(w.e.f 2015-16)

Jawaharlal Nehru Technological University Anantapur
College of Engineering Ananthapuramu *(Autonomous)*

Course Structure for Master of Technology (Software Engineering)

(w.e.f 2015-16)

I Year I Semester

Code	Subject	L	T/P/D	C
15D51103	Advances in Software Engineering	4	0	4
15D52101	Service Oriented Architecture	4	0	4
15D52102	Formal Methods of Software Engineering	4	0	4
15D52103	Software Requirements and Estimation	4	0	4
	Elective –I	4	0	4
15D52104	1. Software Metrics and Reuse			
15D52105	2. Reverse Engineering			
15D52106	3. Software architecture & Design Patterns			
	Elective –II	4	0	4
15D52107	1. Agile Methodologies			
15D52108	2. Protocol Software Engineering			
15D52109	3. Component based software Engineering			
15D52110	Software Engineering and Service oriented architecture Lab	0	4	2
	Total	**24**	**4**	**26**

I Year II Semester

Code	Subject	L	T/P/D	C
15D52201	Software Project planning & Management	4	0	4
15D51203	Software Quality Assurance and Testing	4	0	4
15D52202	Secure Software Engineering	4	0	4
15D52203	Model Driven Software Development	4	0	4
	Elective –III	4	0	4
15D52204	a. Software Agents			
15D52205	b. Software Evolution and Maintenance			
15D52206	c. Software Process Management			
	Elective –IV	4	0	4
15D52207	a. Software Reliability			
15D52208	b. Big Data			
15D52209	c. Software Reengineering			
15D54201	Research Methodology (Audit Course)			
15D52210	Software Quality Assurance and Testing Lab	0	4	2
	Total	**24**	**4**	**26**

(w.e.f 2015-16)

III & IV Semester

Code	Subject		L	P	C
15D52301	III Semester	Seminar - I	0	4	2
15D52401	IV Semester	Seminar - II	0	4	2
15D52302	III & IV Semester	Project Work	--	--	44
	Total		**0**	**8**	**48**

Note: All End Examinations (Theory and Practical) are of three hours duration.

T- Tutorial L- Theory P- Practical/Drawing C - Credits

(w.e.f 2015-16)

JNTUA College of Engineering (*Autonomous)*::Ananthapuramu

Department of Computer Science & Engineeting

M.Tech. I – I Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D51103: Advances in Software Engineering

Objectives:

The course should enable the student

- a broad and critical understanding of all the processes for engineering high quality software and the principles, concepts and techniques associated with software development
- an ability to analyze and evaluate problems and draw on the theoretical and technical knowledge to develop solutions and systems
- a range of skills focused on the analysis of requirements, design and implementation of

  reliable and maintainable software, with strong emphasis on engineering principles applied over the whole development lifecycle
- an awareness of current research in software development, the analytical skills and research techniques for their critical and independent evaluation and their application to new problems.

Unit - I :

Software and Software Engineering: The Nature of Software, The Unique Nature of WebApps, Software Engineering, Software Process, Software Engineering Practice, Software Myths.

Process Models: A Generic Process Model, Process Assessment and Improvement, Prescriptive Process Models, Specialized Process Models, The Unified Process, Personal and Team Process Models, Process Terminology, Product and Process.

(w.e.f 2015-16)

Unit – II:

Understanding Requirements: Requirements Engineering, Establishing the Groundwork, Eliciting Requirements, Developing Use Cases, Building the Requirements Model, Negotiating Requirements, Validating Requirements.

Requirements Modeling: Requirements Analysis, Scenario-Based Modeling, UML Models That Supplement the Use Case, Data Modeling Concepts, Class-Based Modeling.

Unit – III :

Design Concepts: Design within the Context of Software Engineering, Design Process, Design Concepts, The Design Model.

Architectural Design: Software Architecture, Architectural Genres, Architectural Styles, Architectural Design, Assessing Alternative Architectural Designs, Architectural Mapping Using Data Flow.

Component-Level Design: What is a Component, Designing Class-Based Components, Conducting Component-Level Design, Component-Level Design for WebApps, Designing Traditional Components, Component-Based Development.

Unit – IV :

User Interface Design: The Golden Rules, User Interface Analysis and Design, Interface Analysis, Interface Design Steps, Design Evaluation.

Coding and Testing: Coding, Code Review, Software Documentation, Testing, Testing in the Large versus Testing in the Small, Unit Testing, Black-Box Testing, White-Box Testing, Debugging, Program Analysis Tools, Integration Testing, Testing Object-Oriented Programs, System Testing, Some General Issues Associated with Testing.

Unit – V :

Verification and Validation: Planning Verification and Validation, Software Inspections, Automated Static Analysis, Verification and Formal Methods.

Software Maintenance: Characteristics of Software Maintenance, Software Reverse

Engineering, Software Maintenance Process Models, Estimation of Maintenance cost.

Text Books :

1. Software Engineering A Practitioner's Approach, Roger S. Pressman, Seventh Edition McGrawHill International Edition.
2. Fundamentals of Software Engineering, Rajib Mall, Third Edition, PHI.

Reference Books :

1. Software Engineering, Ian Sommerville, Eighth Edition, Pearson education.
2. Software Engineering : A Primer, Waman S Jawadekar, Tata McGraw-Hill, 2008
3. Software Engineering, A Precise Approach, Pankaj Jalote, Wiley India,2010.
4. Software Engineering, Principles and Practices, Deepak Jain, Oxford University Press.
5. Software Engineering1: Abstraction and modeling, Diner Bjorner, Springer International edition, 2006.
6. Software Engineering2: Specification of systems and languages, Diner Bjorner, Springer
   International edition , 2006.
7. Software Engineering Foundations, Yingxu Wang, Auerbach Publications,2008.
8. Software Engineering Principles and Practice, Hans Van Vliet,3rd edition, John Wiley &Sons Ltd.
9. Software Engineering 3:Domains,Requirements,and Software Design, D.Bjorner, Springer
   International Edition.
10. Introduction to Software Engineering, R.J.Leach, CRC Press.

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (*Autonomous*)::Ananthapuramu

Department Of Computer Science & Engineering

M.Tech. I – I Sem.(SE)	T	P	C
	4	0	4

15D52101 :Service Oriented Architecture

Objectives:

The course should enable the student

- Understand SOA and evolution of SOA.
- Understand web services and primitive, contemporary SOA.
- Understand various service layers.
- Understand service-oriented analysis and design based on guidelines.

UNIT I

Introducing SOA: Fundamental SOA, Common Characteristics of Contemporary SOA, Common Tangible Benefits of SOA, Common Pitfalls of Adopting SOA.

The Evolution of SOA: An SOA Timeline, The Continuing Evolution of SOA, The Roots of SOA.

UNIT II

Web Services and Primitive SOA: The Web Services Frame Work, Services, Service Descriptions, Messaging.

Web Services and Contemporary SOA (Part I-Activity management and Composition):

Message Exchange Patterns, Service Activity, Coordination, Atomic Transactions, Orchestration, Choreography.

Web Services and Contemporary SOA (Part-II-Advanced Messaging, Metadata and Security): Addressing, Reliable Messaging, Correlation, Policies, Metadata exchange, Security.

(w.e.f 2015-16)

UNIT III

Principles of Service-Orientation: Service–Orientation and the Enterprise, Anatomy of SOA, Common Principles of Service–Orientation, Interrelation between Principles of Service-Orientation, Service Orientation and Object Orientation, Native Web Services Support for Principles of Service-Orientation.

Service Layers: Service-Orientation and Contemporary SOA, Service Layer abstraction, Application Service Layer, Business Service Layer, Orchestration Service Layer, Agnostic Services, Service Layer Configuration Scenarios.

UNIT IV

SOA Delivery Strategies: SOA Delivery Lifecycle Phases, The Top-Down Strategy, The Bottom-up Strategy, The Agile Strategy.

Service Oriented Analysis (Part I-Introduction): Introduction to Service Oriented Analysis, Benefits of a Business Centric SOA, Deriving Business Services.

Service Oriented Analysis (Part-II-Service Modelling): Service Modeling, Service Modelling Guidelines, Classifying Service Model Logic, Contrasting Service Modeling Approaches.

Service Oriented Design (Part I-Introduction): Introduction to Service-Oriented Design, WSDL Related XML Schema Language Basics, WSDL Language Basics, Service Interface Design Tools.

Service Oriented Design (Part II-SOA Composition Guidelines): SOA Composing Steps, Considerations for Choosing Service Layers, Considerations for Positioning Core SOA Standards, Considerations for Choosing SOA Extensions.

UNIT V

Service Oriented Design (Part III- Service Design): Service Design Overview, Entity-Centric Business Service Design, Application Service Design, Task-Centric Business Service Design, Service Design Guidelines.

Service Oriented Design (Part IV-Business Process Design): WS-BPEL Language Basics, WS- Coordination Overview, Service Oriented Business Process Design.

(w.e.f 2015-16)

TEXT BOOKS:

1. Service-Oriented Architecture-Concepts, Technology, and Design, Thomas Erl, Pearson Education.
2. Understanding SOA with Web Services, Eric Newcomer, Greg Lomow, Pearson Education.

REFERENCE BOOKS:

1. The Definitive guide to SOA, Jeff Davies & others, Apress, Dreamtech.
2. Java SOA Cook book, E.Hewitt, SPD.
3. SOA in Practice, N.M.Josuttis, SPD.
4. Applied SOA, M.Rosen and others, Wiley India pvt. Ltd.
5. Java Web Services Architecture, J.Mc Govern, and others, Morgan Kaufmann Publishers, Elsevier.
6. SOA for Enterprise Applications, Shankar.K, Wiley India Edition.
7. SOA-Based Enterprise Integration, W.Roshen, TMH.
8. SOA Security, K.Rama Rao, C.Prasad, dreamtech press.

