers within software projects, the profession and the wider community.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**INTRODUCTION TO SOFTWARE TESTING** Review of software testing, Overview of software testing evolution, Myths, facts, goals, principles of testing, Models and psychology of testing, Study of bugs, Classification, priority, severity of bugs, Tracking of bugs, Software Testing, Life cycle of software testing, Types of testing, Test Plan, Test plan specification, Leveled Test Plan, Development of Test Plan, Master Test Plan, Phase Wise Test Plan, Test Management, Software Testing Guidelines, Defect Management, Analyzing and Reporting Test	30 Hours	1
II	**TESTING TECHNIQUE** Static Testing, Inspection, Structured Walkthrough, Technical reviews Black Box Testing, Boundary Value Analysis, Equivalence Class Testing, Cause Effect Graph Based Testing, White Box Testing, Basis Path Testing, Control Structure Testing, Mutation Testing, Grey Box Testing	30 Hours	1
III	**SOFTWARE TESTING STRATEGIES** Model of Software Testing, Unit Testing, Integration Testing, System and Acceptance Testing, Alpha Testing and Beta Testing, Stress Testing. Load Testing and Reliability Testing, Scalability Testing, Performance Testing, Regression Testing, Ad-hoc Testing, Usability and Accessibility Testing, Object Oriented Testing	30 Hours	1
IV	**SOFTWARE QUALITY AND QUALITY MODELS** Introduction To Software Quality, Concepts of	30 Hours	1

	quality, Quality Engineering, Quality Assurance, Q.A. activities, Q.A. Techniques, Software Quality Measurement and Metrics , Quality Models, McCall's model, Bohem's model , Dromey's model, FURPS model, ISO-9126 model , Cost of Quality, Software Quality Factors, Quality Control, CMMI-framework, Process Area Components, Capability and Maturity Levels, Relationship Among Process Areas		

Text/Reference Books:

1. William Perry, "Effective Methods for Software Testing", John Wiley & Sons, New York, Van Nostrand Reinhold, New York, 2nd Ed., 1995.
2. CemKaner, Jack Falk, Nguyen Quoc, "Testing Computer Software", Van Nostrand Reinhold, New York, 2nd Ed., 1993.
3. Boris Beizer, "Software Testing Techniques", Second Volume, Van Nostrand Reinhold, New York, .2nd Ed., 1990.
4. Louise Tamres, "Software Testing", Pearson Education Asia, 2002.
5. Aditya P. Mathur, "Foundation of Software Testing", Pearson, 2008.

MCS3212 Web Semantics

Course Objective:

1. Explain the origins of Semantic Web and understand the basic underlying technology.
2. Motivate the need for new mechanisms for representing and using information in the Web.
3. To teach the students the concepts, technologies and techniques underlying and making up the Semantic Web.

Learning Outcome:

At the end of the course, the student should be able to:

1. Know what is the "Semantic Web"
2. Learn some of Semantic Web technologies and applications.
3. Use ontology engineering approaches in semantic applications

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction** Semantics for the Semantic Web: The Implicit, the Formal, and the Powerful, The Human Semantic Web: Shifting from Knowledge Push to Knowledge Pull, General Adaptation Framework: Enabling interoperability for Industrial Web Resources.	30 Hours	1
II	**Ontology Creation Methodologies** A Survey on Ontology Creation Methodologies, A Tool for Working with Web Ontologies, An Ontology-Based Multimedia ,Annotator for the Semantic Web of Language Engineering, A Layered	30 Hours	1

	Model for Building Ontology Translation Systems. Querying the Web Reconsidered: Design Principles for Versatile Web Query Languages.		
III	**Semantic Applications** Semantic E-Business, A Distributed Patient Identification Protocol Based on Control Numbers with Semantic Annotation, Static and Dynamic Semantics of the Web, Sources of Dynamic Semantics, Web Agents Make Use of Dynamic Semantics, Information Retrieval and Theorem-Proving Perspectives.	30 Hours	1
IV	**Semantic Annotation** Semantic Annotation for Web Content Adaptation, External Annotation Framework, Annotation-Based Transcoding System, HTML Page Splitting for Small-Screen Devices.	30 Hours	1

Text/Reference Books:

1. Deitel & Deitel, Goldberg, "Internet and World Wide Web – How to Program", Pearson Education Asia, 2001.
2. Eric Ladd, Jim O' Donnel, "Using HTML 4, XML and JAVA", Prentice Hall of India .
3. Aferganatel, "Web Programming: Desktop Management", PHI, 2004.
4. Rajkamal, "Web Technology", Tata McGraw-Hill, 2001.