(w.e.f 2015-16)

JNTUA College of Engineering (*Autonomous*)::Ananthapuramu

Department of Computer Science & Engineeting

M.Tech. I – I Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52102: Formal Methods of Software Engineering

OBJECTIVE:

Introduction to FMs used in software engineering. Elements of discrete mathematics, formal mechanisms for specifying and verifying the correctness, reliability and efficiency of software systems, finite state machines, regular expression, assertions, algebraic and model based specification techniques including case studies.

UNIT I

Introduction: Formal methods, The CICS Experience, The Z notation, The importance of Proof, Abstacion.

Propositional Logic: Proportional logic, Conjunction, Disjunction, Implication, Equivalence, Negation, Tautologies and Contradictions.

Predicate Logic: Predicate calculus, Quantifiers and declarations, Substitution, Universal Introduction and elimination, Existential introduction and elimination, Satisfaction and validity.

Equality and Definite Description: Equality, The one-point rule, Uniqueness and quantity, Definite description.

UNIT II

Sets: Membership and extension, Set comprehension, Power sets, Cartesian products, Union, intersection, and difference, Types.

Definitions: Declarations, Abbreviations, Generic abbreviations, Axiomatic definitions, Generic definitions, Sets and predicates.

Relations: Binary relations, Domain and range, Relational inverse, Relational composition, Closures.

Functions: Partial functions, Lambda notation, Functions on relations, Overriding, Properties of functions, Finite sets.

UNIT III

Sequences: Sequence notation, A model for sequences, Functions on sequences, Structural induction, Bags.

Free Types: The natural numbers, Free type definitions, Proof by induction, Primitive recursion, Consistency.

Schemas: The schema, Schemas as types, Schemas as declarations, Schemas as predicates, Renaming, Generic schemas.

Schema Operators: Conjunction, Decoration, Disjunction, Negation, Quantification and hiding, Composition.

UNIT IV

Promotion: Factoring operations, Promotion, Free and constrained promotion.

Preconditions: The initialisation theorem, Precondition investigation, Calculation and simplification, Structure and preconditions.

A File System: A Programming interface, Operations upon files, A more complete description, A file system, Formal analysis.

Data Refinement: Refinement, Relations and nondeterminism, Data types and data refinement, Simulations, Relaxing and unwinding.

UNIT V

Data Refinement and Schemas: Relations and schema operations, Forwards simulation, Backwards simulation.

Functional Refinement: Retrieve functions, Functional refinement, Calculating data refinements, Refining promotion.

Refinement Calculus : The specification statement, Assignment, Logical constants, Sequence composition, Conditional statements, Iteration.

(w.e.f 2015-16)

Text Book:

1. Jim **Woodcock and Jim Davies, "Using Z:** Specification, Refinement, and Proof", Prentice Hall (ISBN 0-13-948472-8), 1996.

Reference Books:

1. Diller, *Z An Introduction to Formal Methods* (2nd ed.), Wiley, 1994.
2. J. M. Spivey, "The Z Notation: A Reference Manual", Second Edition, Prentice Hall, 1992.

(w.e.f 2015-16)

JNTUA College of Engineering (*Autonomous)*::Ananthapuramu

Department of Computer Science & Engineeting

M.Tech. I – I Sem.(SE)	T	P	C
	4	0	4

15D52103: Software Requirements and Estimation

Objectives:

The course should enable the student

- To demonstrate knowledge of the distinction between critical and non- critical systems.
- To demonstrate the ability to manage a project including planning, scheduling and risk assessment/management.
- To author a software requirements document.
- To demonstrate an understanding of the proper contents of a software requirements document.
- To author a formal specification for a software system.
- To demonstrate an understanding of distributed system architectures and application architectures.
- To demonstrate an understanding of the differences between real-time and non-real time systems.
- To demonstrate proficiency in rapid software development techniques.
- To demonstrate proficiency in software development cost estimation
- To author a software testing plan.

UNIT-I :

Software Requirements: What And Why

Essential Software requirement, Good practices for requirements engineering, Improving requirements processes, Software requirements and risk management.

UNIT II:

Software Requirements Engineering

Requirements elicitation, requirements analysis documentation, review, elicitation techniques, analysis models, Software quality attributes, risk reduction through prototyping, setting requirements priorities, verifying requirements quality.

UNIT-III:

Software Requirements Modeling: Use Case Modeling, Analysis Models, Dataflow diagram, state transition diagram, class diagrams, Object analysis, Problem Frames.

Software Requirements Management: Requirements management Principles and practices, Requirements attributes, Change Management Process, Requirements Traceability Matrix, Links in requirements chain.

UNIT - IV

Software Estimation: Components of Software Estimations, Estimation methods, Problems associated with estimation, Key project factors that influence estimation.

Size Estimation: Two views of sizing, Function Point Analysis, Mark II FPA, Full Function Points, LOC Estimation, Conversion between size measures.

Effort, Schedule and Cost Estimation: What is Productivity? Estimation Factors, Approaches to Effort and Schedule Estimation, COCOMO II, Putnam Estimation Model, Algorithmic models, Cost Estimation.

UNIT-V

Requirements Management Tools: Benefits of using a requirements management tool, tool, commercial requirements management Rational Requisite pro, Caliber – RM, implementing requirements management automation.

Software Estimation Tools: Desirable features in software estimation tools, IFPUG, USC's COCOMO II, SLIM (Software Life Cycle Management) Tools.

(w.e.f 2015-16)

TEXT BOOKS:

1. Software Requirements by Karl E. Weigers,Microsoft Press.

2. Software Requirements and Estimation by *Rajesh Naik and Swapna Kishore*, Tata Mc Graw Hill.

REFERENCES:

1. Managing Software Requirements, Dean Leffingwell & Don Widrig, Pearson Education,2003.

2. Mastering the requirements process, second edition, Suzanne Robertson & James Robertson, Pearson Education, 2006.

3. Estimating Software Costs, Second edition, Capers Jones, Tata McGraw-Hill, 2007.

4. Practical Software Estimation, M.A. Parthasarathy, Pearson Education, 2007.

5. Measuring the software process, William A. Florac & Anita D. Carleton, Pearson Education,1999.

(w.e.f 2015-16)

JNTUA College of Engineering (*Autonomous)*::Ananthapuramu

Department of Computer Science & Engineering

	T	P	C
M.Tech. I – I Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52104: Software Metrics and Reuse

ELECTIVE -I

Objectives:

The course should enable the student

- To understand why measurement is important
- To know how to extract, when and where to apply relevant metrics
- To understand the importance of measurement in software engineering
- To describe and compare the different metrics that can be used for measuring software
- To understand the important factors that affect the measurement of software
- To explain the benefits of software reuse and some reuse problems
- To discuss several different ways to implement software reuse
- To discuss software reuse technologies like COTS , CBSE

UNIT - I

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.

UNIT - II

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.

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.

(w.e.f 2015-16)

UNIT - III

Measuring external product attributes: Modeling software quality, measuring aspects of software quality.

Metrics for object-oriented systems: The intent of object-oriented metrics, distinguishing characteristics of object-oriented metrics, various object-oriented metric suites – LK suite, CK suite and MOOD metrics.

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

UNIT - IV

Introduction: Software Reuse and 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, Aspects of Software Reuse, Organizational Aspects, Technical Aspects and Economic Aspects.

Programming Paradigm and Reusability: Usability Attributes, Representation and Modeling Paradigms, Abstraction and Composition in development paradigm.

UNIT - V

Object-Oriented Domain Engineering: Abstraction and Parameterization Techniques, Composition Techniques in Object Orientation.

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.

Text books:

1. Norman E. Fenton and Shari Lawrence Pfleeger; Software Metrics – A Rigorous and Practical Approach, Thomson Asia Pte., Ltd, Singapore.

2. Stephen H. Kan; Metrics and Models in Software Quality Engineering, Addison Wesley, New York.

3. Reuse Based Software Engineering Techniques, Organization and Measurement by Hafedh Mili, Ali Mili, Sherif Yacoub and Edward Addy, John Wiley & Sons Inc

4. The Three Rs of Software Automation: Re-engineering, Repository, Reusability by Carma McClure, Prentice Hall New Jersey

References:

1. K. H. Möller and D. J. Paulish; Software Metrics - A Practitioner's Guide to Improved Product Development, Chapman and Hall, London.
2. Mark Lorenz and Jeff Kidd; Object-Oriented Software Metrics, Prentice Hall, New York.
3. McClure, Carma L. Software reuse techniques : adding reuse to the system development process / : Prentice Hall
4. Poulin, Jeffrey S. Measuring software reuse : principles, practices, and economic models / Jeffrey S. Poulin. Reading, Mass. : Addison-Wesley

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (*AUTONOMOUS*) : : ANANTAPUR

Department Of Computer Science & Engineering

	T	P	C
M.Tech. I – I Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52105: Reverse Engineering
ELECTIVE-I

Objectives:

- To discuss the problems of reliability specification and measurement
- To introduce reliability metrics and to discuss their use in reliability specification
- To describe the statistical testing process
- To show how reliability predications may be made from statistical test results.

UNIT I

Foundations: What is Reverse Engineering, Software Reverse Engineering, Reverse Applications, Low Level Software, The Reversing Process, The Tools, Is Reversing Legal, Code Samples & Tools.

Object Flow Graph: Abstract Language, Object Flow Graph, Containers, Flow Propagation Algorithm, Object Sensitivity, The elib Program.

Low Level Software: High Level Perspectives, Low Level Perspectives, Assembly Language, A Primer on Compilers and Compilation, Execution Environments.

UNIT II

Reversing Tools: Different Reversing Approaches, Disassemblers, Debuggers, Decompilers, System-Monitoring Tools, Patching Tools, Miscellaneous Reversing Tools.

UNIT III

Beyond the Documentation: Reversing and Interoperability, Laying The Ground Rules, Locating Undocumented APIs, Case Study.

UNIT IV

Class Diagram: Class Diagram Recovery, Declared Vs Actual Types, Containers, The elib Program.

Object Diagram: The Object Diagram, Object Sensitivity, Dynamic Analysis, The elib Program. **Interaction Diagram:** Interaction Diagram, Interaction Diagram, Intreraction Diagram Recovery, Dynamic Analysis, The elib Program.

State Diagram: State Diagram, Abstract Interpretation, State Diagram Recovery, The elib Program.

(w.e.f 2015-16)

UNIT V

Package Diagram : Package Diagram Recovery, Clustering, Concept Analysis, The elib Program, Tool Architecture, The elib Program, Perspectives.

Reversing Malware : Types of malware, Sticky software, Future malware, Uses of malware, Malware vulnerability, Polymorphism, Metamorphism, Establishing a secure environment. **Antireversing Techniques :** Why anti reversing?, Basic approaches to anti reversing, Eliminating symbolic information, Code encryption, Active anti debugger techniques, Confusing Disassemblers, Code obfuscation, Control flow transformations, Data transformations.

TEXT BOOKS:

1. Reverse Engineering of Object Oriented Code Paolo Tonella by Alessandra Potrich.
2. Reversing: Secrets of Reverse Engineering by Eldad Eilam.

(w.e.f 2015-16)

JNTUA College of Engineering (*Autonomous*)::Ananthapuramu

Department of Computer Science & Engineering

M.Tech. I – I Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52106: Software Architecture and Design Patterns

ELECTIVE -I

Objectives:

The course should enable the student

- To understand interrelationships, principles and guidelines governing architecture and evolution over time.
- To understand various architectural styles of software systems.
- To understand design patterns and their underlying object oriented concepts.
- To understand implementation of design patterns and providing solutions to real world software design problems.
- To understand patterns with each other and understanding the consequences of combining patterns on the overall quality of a system.

UNIT I

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

Analyzing Architectures

(w.e.f 2015-16)

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

Patterns

Pattern Description, Organizing catalogs, role in solving design problems, Selection and usage.

Creational and Structural Patterns

Abstract factory, builder, factory method, prototype, singleton, adapter, bridge, composite, façade, flyweight.

UNIT IV

Behavioral Patterns

Chain of responsibility, command, Interpreter, iterator, mediator, memento, observer, state, strategy, template method, visitor.

UNIT V

Case Studies

A-7E – A case study in utilizing architectural structures, The World Wide Web - a case study in

interoperability, Air Traffic Control – a case study in designing for high availability, Celsius Tech – a case study in product line development.

A Case Study (Designing a Document Editor): Design Problems, Document Structure, Formatting, Embellishing the User Interface, Supporting Multiple Look-and-Feel Standards, Supporting Multiple Window Systems, User Operations, Spelling Checking and Hyphenation.

(w.e.f 2015-16)

TEXT BOOKS:

1. Software Architecture in Practice, second edition, Len Bass, Paul Clements & Rick Kazman, Pearson Education, 2003.
2. Design Patterns, Erich Gamma, Pearson Education, 1995.

REFERENCE BOOKS:

1. Beyond Software architecture, Luke Hohmann, Addison wesley, 2003.
2. Software architecture, David M. Dikel, David Kane and James R. Wilson, Prentice Hall PTR, 2001
3. Software Design, David Budgen, second edition, Pearson education, 2003
4. Head First Design patterns, Eric Freeman & Elisabeth Freeman, O'REILLY, 2007.
5. Design Patterns in Java, Steven John Metsker & William C. Wake, Pearson education, 2006
6. J2EE Patterns, Deepak Alur, John Crupi & Dan Malks, Pearson education, 2003.
7. Design Patterns in C#, Steven John metsker, Pearson education, 2004.
8. Pattern Oriented Software Architecture, F.Buschmann & others, John Wiley & Sons.

(w.e.f 2015-16)

JNTUA College of Engineering (*Autonomous*)::Ananthapuramu

Department of Computer Science & Engineering

M.Tech. I – I Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52107: Agile Methodologies

ELECTIVE -II

Objectives:

The course should enable the student

- To understand how an iterative, incremental development process leads to faster delivery of more useful software
- To understand the essence of agile development methods
- To understand the principles and practices of extreme programming

UNIT I

Why Agile? , How to be Agile, Understanding XP, Values and Principles, Improve the Process, Eliminate Waste, Deliver Value.

UNIT II

Practicing XP-Thinking, Pair Programming, Energized Work, Informative Workspace, Root-Cause Analysis, Retrospectives, Collaborating, Sit Together, Real Customer Involvement, Ubiquitous Language, Stand-Up Meetings, Coding Standards, Iteration Demo, Reporting.

UNIT III

Releasing-Done Done, No Bugs, Version Control, Ten-Minute Build, Continuous Integration, Collective Code Ownership, Documentation.

UNIT IV

Planning-Vision, Release Planning, Risk Management, Iteration Planning, Stories, Estimating.

UNIT V

Developing-Incremental Requirements, Customer Tests, Test- Driven Development, Refactoring, Incremental Design and Architecture, Spike Solutions, Performance Optimization.

(w.e.f 2015-16)

Text Books:

1. James Shore and Shane Warden, " The Art of Agile Development", O'REILLY, 2007.

References:

1. **Robert C. Martin, "Agile Software Development, Principles, Patterns, and Practices" , PHI, 2002.**

2. Angel Medinilla, "Agile Management: Leadership in an Agile Environment", Springer, 2012.

3. Bhuvan Unhelkar, "The Art of Agile Practice: A Composite Approach for Projects and Organizations", CRC Press.

4. Jim Highsmith, "Agile Project Management", Pearson education, 2004.

(w.e.f 2015-16)

JNTUA College of Engineering (*Autonomous*)::Ananthapuramu

Department of Computer Science & Engineering

M.Tech. I – I Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52108: Protocol Software Engineering

ELECTIVE -II

Objectives:

The course should enable the student

- To identify fundamental concepts of communication protocols such an encapsulation/decapsulation and overhead; switching, routing, fragmentation, IP addressing, transport layer, and IP/MPLS based services.
- To use the 'wireshark' packet sniffing program
- To combine 1 and 2 above in the implementation, analysis, performance measurement and troubleshooting of networks
- To recognize basic configuration and measurement procedures on an industry-standard service router platform.

UNIT I

Network Reference Model: Layered Architecture, Network Services and Interfaces, Protocol Functions, OSI Model, TCP/IP Protocol Suite, Application Protocols.

Formal Specification: Formal Specification in the Software Process, Sub-system Interface

Specification, Behavioural Specification. Protocol Specification: Components of Protocol

to be Specified, Communication Service Specification, Protocol Entity Specification, Interface Specifications, Interactions, Multimedia Protocol Specifications, Internet Protocol Specifications.

(w.e.f 2015-16)

UNIT II

Architectural Design: Architectural Design Decisions, System Organisation, Modular Decomposition Styles, Control Styles, Reference Architectures. Distributed Systems Architectures: Multiprocessor Architectures, Client-server Architectures, Distributed Object Architectures, Inter-organisational Distributed Computing.

UNIT III

Formal Description Testing for Protocol Specification, Extended State Transition Language, Language for temporal Ordering Specfication, Format and Protocol Languages.

SDL: A Protocol Specification Language: SDL, Examples of SDL Based Protocol Specifications, Other Protocol Specificaiton Languages.

Protocol Verification/Validation: Protocol Verification, Verification of a Protocol Using Finite State Machines, Protocol Validation, Protocol Design Errors, Protocol Validation Approaches, SDL Based Protocol Verification, SDL Based Protocol Validation.

UNIT IV

Protocol Conformance Testing: Conformance Testing, Conformance Testing Methodology and Framework, Conformance Test Architectures, Test Sequence Generation Methods, Distributed Architecture by Local Methods, Conformance Testing with TTCN, Conformance Testing in Systems with Semicontrollable Interfaces, Conformance Testing of RIP, Multimedia Applications Testing, SDL Based Tools for Conformance Testing, SDL Based Conformance Testing of MPLS.

UNIT V

Protocol Performance Testing: Performance Testing, SDL Based Performance Testing of

TCP, SDL Based Performance Testing of OSPF, Interoperability Testing, SDL Based Interoperability Testing of CSMA/CD and CSMA/CA Protocol Using Bridge, Scalability

Testing. Protocol Synthesis: Protocol Synthesis, Interactive Synthesis Algorithm, Automatic Synthesis Algorithm, Automatic Synthesis of SDL from MSC, Protocol Resynthesis.

Testing Models, PICS and PIX IT, Abstract Test Methods, Simulation Based Evaluation of Conformance Testing Methodologies. Examples include actual implementation like OSINET, based on ESTELLE tools and TTCU, PICS, PIX IT for OSINET.

TEXT BOOKS:

1. Communication Protocol Engineering, Pallapa Venkataram, Sunilkumar S. Manvi, PHI.

2. Protocol Specification for OSI*1 , Gregor V. Bochmann, University of Motreal, Montreal, Quebec, Canada.

3. ASN.1: Communication Between Heterogeneous Systems, Olivier Dubuisson, Morgan Kaufmann.

REFERENCES:

1. Tools for Protocols Driven by Formal Specifications, Harry Rudin.

2. Network Protocols and Tools to help produce them*, Harry Rudin, IBM

Research Division, Zurich Research Laboratory, 8803 Ruschlikon, Switzerland.

(w.e.f 2015-16)

JNTUA College of Engineering (*Autonomous)*::Ananthapuramu

Department of Computer Science & Engineeting

M.Tech. I – I Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52109: Component Based Software Engineering

ELECTIVE-II

Objectives:

- To understand the essentials of component-based software engineering
- To know the main characteristics of components and component models
- To be aware of software development processes for component-based systems
- To be aware of the mutual relations between software architecture and component models

UNIT I

Component definition - Definition of a Software Component and its elements, The Component Industry Metaphor, Component Models and Component Services, An example specification for implementing a temperature regulator Software Component.

The Case for Components- The Business Case for components, COTS Myths and Other Lessons Learned in Component-Based Software Development.

UNIT II

Planning Team Roles for CBD, Common High-Risk Mistakes, CBSE Success Factors: Integrating Architecture, Process, and Organization.