MCS3213 Software Project Management

Course Objective:

1. Understand the fundamental principles of Software Project management.
2. To provide basic project management skills with a strong emphasis on issues and problems associated with delivering successful IT projects.
3. The module is designed to provide an understanding of the particular issues encountered in handling IT projects and to offer students methods, techniques and 'hands-on' experience in dealing with them.

Learning Outcome:

At the end of the course, the student should be able to:

1. Participate in a software development project as a project manager.
2. Take responsibility of a project team and project organization.
3. Apply theoretical knowledge on project management and software development into practice
4. Have good knowledge of the issues and challenges faced while doing the Software project Management
5. Understand why majority of the software projects fails and how that failure probability can be reduced effectively.
6. Do the Project Scheduling, tracking, Risk analysis, Quality management and Project Cost estimation using different techniques

Course Contents:

Module	Course Topics	Total Hours	Credits
I	Planning fundamentals-major issues in software project planning –planning activities -Project master schedule-software risk management –risk monitoring – risk analysis.	30 Hours	1
II	Software cost- major issues in estimating software cost- cost estimation method- Experience based model- parameter based model- COCOMO- versions of COCOMO – Software size estimating – function points – software project schedule – Raleigh model.	30 Hours	1
III	Function organization – project organization – matrix organization – staffing quality replacement – turnover management. Directing a software engineering project – issues – activities conflict management. Issues in controlling software project- controlling activities threads of control- Work breakdown structures- earned value tracking.	30 Hours	1
IV	Team Software Process TSPi Overview: What is TSPi? TSPi Principles, the TSPi Design, TSPi Structure and Flow the TSPi Process, the Textbook Structure and Flow. The Logic of the Team Software Process: Why Projects Fail, Common Team Problems, What is Team? Building Effective Teams, How Teams Develop, How TSPi Builds Teams.	30 Hours	1

Text/Reference Books:

1. Watts S. Humphrey "Introduction to the Team Software Process(sm)", Carnegie-Mellon University, Addison Wesley Professional.
2. Bob Hughes and Mike Cotterell; Software Project Management, third edition, Tata
   McGraw Hill Publishing Company Ltd., New Delhi.
3. Pankaj Jalote; Software Project Management in Practice, Pearson Education Asia.
4. Watts S. Humphrey; Winning with Software – An Executive Strategy, Pearson Education Asia.
5. Software Engineering Project Management: Richard Thayer, IEEE Computer Society

Generic Elective II

GE34921 Software Reliability Engineering

Course Objective:

1. To apply engineering knowledge and specialist techniques to prevent or to reduce the likelihood or frequency of failures.
2. To identify and correct the causes of failures that may occur, despite the efforts to prevent them.
3. To determine ways of coping with failures that may occur, if their causes have not been corrected.

4. To apply methods for estimating the likely reliability of new designs, and for analyzing reliability data.

Learning Outcome:

At the end of the course, the student should be able to:

1. Calculate failure rates, MTTF, etc. including confidence limits.
2. Design a reliability experiment and fit data to a model.
3. Design a reliability validation plan.
4. Be able to build system-level quality and reliability models from component-level models.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**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; Reliability Evaluation.	30 Hours	1
II	**SOFTWARE RELIABILITY MODELS** 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: Little wood verral Model; Phase Based Model .	30 Hours	1
III	**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.	30 Hours	1
IV	**THE OPERATIONAL PROFILE AND RELIABILITY MEASUREMENT** 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; Time/Structure based software reliability: Assumptions, Testing methods, Limits.	30 Hours	1

Text/Reference Books:

1. Patric D. T.O Connor, “Practical Reliability Engineering”, 4th Edition, John Wesley & sons, 2003.
2. John D. Musa, “Software Reliability Engineering”, Tata McGraw Hill, 1999.

3. Michael Lyu, "Handbook of Software Reliability Engineering", IEEE Computer Society Press

GE34922 Software Architecture and Integration

Course Objective:

1. To test software performance under given conditions without any type of corrective measure.
2. To find the number of failures occurring in a specified amount of time.
3. To discover the main cause of failure.
4. To find the mean life of the software.