Software Engineering Practices - Practices of Software Engineering, From Subroutines to

Subsystems: Component-Based Software Development, Status of CBSE in Europe.

UNIT III

The Design of Software Component Infrastructures - Software Components and the UML, Component Infrastructures, Business Components, Components and Connectors, An OPEN process for CBD, Designing Models of Modularity and Integration. Software Architecture, Software Architecture Design Principles, Product-Line Architectures.

UNIT IV

The Management of Component-Based Software Systems - Measurement and Metrics for Software Components, Implementing a Practical Reuse Program for Software Components, Selecting the Right COTS Software, Building instead of Buying, Software Component Project Management, The Trouble with Testing Components, Configuration Management and Component Libraries, The Evolution, Maintenance, and Management of CBS.

UNIT V

Component Technologies - Overview of the CORBA Component Model, Overview of COM+, Overview of the EJB Component Model, Bonobo and Free Software GNOME Components, Choosing between COM+, EJB, and CCM, Software Agents as Next Generation Software Components.

TEXT BOOKS:

1. Component - Based Software Engineering, G.T. Heineman and W.T. Councill, Addison-Wesley, Pearson Education.

REFERENCE BOOKS:

1. Component Software, C.Szyperski, D.Gruntz and S.Murer, Pearson Education.

2. Software Engineering, Roger S. Pressman, 6th edition, Tata McGraw-Hill.

3. Software Engineering, Ian Sommerville, seventh edition, Pearson education, 2004.

4. Software Engineering Principles and Practice, Hans Van Vliet, 3rd edition, Wiley India edition.

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (*Autonomous*)::Ananthapuramu

Department Of Computer Science & Engineering

M.Tech. I – I Sem.(SE)	T	P	C
	0	4	2

15D52110: Software Engineering & Service Oriented Architecture Lab

Student is expected to complete the following experiments as a part of laboratory work.

1. Develop at least 5 components such as Order Processing, Payment Processing, etc., using .NET component technology.
2. Develop at least 5 components such as Order Processing, Payment Processing, etc., using EJB Component Technology.
3. Invoke .NET components as web services.
4. Invoke EJB components as web services.
5. Develop a Service Orchestration Engine (workflow) using WS-BPEL and Implement Service Composition. For Example, a business process for planning business travels will invoke several services. This process will invoke several airline companies (such as American Airlines, Delta Airlines etc.) to check the airfare price and buy at the lowest price.
6. Develop a J2EE client to access a .NET web service.
7. Develop a .NET client to access a J2EE web service.

Write problem definition, overall description, specific requirements, front – end description, back – end description and draw the data flow diagrams & UML diagram for following CASE Studies.

1. Library Management System
2. Automated banking system
3. Airline reservation system
4. Employee management application
5. Hospital management Application

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) : : ANANTAPURAMU

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52201: Software Project Planning &Management

Objectives:

The student should be able to:

- Describe and determine the purpose and importance of project management from the perspectives of planning, tracking and completion of project.
- Compare and differentiate organization structures and project structures.
- To discuss the various aspects of project management
- To understand the tasks in software project management
- To describe the requirements of a project plan

UNIT I

Manage Your People: Managing project culture, Managing Good People, Making Good People Better, Leading Good People.

Implement Your Process: Putting a process in place, implementing a Process, Adopting a Process.

Leverage Your Tools: Choosing Tools, Training to Use Tools, leveraging Tools.

Use Your Measurements: Selecting Measurements, Planning Measurements, Leveraging Measurements.

UNIT II

Form Your Vision: Analyzing Stakeholders, Balancing Project Needs, Ascending Project Risks, Specifying Project Payoffs, Specifying and Communicating a Project Vision.

Organize Your Resources: Identifying Hardware, Identifying Software, Identifying Support.

Sketch Your Schedule: Estimating Project Size and Effort, Scheduling Immovable Milestones, Scheduling Synchronization Points, Facilitating Communication.

Write Your Plan: Organizing the Plan, Covering all the bases, Reviewing the Plan.

(w.e.f 2015-16)

UNIT III

Roll Out Your Roles : Identifying Roles, Matching People to Roles, Highlighting Commitments and Dependencies.

Schedule Your Schedule: Identifying and Scheduling Tasks, Assigning Tasks to Roles, Creating a Backup Plan, Examining a Case Study.

Leaving the Starting Line: Directing the Team, Implementing the Technology, Capturing the Measurements.

UNIT IV

Monitor Your Project: Gathering Information, Understanding the Information, Avoiding Problems, Finding Solutions.

Reschedule Your Schedule: Making the Schedule Important, Knowing when the Schedule Slipped, Rescheduling Correctly, Examining a Case Study.

Engineer a Great Product: Requiring Your Requirements, Designing in Quality, Implementing Smartly, Testing Effectively.

UNIT V

Deliver Your System: Planning to Finish, Finishing a Plan Supporting a Product Examining a Case Study.

Assess your Project: Planning a Project Assessment, Analyzing Measurements, Presenting the Assessments Results, Examining a Case Study.

Text books:

1. Joel Henry, "Software Project Management A Real Word to Guide to Success", Pearson Education, 2004.

Reference Books:

1. Walker Royce, "Software Project Management", Pearson Education, 1998.
2. Pankaj Jalote, "Software Project Management in Practice", Addison-Wesley Professional, 2002.
3. Bob Hughes & Mike Cotterell, "Software Project Management", fourth edition,Tata Mc-Graw Hill,2006
4. Andrew Stellman & Jennifer Greene, "Applied Software Project Management, O'Reilly, 2006.
5. Richard H. Thayer & Edward Yourdon, "Software Engineering Project Managent" , second edition, Wiley India, 2004.

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) : : ANANTAPURAMU

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	T	P	C
	4	0	4

15D51203: Software Quality Assurance & Testing

Objectives:

The student should be able to:

- Understand software testing and quality assurance as a fundamental component of software life cycle
- Define the scope of software testing & quality assurance projects
- Efficiently perform testing & quality assurance activities using modern software tools
- Estimate cost of a testing & quality assurance project and manage budgets
- Prepare test plans and schedules for a testing & quality assurance project
- Develop testing & quality assurance project staffing requirements
- Effectively manage a testing & quality assurance project

UNIT I

Introduction to software quality, Challenges, Objectives, Quality Factors, Components of SQA, Contract review, Development and quality Plans, SQA Components in Project Life Cycle, SQA Defect Removal Policies, Reviews.

UNIT II

Software Testing Strategy and Environment: Minimizing Risks, Writing a Policy for Software Testing, Economics of Testing, Testing-an organizational issue, Management Support for Software Testing, Building a Structured Approach to Software Testing, Developing a Test Strategy.

Building Software Testing Process: Software Testing Guidelines, Workbench Concept, Customizing the Software Testing Process, Process Preparation Checklist.

UNIT III

Software Testing Techniques: Dynamic Testing – Black Box Testing Techniques, White Box Testing Techniques, Static Testing, Validation Activities, Regression Testing.

Software Testing Tools: Selecting and Installing Software Testing tools

Automation and Testing Tools: Load Runner, Win runner and Rational Testing Tools, Silk test, Java Testing Tools, JMetra, JUNIT and Cactus.

UNIT IV

Seven Step Testing Process–I: Overview of the Software Testing Process, Organizing of Testing, Developing the Test Plan, Verification Testing, Validation Testing.

(w.e.f 2015-16)

UNIT V

Seven Step Testing Process-II: Analyzing and Reporting Test results, Acceptance and Operational Testing, Post-Implementation Analysis

Specialized Testing Responsibilities: Software Development Methodologies, Testing Client/Server Systems.

TEXT BOOKS:

1. Effective Methods for Software Testing, Third edition, William E. Perry, Wiley India, 2009
2. Software Testing – Principles and Practices, Naresh Chauhan, Oxford University Press, 2010.
3. Software Quality Assurance – From Theory to Implementation, Daniel Galin, Pearson Education, 2009.

Reference Books:

1. Testing Computer Software, Cem Kaner, Jack Falk, Hung Quoc Nguyen, Wiley India, rp2012.
2. Software Testing – Principles, Techniques and Tools, *M.G.Limaye*, Tata McGraw-Hill, 2009.
3. Software Testing - A Craftsman's approach, *Paul C. Jorgensen*, Third edition, Auerbach Publications, 2010.
4. Software Quality Assurance, *Milind Limaye*, Tata McGraw-Hill, 2011.

5. Software Quality – Theory and Management, *Alan C. Gillies*, Second edition, Cengage Learning,2009.

6. Software Quality – A Practitioner's approach, *Kamna Malik, Praveen Choudhary,* Tata McGraw-Hill, 2008.

7. Software Quality Models and Project Management in a Nutshell, *Shailesh Mehta*, Shroff Publishers and Distributors, 2010.

8. Software Quality Engineering – Testing, Quality Assurance and Quantifiable Improvement, *Jeff Tian*, Wiley India, 2006.

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) : : ANANTAPURAMU

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52202: Secure Software Engineering

Objectives:

- Students will demonstrate knowledge of the distinction between critical and non-critical systems.
- Students will demonstrate the ability to manage a project including planning, scheduling and risk assessment/management.
- Students will demonstrate an understanding of the proper contents of a software requirements document for secure software engineering.
- Students will author a formal specification for secure software systems.
- Students will demonstrate an understanding of distributed system architectures and application architectures.
- Students will demonstrate an understanding of the differences between real-time and non-real time systems.
- Students will be able to identify specific components of a software design that can be targeted for reuse.
- Students will author a software testing plan and metrics for secure software engineering.

UNIT I

Why Is Security a Software Issue?

Introduction, The problem, Software assurance and software security, Threats to software security, Sources of software insecurity, The benefits of detecting software security defects early, Managing secure software development.

What Makes Software Secure?

Defining properties of secure software, How to influence the security properties of software, How to assert and specify desired security properties.

(w.e.f 2015-16)

UNIT II

Requirements Engineering for Secure Software

Introduction, Misuse and Abuse Cases, The SQUARE process model: SQUARE sample outputs, Requirements elicitation, Requirements Prioritization.