Learning Outcome:

At the end of the course, the student should be able to:

1. Be aware of the key elements of software architecture
2. Be familiar with a variety of architectural styles and how they may be combined in a single system
3. Have a working knowledge of software architecture design for a non-trivial system
4. Understand how software architecture aids different stages of the software lifecycle

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Software Architecture terms:** Component; Relationship; View; Architectural Styles; Frameworks; Patterns; Methodologies; Processes; Functional and Non-Functional Properties of Software Architectures;	30 Hours	1
II	**Enabling Techniques for Software Architecture:** Abstraction; Encapsulation; Information Hiding; Modularization Separation of Concerns; Coupling and Cohesion; Sufficiency; Completeness and Primitiveness Separation of Policy and Implementation; Separation of Interface and Implementation.	30 Hours	1
III	**Architectural Styles:** Pipes and Filters; Data Abstraction and Object-Orientation; Event-Based; Implicit Invocation; Layered Systems; Repositories ; Interpreters; Process Control; Heterogeneous Architectures ;	30 Hours	1
IV	**Software Implementation - development environment facilities:** Code Generation; Reverse Engineering; Profiling; Software Libraries; Testing and debugging; Software Quality**:** Changeability, Efficiency, Interoperability, Reliability, Testability, Reusability,	30 Hours	1

	Fault tolerant software.		

Text/Reference Books:

1. M. Shaw: Software Architecture Perspectives on an Emerging Discipline, Prentice-Hall.
2. "Software Architecture: Foundations, Theory and Practice" by Richard N.Taylor, Nenad Medvidovic, Eric Dashofy, ISBN:978-0-470-16774-8
3. Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice, Pearson.

GE34923 System Simulation and Modeling

Course Objective:

1. The basic system concept and definitions of system.
2. Techniques to model and to simulate various systems the ability to analyze a system and to make use of the information to improve the performance.

Learning Outcome:

At the end of the course, the student should be able to:

1. Define basic concepts in modeling and simulation (M&S)
2. Classify various simulation models and give practical examples for each category
3. Construct a model for a given set of data and motivate its validity
4. Generate and test random number varieties and apply them to develop simulation models
5. Analyze output data produced by a model and test validity of the model
6. Explain parallel and distributed simulation methods

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**INTRODUCTION TO SIMULATION** Simulation: Simulation as a tool, Advantages and Disadvantages of Simulation, Areas of Application, Systems and System Environment. Components of a System. Discrete and Continuous Systems. Model of a System, Types of Models, Discrete-Event System Simulation. Steps of Simulation Study	30 Hours	1
II	**GENERAL PRINCIPLES** Concepts in Discrete-Event Simulation: The Event-Scheduling / Time-Advance: Algorithm, World Views. Manual simulation, Using Event Scheduling. Properties of Random Numbers, Generation of Pseudo-Random Numbers. Techniques for Generating Random Numbers. Tests for Random Numbers	30 Hours	1
III	**RANDOM-VARIATE GENERATION** Inverse Transform technique: Exponential Distribution, Uniform Distribution, Discrete Distributions, Acceptance-Rejection Technique,	30 Hours	1

	Poisson Distribution. Data Collection, Identifying the distribution with Data, Parameter Estimation, Goodness of Fit Tests, Selecting Input Models without Data Multivariate and Time-Series Input Models.		
IV	**VERIFICATION AND VALIDATION OF SIMULATION MODELS** Model Building, Verification and Validation: Verification of Simulation Models, Calibration and Validation of Models. Types of Simulations with Respect to Output Analysis .Stochastic Nature of Output Data. Measures of Performance and Their Estimation. Output Analysis for Terminating Simulations, Output Analysis for Steady-State Simulations. Simulation Tools, Model Input. High-Level Computer-System Simulation, CPU Simulation, Memory Simulation.	30 Hours	1

Text/Reference Books:

1. Jerry Banks, John S. Carson, Barry L. Nelson, David M. Nicol, "Discrete-Event System Simulation", Third Edition, Prentice-Hall India
2. Averill M. Law, W. David Kelton, "Simulation Modeling and Analysis" Third Edition, McGraw Hill.
3. Geoffrey Gordon, "System Simulation", Second Edition, Prentice-Hall India.

GE34924 Agile Software Engineering

Course Objective:

1. Understand the principles of agile development.
2. Understand a range of practices that agile software development teams can apply.
3. Understand the principles of component based development.
4. Develop a large software artifact using .NET technologies as a member of a team using both agile project management and agile software engineering practices.
5. Demonstrate proficiency in using a range of contemporary software development tools and techniques.