Secure Software Architecture and Design

Introduction, Software security practices for architecture and design: Architectural risk analysis. Software security knowledge for architecture and design: Security principles, Security guidelines, and Attack patterns.

UNIT III

Considerations for Secure Coding and Testing

Introduction, Code analysis, Coding practices, Software security testing, Security testing considerations throughout the SDLC.

Security and Complexity: System Assembly Challenges

Introduction, Security failures, Functional and attacker perspectives for security analysis, System complexity drivers and security, Deep technical problem complexity.

UNIT IV

Governance, and Managing for More Secure Software

Introduction, Governance and security, Adopting an enterprise software security framework, How much security is enough?, Security and project management, maturity of practice.

UNIT V

Security Metrics

Defining security metrics, Diagnosing problems and measuring technical security, Analysis techniques, Organize, aggregate, and analyze data to bring out key insights.

(w.e.f 2015-16)

TEXT BOOKS

1. Software Security Engineering: A Guide for Project Managers, by Julia H. Allen, Sean Barnum, Robert J. Ellison, Gary McGraw, Nancy R. Mead, Addison-Wesley , 1st edition, 2008.

2. Security Metrics: Replacing Fear, Uncertainty, and Doubt , by Andrew Jaquith, Addison-Wesley , 1st edition , 2007.

REFERENCES

1. Integrating Security and Software Engineering: Advances and Future Vision, by Haralambos Mouratidis, Paolo Giorgini, IGI Global, 2006.

2. Software Security: Building Security In , by Gary McGraw , Addison-Wesley, 2006

3. The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities, by Mark Dowd, John McDonald, Justin Schuh, Addison-Wesley, 1st edition, 2006

4. Building Secure Software: How to Avoid Security Problems the Right Way by John Viega,Gary McGraw, Addison-Wesley, 2001

5. Writing Secure Code, by M. Howard, D. LeBlanc, Microsoft Press, 2nd Edition, 2003.

6. Exploiting Software: How to break code, by G. Hoglund, G. McGraw, Addison Wesley, 2004.

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) : : ANANTAPURAMU

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52203: Model Driven Software Development

Objectives:

- Develop enabling technologies for supporting model driven engineering approaches to software development
- Develop improved techniques and tool support for using executable specifications and model-based testing to better capture, manage and test software against its requirements
- Better integrate social networking tools and techniques into the software development process to improve the efficiency of collaborative and community development of software
- Better support "early phase" decision making by providing tools and techniques to assess non functional requirement adherence at early stages in the software development process.

UNIT I

MDSD Basic Terminology

Goals of MDSD, MDSD Approach, Overview of MDA concepts, Architecture-Centric MDSD, Common MDSD concepts and terminology, Model-Driven Architecture, Generative Programming, Software factories, Model-Integrated computing, Language-Oriented Programming, Domain specific modeling.

UNIT II

Metamodeling

What is Metamodeling?, Metalevels vs. Level of Abstraction, MOF and UML, Extending UML, UML profiles, Metamodeling and OCL, Examples, Tool-supported Model validation, Metamodeling and Behavior, Pitfalls in Metamodeling, MDSD classification.

UNIT III

Model Transformation with QVT

History, M2M language requirements, Overall Architecture, An Example Transformation, The OMG standardization Process and Tool Availability, Assesment.

MDSD Tools:Roles, Architecture, Selection Criteria, and Pointers

Role of Tools in the Development Process, Tool Architecture and selection criteria, pointers.

The MDA Standard: Goals, Core concepts

UNIT IV

MDSD Process Building Blocks and Best Practices

Introduction, Separation between Application and domain Architecture Development, Two track Iterative Development, Target Architecture Development Process, Product-line Engineering.

Testing

Test Types, Tests in Model-driven Application Development, Testing the Domain Architecture

Versioning

What is Versioned? Projects and Dependencies, The structure of Application Projects, Version management and Build Process for mixed files, Modeling in a team and versioning of partial models

UNIT V

Quality : Quality in Model Driven Engineering

Case study: Embedded Component Infrastructures

Overview, Product-Line Engineering, Modeling, Implementation of Components, Generator Adaptation, Code Generation.

(w.e.f 2015-16)

TEXT BOOKS:

1. Model-Drievn Software Development-Technology, Engineering, Management by Thomos Stahl, Markus Volter, jul 2006, John Wiley & Sons.
2. Model-Driven Software Development: Integrating Quality Assurance by Jorg Rech, Christian Bunse,2008,Information Science Publishing.

REFERENCE BOOKS :

1. Model-Driven Software Development by Sami Beydeda Matthias Book , Volker Gruhn, Springer.
2. Model Driven Systems Development with Rational Products By Brian Nolan, Barclay Brown, Dr. Laurent Balmelli, Et Al Tim Bohn, 2008,IBM.
3. Model Driven Development with Executable UML by Dragan Milicev, 2009,Wilei India pvt Ltd.
4. Model Driven Software Development by Kevin Lano, Apr 2009, Ci Business Press.

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) : : ANANTAPURAMU

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52204: Software Agents

ELECTIVE-III

Objectives:

- To learn the principles and fundamentals of designing agents
- To study the architecture design of different agents.
- To learn to do detailed design of the agents
- To understand user interaction with agents
- To explore the role of agents in assisting the users in day to day activities
-

UNIT I INTRODUCTION

Agents and Multi Agent Systems- Intelligent Agent- Concepts of Building Agent – Situated Agents – Proactive and Reactive agents- Challenging Agent Environment- Social Agents- Agent Execution Cycle- Prometheus Methodology- Guidelines for using Prometheus- Agent Oriented Methodologies- System Specification – Goal Specification – Functionalities – Scenario Development – Interface Description – Checking for Completeness and Consistency.

UNIT II ARCHITECTURAL DESIGN

Agent Types - Grouping Functionalities - Agent Coupling - Develop Agent Descriptors - Interactions - Interaction Diagram from Scenarios- Interaction Protocol from Interaction Diagram- Develop Protocol and Message Descriptors –Architectural Design - Identifying the Boundaries of Agent System – Percepts and Action - Shared Data Objects – System Overview – Checking for Completeness and Consistency.

(w.e.f 2015-16)

UNIT III DETAILED DESIGN

Capability Diagrams – Sub Tasks - Alternative Programs – Events and Messages – Action and Percept Detailed Design – Data – Develop and Refine Descriptors – Missing or Redundant Items- Consistency between Artifacts – Important Scenarios- Implementing Agent Systems - Agent Platform – JACK

UNIT IV AGENTS AND USER EXPERIENCE

Interact with Agents - Agents from Direct Manipulation to Delegation – Interface Agents - Designing Agents - Direct Manipulation versus Agents- Agents for Information Sharing and Coordination- Agents that Reduce Work and Information Overload - KidSim: Programming Agents without a Programming Language.

UNIT V AGENTS FOR INTELLIGENT ASSISTANCE

Computer Characters- Software Agents for Cooperative Learning – Integrated Agents- Agent Oriented Programming- KQML as an Agent Communication Language- Agent Based Framework for Interoperability - Agents for Information Gathering - KAoS- Communicative Actions for Artificial Agents – Mobile Agents.

REFERENCES:

1. Lin Padgham and Michael Winikoff, "Developing Intelligent Agent Systems: A Practical Guide", John Wiley & sons Publication, 2004.

2. Jeffrey M. Bradshaw, "Software Agents", MIT Press , 1997.

3. Steven F. RailsBack and Volker Grimm, "Agent-Based and Individual Based modeling:APractical Introduction", Princeton University Press, 2012.

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) : : ANANTAPURAMU

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52205: Software Evolution and Maintenance

ELECTIVE-III

Objectives:

This subject introduces basic concepts of maintenance and how the concept of system evolution fits into maintenance, presents different technical and managerial problems of maintenance, addresses the formal types of maintenance, and discusses standard maintenance processes.

UNIT- I

Introduction and Roadmap: History and Challenges of Software Evolution, Understanding and Analysing Software Evolution: Identifying and Removing Software Clones, Analysing Software Repositories to Understand Software Evolution.

UNIT-II

Novel Trends in Software Evolution: Evolution Issues in Aspect-Oriented Programming, Software Architecture Evolution, Empirical Studies of Open Source Evolution.

UNIT- III

Software Maintenance and Organizational Health and Fitness, Problem Management within Corrective Maintenance, The Impact of eXtreme Programming on maintenance.

UNIT- IV

Patterns in software Maintenance: Learning from Experience, Enhancing Software Maintainability by Unifying and Integrating Standards, Requirements Risk and Maintainability.

(w.e.f 2015-16)

UNIT- V

Software Maintenance Cost Estimation, A Methodology for Software Maintenance, Environment for Managing Software Maintenance Projects.

Text Books:

1. Software Evolution By Tom Mens and Serge Demeyer, Springer,2008.
2. Polo, M., 2003, Advances in software maintenance management : technologies and solutions Hershey, PA: Idea Group Pub.

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) : : ANANTAPURAMU

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52206: Software Process Management

ELECTIVE-III

Objectives:

- To make predictions and commitments relative to the products it produces.
- To understand Effective measurement processes
- To develop achievable plans for producing and delivering products and services
- To identify important events and trends and that effectively separate signals from noise are invaluable in guiding software organizations to informed decisions.

UNIT-I

SOFTWARE PROCESS MATURITY:

A SOFTWARE MATURITY FRAMEWORK: Software Process Improvement, Process Maturity Levels, People in the Optimizing Process, The Need for the Optimizing Process

THE PRINCIPLES OF SOFTWARE PROCESS CHANGE: Process in Perspective, The Six Basic Principles, Some Common Misconceptions about the Software Process, A Strategy for Implementing Software Process Change

SOFTWARE PROCESS ASSESSMENT: Assessment Overview, Assessment Phases, Five Assessment Principles, The Assessment Process, Assessment Conduct, implementation Considerations

THE INITIAL PROCESS: The Nature of the Initial Process, A Case Study of a Chaotic Project, why Software Organizations are Chaotic, Software Process Entropy, The Way Out.

UNIT-II

THE REPEATABLE PROCESS:

MANAGING SOFTWARE ORGANIZATIONS: Commitment Discipline, The Management System, Establishing a Project management System

THE PROJECT PLAN: Project Planning Principles, Project Plan Contents, Size Measures, Estimating, Productivity Factors, Scheduling, Project Tracking, The Development Plan, Planning Models, Final Considerations.

SOFTWARE CONFIGURATION MANAGEMENT – PART 1: The Need for Configuration Management, Software Product Nomenclature, Basic configuration Management Functions, Baselines, Configuration Management Responsibilities, The need for Automated Tools.

UNIT-III

THE DEFINED PROCESS:

SOFTWARE STANDARDS: Definitions, The Reasons for Software Standards, Benefits of Standards, Examples of Some Major Standards, Establishing Software Standards, Standards Versus Guidelines

SOFTWARE INSPECTIONS: Types of Reviews, Inspection Objectives, Basic Inspection Principles, The Conduct of Inspections, Inspection Training, Reports and Tracking, Other Considerations, Initiating an Inspection Program, Future Directions

SOFTWARE TESTING: Software Testing Principles, Types of Software Tests, Test Planning, Test Development, Test Execution and Reporting, Test Tools and Methods, Real-Time Testing, The Test Organization.

UNIT-IV

SOFTWARE CONFIGURATION MANAGEMENT (CONTINUED): The Software Configuration Management Plan, Software Configuration, Management Questions, SCM Support Functions, The Requirements Phase, Design Control, The Implementation Phase, Operational Data, The Test Phase, SCM for Tools, Configuration Accounting, The Software Configuration Audit

DEFINING THE SOFTWARE PROCESS: Process Standards, Definitions, Levels of Software Process Models, Prescriptive and Descriptive Uses of Models, A Software Process Architecture, Critical Software Process Issues, A Preliminary Process Architecture, Larger Process Models, Detailed Process Models, Entity Process Models, Process Model Views, Establishing and Using a Process Definition, Basic Process Guidelines

THE SOFTWARE ENGINEERING PROCESS GROUP: Changing the Software Process, The Role of the SEPG, Establishing Standards, The Process Database, Technology Insertion

Focal Point, Education and Training, Process Consultation, Process Status and Assessment, Establishing the SEPG

THE MANAGED PROCESS:

DATA GATHERING AND ANALYSIS: The Principles of Data Gathering, The Data Gathering Process, Software Measures, Data Analysis.

UNIT- V

MANAGING SOFTWARE QUALITY: The Quality Management Paradigm, Quality Examples, Quality Motivation, Measurement Criteria, Establishing a Software Quality Program, Estimating Software Quality, Removal Efficiency, Quality Goals, Quality Plans, Tracking and Controlling Software Quality

THE OPTIMIZING PROCESS:

DEFECT PREVENTION: Defect Prevention Not a New Idea, The Principles of Software Defect Prevention, Process Changes for Defect Prevention, Defect Prevention Considerations, Management's Role.

Textbooks:

1. Watts S. Humphrey, "**Managing the Software Process", Pearson Edocation.**

Reference Books:

1. Watts S. Humphrey, "An Introduction to the Team Software Process", Pearson Education,2000
2. James R. Persse, "Process Improvement essentials", O'Reilly,2006

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (*AUTONOMOUS*) : : ANANTAPUR

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52207: Software Reliability
ELECTIVE-IV

Objectives:

- To discuss the problems of reliability specification and measurement
- To introduce reliability metrics and to discuss their use in reliability specification
- To describe the statistical testing process
- To show how reliability predications may be made from statistical test results.

UNIT I:

Introduction: The Need for Reliable Software, Software Reliability Engineering Concepts, Basic definitions, Software practitioners biggest problem, software reliability engineering approach, software reliability engineering process, defining the product.

The Operational Profile: Reliability concepts, software reliability and hardware reliability, developing operational profiles, applying operational profiles, learning operations and run concepts.

UNIT II:

Software Reliability Concepts: Defining failure for the product, common measure for all associated systems, setting system failure intensity objectives, determining develop software failure intensity objectives, software reliability strategies, failures, faults and errors, availability, system and component reliabilities and failure intensities, predicting basic failure intensity.

UNIT III:

Software Reliability Modeling Survey: Introduction, Historical Perspective and Implementation, Exponential Failure Time Class of Models, Weibull and Gamma Failure Time Class of Models, Infinite Failure Category Models, Bayesian Models, Model Relationship, Software Reliability Prediction in Early Phases of the Life Cycle.

UNIT IV:

Software Metrics for Reliability Assessment: Introduction, Static Program Complexity, Dynamic Program Complexity, Software Complexity and Software Quality, Software Reliability Modeling.

Software Testing and Reliability: Introduction, Overview of Software Testing, Operational profiles, Time/Structure Based Software Reliability Estimation.

UNIT V:

Best Practice of SRE: Benefits and approaches of SRE, SRE during requirements phase, SRE during implementation phase, SRE during Maintenance phase.

Neural Networks for Software Reliability: Introduction, Neural Networks, Neural Networks for software reliability, software reliability growth modeling.

Text Books

1. Handbook of Software Reliability Engineering Edited by Michael R. Lyu, published by IEEE Computer Society Press and McGraw-Hill Book Company.
2. Software Reliability Engineering John D. Musa, second edition Tata McGraw-Hill.

Reference Books

1. Practical Reliability Engineering, Patric D. T. O connor 4th Edition, John Wesley & Sons, 2003.
2. Fault tolerance principles and Practice, Anderson and PA Lee, PHI, 1981.
3. Fault tolerant computing-Theory and Techniques, Pradhan D K (Ed.): Vol 1 and Vol 2, Prentice hall, 1986.
4. Reliability Engineering E. Balagurusamy, Tata McGrawHill, 1994.

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) : : ANANTAPURAMU

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	**T**	**P**	**C**
	4	**0**	**4**

15D52208: Big Data

ELECTIVE-IV

Objectives:

- To understand Big Data Analytics for different systems like Hadoop.
- To learn the design of Hadoop File System.
- To learn how to analyze Big Data using different tools.
- To understand the importance of Big Data in comparison with traditional databases.

UNIT- I

Introduction to Big Data. What is Big Data? Why Big Data is Important. Meet Hadoop Data, Data Storage and Analysis, Comparison with other systems, Grid Computing. A brief history of Hadoop. Apache hadoop and the Hadoop Ecosystem. Linux refresher, VMWare Installation of Hadoop.

UNIT-II

The design of HDFS. HDFS concepts. Command line interface to HDFS.Hadoop File systems. Interfaces. Java Interface to Hadoop. Anatomy of a file read. Anatomy of a file writes. Replica placement and Coherency Model. Parallel copying with distcp, keeping an HDFS cluster balanced.

UNIT-III

Introduction. Analyzing data with unix tools. Analyzing data with hadoop. Java MapReduce classes (new API). Data flow, combiner functions, Running a distributed MapReduce Job. Configuration API. Setting up the development environment. Managing configuration. Writing a unit test with MRUnit. Running a job in local job runner. Running on a cluster, Launching a job. The MapReduce WebUl.

UNIT-IV

Classic Mapreduce. Job submission. Job Initialization. Task Assignment. Task execution .Progress and status updates. Job Completion. Shuffle and sort on Map and reducer side.

Configuration tuning. Map Reduce Types. Input formats. Output cormats. Sorting. Map side and Reduce side joins.

UNIT-V

The Hive Shell. Hive services. Hive clients. The meta store. Comparison with traditional databases. Hive QI. Hbasics. Concepts. Implementation. Java and Map reduce clients. Loading data, web queries.

Text Books:

1. Tom White, Hadoop,"The Definitive Guide", 3rd Edition, O'Reilly Publications, 2012.

2. Dirk deRoos, Chris Eaton, George Lapis, Paul Zikopoulos, Tom Deutsch ,"Understanding Big Data Analytics for Enterprise Class Hadoop and Streaming Data", 1st Edition, TMH,2012.

References:

1. Big Data and Health Analytics Hardcover Katherine Marconi (Editor), Harold Lehmann (Editor)
2. Analytics in a Big Data World: The Essential Guide to Data Science and its Applications by bart baesens, Wiley publications.

(w.e.f 2015-16)

JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) : : ANANTAPURAMU

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	T	P	C
	4	0	4

15D52209: Software Reengineering

ELECTIVE-IV

Objectives:

- To explain why software re-engineering is a cost-effective option for system evolution
- To describe the activities involved in the software re-engineering process
- To distinguish between software and data re-engineering and to explain the problems of data re-engineering

UNIT I:

Software, Software Evolution and Maintenance: Software, Legacy software, Well designed software, Software evolution challenges, Lehman's laws, Software deterioration curve.

Software maintenance: Software change, Types of change encountered during the support phase, Maintenance costs, Why is software maintenance expensive?, Factors affecting maintenance, Maintenance process, Change and maintenance prediction.

Software Quality Factors, Quality and Maintainability Metrics: Internal and external attributes, McCall's quality factors, ISO 9126 quality factors, Need and importance of quality and maintainability metrics, Metric for software correctness (Defects/KLOC), Metric for software integrity, Software reliability (MTBF), Metrics for maintainability (Mean-time-to-change (MTTC), Spoilage metric, Software maturity index, McCabe and Halstead metrics).

UNIT II:

Design maintainability: Cohesion, Coupling, Understandability and Adaptability.

Legacy software structure, Software reengineering process model: Software change strategies include: Software maintenance, Architectural transformation, Software reengineering. Legacy software structure and distribution: Ideal structure, Real structure, Layered distribution model, Legacy software distribution, Architectural problems.

Business process reengineering: Business processes, A BPR Model, Software reengineering and its importance, Goals of reengineering, A software reengineering process model, Software reengineering activities.