Learning Outcome:

At the end of the course, the student should be able to:

1. Understand and explain the range of factors that can influence the success of an agile development project.
2. Develop a large software artifact using Estreme Programming as part of a team which demonstrates good software design skills, code refactoring skills, comprehensive software testing skills and good coding standards and documentation skills.
3. Complete a project which demonstrates strong time management and agile project management skills.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction** Introduction to Agile Methodology, The Quality Agenda, Do We Really Need All This Mountain of Documentation? The Human Factor, Some Agile Methodologies, Dynamic Systems Development Method, Feature-Driven Design, Crystal Agile Modeling, SCRUM.	30 Hours	1
II	**Extreme Programming** Extreme Programming Outlined, Some Guiding Principles, The Five Values, Communication Feedback, Simplicity Courage, The 12 Basic Practices of XP 25, Test First Programming, Pair Programming, On-Site Customer, The Planning Game, System Metaphor, The Evidence for XP, Evidence for Test First, Evidence for Pair Programming	30 Hours	1
III	**Foundations: People and Teams Working Together** Observations of Team Behavior in XP Projects, Setting Up a Team, Developing Team Skills, Training Together, Finding and Keeping a Client for a University-Based Project or a Small Business Start-Up, The Organizational Framework, Planning PERT (Program Evaluation and Review Technique), Gantt Charts	30 Hours	1
IV	**XP Project** Starting an XP Project, The First Meetings with the Client, Business Analysis and Problem Discovery, The Initial Stages of Building a Requirements Document, Techniques for Requirements Elicitation, XP Architectures and Patterns, Finite State Machines, Extreme Modeling (XM), Multiple Stories and XXMs, Building the Architecture to Suit the Application: A Dynamic System Metaphor, Another Look at Estimation; Testing Internet Applications and Web Sites, Units and Their Tests, Writing Unit Tests in JUniti.	30 Hours	1

Text/Reference Books:

1. Kent Beck, “Extreme Programming Explained: Embrace Changes”, Addison Wesley, 1999.
2. Ken Schwaber and Mike Beedle, “Agile Software Development with SCRUM”, Prentice Hall, 2002 ISBN 0-13-067634-9
3. Jim Highsmith, “Agile Software Development Ecosystems”, Addison Wesley, 2002
4. Alistair Cockburn, “Agile Software Development”, Addison Wesley, 2002

5. Robert Martin, "Agile Software Development: Principles, Patterns, and Practices", Prentice Hall, 2003.

Generic Elective III

GE34931 Embedded System Theory & Design

Course Objective:

1. To impart fundamental concepts in the area of Embedded Systems.
2. To impart the design an embedded system.
3. To impart the partition a system to hardware and software parts efficiently.
4. To impart the Hardware/software Co-design concepts.

Learning Outcome:

At the end of the course, the student should be able to:

1. Design embedded system architectures for various applications.
2. Identify, formulate, and solve engineering problems
3. Learn Function on multidisciplinary teams.
4. Apply knowledge of mathematics, science and engineering.
5. Explain applications, benefits, and limitations of networked embedded systems for environmental science, health, and safety, industrial, and consumer usage objectives.
6. Develop standard project plans for a software development team including interface definition.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Unit 1:Introduction:** Definition**,** Embedded System Project Management**,** Embedded System Design and Issues**,** Design Cycle**,** Use of Target System and In-Circuit Emulator**,** Use of Software Tools for Development of Embedded System**,** Middleware Design and Implementation for Networked Embedded Systems	30 Hours	1
II	**UNIT:2 RTOS and Microcontroller:** Task and Task States**,** Task and Data**,** Semaphores and Shared Data Operating System Services**:** Message Queues**,** Timer Functions**,** Events**,** Memory management**,** Interrupts Routines in an RTOS**,** Microprocessor Vs. microcontroller**,** 8051 microcontroller	30 Hours	1
III	**UNIT:3 Embedded System Development:** Embedded System Evolution Trends**,** Round Robin**,** Robin with Interrupts**,** Function One Scheduling Architecture, Assembler, Compiler, Cross Compiler and IDE**,** Object Oriented Interfacing**,** Recursion**,** Debugging Strategies, Simulators, Wireless Sensor Networks: Introduction**,** Architectures for Wireless	30 Hours	1