UNIT III:

Design Extraction: Reverse Engineering: Goals of reverse engineering, Why design extraction is needed?, Reverse engineering process, Reverse engineering to understand processing, Code duplication detection, Reverse engineering to understand data, Reverse engineering user interfaces, Design extraction with UML, Heuristics to extract the design, Tools for reverse engineering.

Restructuring (In Traditional context): Code restructuring: Characteristics of unstructured code, Characteristics of structured code, Spaghetti logic, Structured control logic, Restructuring problems, Flow graph restructuring, Warnier's logical simplification techniques, Some basic code restructuring methods: Interchange, Transposition, Combination, Resolution, Substitution.

UNIT IV:

Data restructuring (Data reengineering): Data reengineering process, Data problems, Approaches: Data cleanup, Data extension, Data migration. Tools for restructuring.

Refactoring (Restructuring in object oriented context): What is refactoring?, Principles in refactoring: Why should you refactor?, When should you refactor?, Problems with refactoring, Refactoring and design, Refactoring and performance. Refactoring opportunities, Top ten of code bad smells, Different refactorings and their use, Refactoring tools.

UNIT V:

Forward Engineering: What is forward engineering ? Goals of forward engineering, Forward engineering for client/server applications, Forward engineering for object oriented architectures, Forward engineering user interfaces, Tools for forward engineering.

Reengineering Metrics, Repositories, and Economics:

Metrics in Reengineering: Why metrics in Reengineering?, Metrics as a reengineering tool, Which metrics to collect ?(Goal Question Metric (GQM) paradigm), Reengineering repositories: Why repositories?, Taxonomy (Functionality + Integration options), Issues.

Reengineering economics.

(w.e.f 2015-16)

TEXT BOOKS:

1. Software Reengineering, Ed. Robert S. Arnold, IEEE Computer Society, 1993.
2. Software Evolution, Tom Mens, Serge Demeyer, Springer publication company, 2008.

REFERENCES

1. Software Engineering, Ian Sommerville, Addison-Wesley, 6th Edition.
2. Software Engineering, A Practitioner's Approach, Roger S. Pressman, 6th Edition.
3. Refactoring: Improving the Design of Existing Code, Martin Fowler, K.Beck, J.Brant, W.Opdyke, D.Roberts, Addison- Wesley, NY, 1999.
4. Software Reengineering, Georg Abfalter, VDM Verlag, Germany, 2008.
5. Successful Software Reengineering, Salvatore Valenti, IRM Press, 2002.
6. Logical construction of programs, J.D.Warnier,Van Nostrand-Reinhold,1974.
7. Tutorial on Software Restructuring, Robert E.Arnold, IEEE Computer Society, 1986.

JNTUA COLLEGE OF ENGINEERING (*AUTONOMOUS*) : : ANANTAPUR

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)

15D54201: Research Methodology (Audit Course)

(Audit Course For M.Tech. –II Semester Program from 2015 admitted batches onwards)

UNIT I

Meaning of Research – Objectives of Research – Types of Research – Research Approaches – Guidelines for Selecting and Defining a Research Problem – research Design – Concepts related to Research Design – Basic Principles of Experimental Design.

UNIT II

Sampling Design – steps in Sampling Design –Characteristics of a Good Sample Design – Random Sampling Design.
Measurement and Scaling Techniques-Errors in Measurement – Tests of Sound Measurement – Scaling and Scale Construction Techniques – Time Series Analysis – Interpolation and Extrapolation.
Data Collection Methods – Primary Data – Secondary data – Questionnaire Survey and Interviews.

UNIT III

Correlation and Regression Analysis – Method of Least Squares – Regression vs Correlation – Correlation vs Determination – Types of Correlations and Their Applications

UNIT IV

Statistical Inference: Tests of Hypothesis – Parametric vs Non-parametric Tests – Hypothesis Testing Procedure – Sampling Theory – Sampling Distribution – Chi-square Test – Analysis of variance and Co-variance – Multi-variate Analysis.

UNIT V

Report Writing and Professional Ethics: Interpretation of Data – Report Writing – Layout of a Research Paper – Techniques of Interpretation- Making Scientific Presentations in Conferences and Seminars – Professional Ethics in Research.

Text books:

1. **Research Methodology:Methods and Techniques – C.R.Kothari, 2nd Edition,New Age International Publishers.**

2. **Research Methodology: A Step by Step Guide for Beginners- Ranjit Kumar, Sage Publications (Available as pdf on internet)**
3. **Research Methodology and Statistical Tools – P.Narayana Reddy and G.V.R.K.Acharyulu, 1st Edition,Excel Books,New Delhi.**

<u>REFERENCES:</u>

1. **Scientists must Write - Robert Barrass (Available as pdf on internet)**
2. **Crafting Your Research Future –Charles X. Ling and Quiang Yang (Available as pdf on internet)**

(w.e.f 2015-16)

JNTU COLLEGE OF ENGINEERING (*AUTONOMOUS*) : : ANANTAPUR

Department Of Computer Science & Engineering

M.Tech. I – II Sem.(SE)	**T**	**P**	**C**
	0	**4**	**2**

15D52210: Software Quality Assurance And Testing Lab

1. Write programs in C Language to demonstrate the working of the following constructs:
   i) do...while ii) while….do iii) if…else iv) switch v) for
2. A program written in C language for Matrix Multiplication fails. Introspect the causes for its failure and write down the possible reasons for its failure.
3. Consider ATM System and Study its system specifications and report the various bugs.
4. Write the test cases for Banking application.
5. Create test plan document for Library Management System.
6. Create test cases for Railway Reservation.
7. Create test plan document for Online Shopping.

Working with Tool's:

Understand the Automation Testing Approach, Benefits, Workflow, Commands and Perform Testing on one application using the following Tool's.

1. Win runner Tool for Testing.
2. Load runner Tool for Performance Testing.
3. Selenium Tool for Web Testing.
4. Bugzilla Tool for Bug Tracking.
5. Test Director Tool for Test Management.
6. Test Link Tool for Open Source Testing.

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR
JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) :: ANANTHAPURAMU
Ananthapuramu – 515 002, Andhra Pradesh, India

Course Structure of R21 Academic Regulations for M.Tech (Regular) Programs with effect from AY 2021-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SOFTWARE ENGINEERING

I SEMESTER

S.No.	Course Code	Subject Name	Cate Gory	Hours Per Week			Credits
				L	T	P	
1	21D52101	Advances in Software Engineering	PC	3	0	0	3
2	21D52102	Service Oriented Architecture	PC	3	0	0	3
3	**Professional Elective – I**						
	21D52103	Advanced Data Structures and Algorithms	PE	3	0	0	3
	21D52104	Software Requirements and Estimation					
	21D52105	Advanced Python Programming					
4	**Professional Elective – II**						
	21D52106	Artificial Intelligence for Software Engineering	PE	3	0	0	3
	21D52107	Software Architecture & Design Patterns					
	21D52108	Software Metrics and Reuse					
5	21D11109	Research Methodology and IPR	MC	2	0	0	2
6	21D11110	English for Research Paper Writing	AC	2	0	0	0
	21D11111	Value Education					
	21D11112	Pedagogy Studies					
7	21D52109	Software Engineering Lab	PC	0	0	4	2
8	21D52110	Service Oriented Architecture Lab	PC	0	0	4	2
Total				**16**	**00**	**08**	**18**

Course Structure of R21 Academic Regulations for M.Tech (Regular) Programs with effect from AY 2021-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SOFTWARE ENGINEERING

II SEMESTER

S.No.	Course Code	Subject Name	Cate Gory	Hours Per Week			Credits
				L	T	P	
1	21D52201	Machine Learning Techniques for Software Engineering	PC	3	0	0	3
2	21D52202	Software Quality Assurance and Testing	PC	3	0	0	3
3	**Professional Elective – III**						
	21D52203	Software Reliability	PE	3	0	0	3
	21D52204	Agile Methodologies					
	21D52205	Software Evolution and Maintenance					
4	**Professional Elective – IV**						
	21D52206	Data Science	PE	3	0	0	3
	21D52207	Secure Software Engineering					
	21D52208	Software Agents					
5	21D11209	Technical Seminar	PR	0	0	4	2
6	21D11210	Disaster Management	AC	2	0	0	0
	21D11211	Constitution of India					
	21D11212	Stress Management by Yoga					
7	21D55108	Machine Learning Lab	PC	0	0	4	2
8	21D52209	Software Testing Lab	PC	0	0	4	2
Total				**14**	**00**	**12**	**18**

SOFTWARE ENGINEERING

III SEMESTER

S.No.	Course Code	Subject Name	Cate Gory	Hours Per Week			Credits
				L	T	P	
1	**Professional Elective – V**						
	21D52301	Block Chain Technologies	PE	3	0	0	3
	21D52302	Software Project Planning &Management					
	21D52303	Programming Slicing Methods and Applications					
2	**Open Elective**						
	21D50301	Software Development and IT Services	OE	3	0	0	3
3	21D52304	Dissertation Phase – I	PR	0	0	20	10
4	21D00301	Co-Curricular Activities	PR				2
Total				**06**	**00**	**20**	**18**

IV SEMESTER

S.No.	Course Code	Subject Name	Cate Gory	Hours Per Week			Credits
				L	T	P	
1	21D52401	Dissertation Phase – II	PR	0	0	32	16
Total				**00**	**00**	**32**	**16**

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR
JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) :: ANANTHAPURAMU
Ananthapuramu – 515 002, Andhra Pradesh, India

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		**Advances in Software Engineering**	**L**	**T**	**P**	**C**
Semester	**I**		**3**	**0**	**0**	**3**

Course Objectives:

• A Broad And Critical Understanding Of All The Processes For Engineering High Quality Software And The Principles, Concepts And Techniques Associated With Software Development
• An Ability To Analyze And Evaluate Problems And Draw On The Theoretical And Technical Knowledge To Develop Solutions And Systems
• A Range Of Skills Focused On The Analysis Of Requirements, Design And Implementation Of Reliable And Maintainable Software, With Strong Emphasis On Engineering Principles Applied Over The Whole Development Lifecycle
• An Awareness Of Current Research In Software Development, The Analytical Skills And Research Techniques For Their Critical And Independent Evaluation And Their Application To New Problems.