	Sensor Networks , Time Synchronization Issues in Sensor Networks, Resource Aware Localization in WSN, Power Efficient Routing in WSN, Energy Efficient MAC Protocols for WSN 3.10.6 Energy Efficient MAC Protocols for WSN		
IV	**UNIT:4 Networks for Embedded System:** I2C Bus, CAN Bus, SHARC Link Ports, Ethernet, Myrinet, Internet, Bluetooth, IEEE 1149.1 (JTAG) Testability, Networked Embedded Systems in Building Automation and ControlData Communications for Distributed Building Automation	30 Hours	1

Text/Reference Books:

1. "Embedded Systems", Raj Kamal, TMH.
2. "The 8051 Microcontroller", K. J. Ayala, Penram International.
3. "Design with PIC Icrocontroller", J. b. Peatman, Printice Hall.
4. " Real Time Systems", H.Kopetz, Kluwer, 1997.
5. "Co-synthesis of hardware and software for embedded systems", R. Gupta, Kluwer,1996.

GE34932 Requirements Elicitation and Analysis

Course Objective:

1. To describe the processes of requirements elicitation and analysis.
2. To introduce a number of requirements elicitation and requirements analysis techniques
3. To discuss how prototypes may be used in the RE process.

Learning Outcome:

At the end of the course, the student should be able to:

Know activities in the requirements engineering process which are concerned with discovering requirements and analyzing requirements for incompleteness, inconsistency, relevance and practicality, as well as negotiating the final requirements for the system.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	Introduction to Requirements Engineering, Misconceptions about Requirements Engineering, Industrial Challenges in Requirements Engineering, Key Success Factors in Requirements Engineering, Characteristics of a Good Requirement, Quality and Metrics in Requirements Engineering. Requirements Engineering Artifact Modeling, Using the Artifact Model, Eliciting Requirements, Issues and Problems in Requirements Elicitation, Requirements Elicitation Methods: Business Goals, Ethnographic Techniques,	30 Hours	1

	Quality Function Deployment (QFD) Method, and Customer-Specific Business Rules.		
II	Requirements Modeling: Model-Driven Requirements Engineering (MDRE)- Advantages, Estimate Project Size and Cost, Rapid Review of Business Processes and Requirements Relationships, Metrics for Quality and Progress, Prerequisites for Using MDRE, MDRE Processes, Elicitation and Analysis Model Heuristics, Determining Model Completeness, Transitioning from Analysis to Design, Design Model Structure.	30 Hours	1
III	Quality Attribute Requirements: Methods for Architectural Requirements Engineering, Testing ASRs. Requirements Engineering for Platforms: Derive the NFRs for the Software Platform Requirements Management: Routine Requirements Management Activities, Identifying Volatile Requirements, Establishing Policies For Requirements Processes And Supporting Them With Workflow Tools, Guidelines, Templates And Examples, Traceability. Measurement and Metrics: Project Metrics, Quality Metrics. Scalability.	30 Hours	1
IV	Requirements-Driven System Testing: RE Inputs for Testing, Model-Based Testing, Testing Performance and Scalability Requirements, Rules of Thumb/Best Practices. Rapid Development Techniques for Requirements Evolution: When to Prototype, Requirements Engineering and Prototype Development in Parallel, Identify and Eliminate Stakeholder Conflicts, Rapid Iteration of Requirements/Stakeholder Feedback.	30 Hours	1

Text/ Reference Books:

1. “Software & Systems Requirements Engineering: In Practice”, Brian Berenbach, Daniel J. Paulish, Juergen Kazmeier, Arnold Rudorfer, McGraw Hill.

GE34933 Software Reusability

Course Objective:

1. To learn the basics of software reuse.
2. To learn about Establishing and managing reuse Business.
3. To learn about various models used for software Reuse.
4. To learn about object oriented business engineering.
5. To learn about Software Reuse technologies.

Learning Outcome:

At the end of the course, the student should be able to:

1. Have an ability to understand the basics of software reuse.
2. Analyze various models used for software reuse.

3. Analyze and understand the effective reuse of components.
4. Have an ability to manage the reuse business.
5. Analyze various Software Reuse technologies.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction** Software Reuse: Software Engineering, Concepts and Terms; Software Reuse products; Software Reuse processes; Software reuse paradigms; State of the Art and the Practice: Software Reuse Management, Software Reuse Techniques, and Aspects of Software Reuse, Organizational Aspects, Technical Aspects and Economic Aspects.	30 Hours	1
II	**Programming Paradigms and Reusability** Usability Attributes; Representation and Modeling Paradigms; Abstraction and Composition in development paradigm.	30 Hours	1
III	**Object - Oriented Domain Engineering** Abstraction and parameterization techniques; Composition techniques in Object Orientation; Organizing a Reuse Business: Transition to a Reuse Business, Managing the reuse business, Making the reuse Business work.	30 Hours	1
IV	**Software Reuse Technologies** Application Engineering: Component Storage and Retrieval, Reusable Asset Integration; Software Reuse technologies: Component Based Software Engineering, COTS based development, Software Reuse Metrics, Tools for Reusability.	30 Hours	1

Text/Reference Books:

1. Reuse Based Software Engineering techniques, Organization and Measurement by Hafedh Mili Sherif Yacoub and Edward Addy, John Wiley & Sons Inc.
2. The Three Rs. of Software Automation: Re-engineering, Repository, Reusability by Carma McClure, Prentice Hall New Jersey.
3. Ivar Jacobson, Martin Gres, Patrick Johnson, “Software Reuse”, Pearson Education, 2004.
4. Even Andre Karisson, “Software Reuse - A Holistic Approach ", John Wiley and Sons, 1996.
5. Karma McClure, “Software Reuse Techniques - Additional reuse to the systems development process ", Prentice Hall, 1997.

Web resources

1. http://alpha.fdu.edu/~levine/reuse_course
2. http://en.wikipedia.org/wiki/Code_reuse
3. http://www.inf.fu-berlin.de/inst/ag-se/teaching/V-SWT-2004/18_software-reuse.pdf

4. http://www.cs.ccsu.edu/~stan/classes/CS530/Slides/SE-18.pdf

GE34934 Formal Software Specifications

Course Objective:

1. To learn the basics of Formal Software Specifications.
2. To learn the basics of Specification in VDM-SL.
3. To learn the basics of Sets, Sequences and Stack.
4. To learn the basics of Composite Objects & Maps.
5. To learn the Implementation of Formal Software Specifications.

Learning Outcome:

At the end of the course, the student should be able to:

1. An ability to understand the basics of Formal Software Specifications.
2. To analyze and understand the the Implementation of Formal Software Specifications.
3. An ability to understand Composite Objects & Maps.
4. To analyze and understand the UML specifications.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Propositional and Predicate Logic** Introduction to Propositions & Predicate Logic; An Introduction to Specification in VDM-SL; UML Specifications: Introduction to UML, Specifying the Operations, Specifying the State Specifying Functions, A Standard Template for VDM-SL Specifications.	30 Hours	1
II	**Sets, Sequences and Stack** Sets: Introduction, Sets for System Modeling, Declaring Sets in VDM-SL, Defining Sets in VDM-SL; Implementing Sets: Introduction, The Collection Classes of Java, Using a Vector to Implement a Set; Sequences: Introduction, Notation, Sequence Operators, Defining a Sequence by Comprehension, Using the Sequence Type in VDM-SL. Stack: Specifying a Stack, Specifying the State of the Stack, Specifying the Operations on the Stack.	30 Hours	1
III	**Composite Objects & Maps** Composite Objects: Defining Composite Object Types ,Composite Object operators, Implementing Composite Objects; Maps :Introduction and Implementation, Using the Map Type in VDM-SL;	30 Hours	1
IV	Case Study : Part 1: Specification (Introduction, The Requirements Definition, The Informal Specification, The Formal Specification) Part 2: Implementation (Developing and Using a Class with any suitable example.)	30 Hours	1

Text/Reference Books:

1. Formal Software Development, Palgrave Macmillan, 2003

MCS3251 Software Design and Testing Lab

List of experiments will cover the following testing techniques:

Black box testing , White box testing , Unit testing , Incremental integration testing , Integration testing , Functional testing , System testing , End to End testing , Sanitytesting or smoke testing , Regression testing ,Acceptance testing , Load / stress /performance testing , Usability testing , Install / Uninstall testing , Recovery / failover testing , Security testing , Compatibility testing , Exploratory testing , Ad-hoc testing , Context driven testing , Comparison testing , Alpha testing , Beta testing, Mutation testing.

MCS3252 Web Semantic Lab

Graphically design RDF instance documents, RDFS vocabularies, and OWL ontologies then output them in either RDF/XML or N-Triples formats.