Course Outcomes (CO): Student will be able to

- Analyze the importance of software quality assurance & testing in software development.
- Evaluate the concepts of software quality assurance techniques and find their relevance of use.
- Implement the concepts of software testing and appraise the most appropriate testing approaches for a given situation.
- Use the principles of testing and develop the necessary test cases in problem solution.

UNIT - I Lecture Hrs:12

Software and Software Engineering: The Nature of Software, The Unique Nature of WebApps, Software Engineering, Software Process, Software Engineering Practice, Software Myths.

Process Models: A Generic Process Model, Process Assessment and Improvement, Prescriptive Process Models, Specialized Process Models, The Unified Process, Personal and Team Process Models, Process Terminology, Product and Process.

UNIT - II Lecture Hrs:10

Understanding Requirements: Requirements Engineering, Establishing the Groundwork, Eliciting Requirements, Developing Use Cases, Building the Requirements Model, Negotiating Requirements, Validating Requirements.

Requirements Modelling: Requirements Analysis, Scenario-Based Modelling, UML Models That Supplement the Use Case, Data Modelling Concepts, Class-Based Modelling.

UNIT - III Lecture Hrs:10

Design Concepts: Design within the Context of Software Engineering, Design Process, Design Concepts, The Design Model.

Architectural Design: Software Architecture, Architectural Genres, Architectural Styles, Architectural Design, Assessing Alternative Architectural Designs, Architectural Mapping Using Data Flow.

Component-Level Design: What is a Component, Designing Class-Based Components, Conducting Component-Level Design, Component-Level Design for WebApps, Designing Traditional Components, Component-Based Development.

UNIT - IV Lecture Hrs:10

User Interface Design: The Golden Rules, User Interface Analysis and Design, Interface Analysis, Interface Design Steps, Design Evaluation.

Coding and Testing: Coding, Code Review, Software Documentation, Testing, Testing in the Large versus Testing in the Small, Unit Testing, Black-Box Testing, White-Box Testing, Debugging, Program Analysis Tools, Integration Testing, Testing Object-Oriented Programs, System Testing, Some General Issues Associated with Testing.

UNIT - V		Lecture Hrs:10
Verification and Validation: Planning Verification and Validation, Software Inspections, Automated Static Analysis, Verification and Formal Methods. **Software Maintenance:** Characteristics of Software Maintenance, Software Reverse Engineering, Software Maintenance Process Models, Estimation of Maintenance cost.		
Textbooks:		
1. Software Engineering A Practitioner's Approach, Roger S. Pressman, Eighth Edition Mc Graw Hill International Edition. 2. Fundamentals of Software Engineering, Rajib Mall, Third Edition, PHI.		
Reference Books:		
1. Software Engineering, Ian Sommerville, Eighth Edition, Pearson education. 2. Software Engineering : A Primer, Waman S Jawadekar, Tata McGraw-Hill, 2008 3. Software Engineering, A Precise Approach, Pankaj Jalote, Wiley India,2010.		

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		**Service Oriented Architecture**	**L**	**T**	**P**	**C**
Semester	**I**		**3**	**0**	**0**	**3**

Course Objectives:

1. Understand SOA and evolution of SOA.
2. Understand web services and primitive, contemporary SOA.
3. Understand various service layers.
4. Understand service-oriented analysis and design based on guidelines.

Course Outcomes (CO): Student will be able to

- Explain the meaning of the "Service Oriented" paradigm both from the business and technical point of view.
- Understand the applicability of SOA design patterns and the meaning of the major SOA implementation technologies.
- Compare SOA with other architectural paradigms. analyse requirements towards the creation of a service.
- Design a service starting from the analysis phase.

UNIT - I		Lecture Hrs:8

Introducing SOA: Fundamental SOA, Common Characteristics of Contemporary SOA, Common Tangible Benefits of SOA, and Common Pitfalls of Adopting SOA.
The Evolution of SOA: An SOA Timeline, The Continuing Evolution of SOA, The Roots of SOA.

UNIT - II		Lecture Hrs:10

Web Services and Primitive SOA: The Web Services Frame Work, Services, Service Descriptions, Messaging. **Web Services and Contemporary SOA (Part I-Activity management and Composition):** Message Exchange Patterns, Service Activity, Coordination, Atomic Transactions, Orchestration, Choreography.
Web Services and Contemporary SOA (Part-II-Advanced Messaging, Metadata and Security): Addressing, Reliable Messaging, Correlation, Policies, Metadata exchange, Security.

UNIT - III		Lecture Hrs:12

Principles of Service-Orientation: Service–Orientation and the Enterprise, Anatomy of SOA, Common Principles of Service–Orientation, Interrelation between Principles of ServiceOrientation, Service Orientation and Object Orientation, Native Web Services Support for Principles of Service-Orientation.
Service Layers: Service-Orientation and Contemporary SOA, Service Layer abstraction, Application Service Layer, Business Service Layer, Orchestration Service Layer, Agnostic Services, Service Layer Configuration Scenarios.

UNIT - IV		Lecture Hrs:12

SOA Delivery Strategies: SOA Delivery Lifecycle Phases, The Top-Down Strategy, The Bottom-up Strategy, The Agile Strategy.
Service Oriented Analysis (Part I-Introduction): Introduction to Service Oriented Analysis, Benefits of a Business Centric SOA, Deriving Business Services.
Service Oriented Analysis (Part-II-Service Modelling): Service Modelling, Service Modelling Guidelines, Classifying Service Model Logic, Contrasting Service Modelling Approaches.
Service Oriented Design (Part I-Introduction): Introduction to Service-Oriented Design, WSDL Related XML Schema Language Basics, WSDL Language Basics, Service Interface Design Tools.
Service Oriented Design (Part II-SOA Composition Guidelines): SOA Composing Steps, Considerations for Choosing Service Layers, Considerations for Positioning Core SOA Standards, Considerations for Choosing SOA Extensions.

UNIT - V		Lecture Hrs:10

Service Oriented Design (Part III- Service Design): Service Design Overview, Entity-Centric Business Service Design, Application Service Design, Task-Centric Business Service Design, Service Design Guidelines. Service **Oriented Design (Part IV-Business Process Design):** WS-BPEL Language Basics, WS- Coordination Overview, Service Oriented Business Process Design.
Textbooks:
1. Service-Oriented Architecture-Concepts, Technology, and Design, Thomas Erl, Pearson Education. Fundamentals of Software Engineering, Rajib Mall, Third Edition, PHI. 2. Understanding SOA with Web Services, Eric Newcomer, Greg Lomow, Pearson Education.
Reference Books:
1. The Definitive guide to SOA, Jeff Davies & others, Apress, Dreamtech. 2. Java SOA Cook book, E.Hewitt, SPD. 3. SOA in Practice, N.M.Josuttis, SPD. 4. Applied SOA, M.Rosen and others, Wiley India pvt. Ltd

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR
JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) :: ANANTHAPURAMU
Ananthapuramu – 515 002, Andhra Pradesh, India

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Program Elective Course - I

Course Code		Advanced Data Structures and Algorithms	**L**	**T**	**P**	**C**
Semester	**I**		**3**	**0**	**0**	**3**

Course Objectives:

- Single Linked, Double Linked Lists, Stacks, Queues, Searching and Sorting techniques, Trees, Binary trees, representation, traversal, Graphs- storage, traversal.
- Dictionaries, ADT for List, Stack, Queue, Hash table representation, Hash functions, Priority queues, Priority queues using heaps, Search trees.
- AVL trees, operations of AVL trees, Red- Black trees, Splay trees, comparison of search trees.
- To learn the advanced concepts of data structure and algorithms and its implementation .

Course Outcomes (CO): Student will be able to

- Ability to write and analyze algorithms for algorithm correctness and efficiency
- Master a variety of advanced abstract data type (ADT) and data structures and their Implementation
- Demonstrate various searching, sorting and hash techniques and be able to apply and solve problems of real life
- Design and implement variety of data structures including linked lists, binary trees, heaps, graphs and search trees
- Ability to compare various search trees and find solutions for IT related problems

UNIT – I		Lecture Hrs:8

Overview of Data Structures - Arrays, Stacks, Queues, linked lists , Linked stacks and Linked queues, Applications
Algorithm Analysis - Efficiency of algorithms, Asymptotic Notations, Time complexity of an algorithm using O notation, Polynomial Vs Exponential Algorithms, Average, Best, and Worst Case Complexities, Analyzing Recursive Programs.

UNIT - II		Lecture Hrs:8

Trees and Graphs – Basics of trees and binary trees, Representation of trees and Binary trees, Binary tree Traversals, Threaded binary trees, Graphs, representation and traversals.
Binary Search Trees, AVL Trees and B Trees - Binary Search Trees: Definition, Operations and applications. AVL Trees: Definition, Operations and applications. B Trees: Definition, Operations and applications.

UNIT - III		Lecture Hrs:8

Red – Black Trees, Splay Trees and Hash Tables - Red–Black Trees, Splay Trees and their applications, Dictionaries, Hash Tables, Hash Functions and various applications, File Organizations.

UNIT - IV		Lecture Hrs:8

Divide – and – Conquer & Greedy Method - General Method, Binary Search, Finding Maximum and Minimum, Quick Sort, Merge sort, Strassen's Matrix Multiplication, Greedy Method- General Method, Minimum Cost Spanning Trees, Single Source Shortest Path.
Back Tracking and Branch – and – Bound - General Method, 8 – Queen's Problem, Graph Coloring. Branch – and – Bound: The Method, LC Search, Control Abstraction, Bounding, 0 / 1 Knapsack Problem.

UNIT - V		Lecture Hrs:7

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Dynamic Programming - General Method, All Pairs Shortest Path, Single Source Shortest Path, 0 /1 Knapsack problem, Reliability Design, Traveling Sales Person's Problem.

Textbooks:

1. Fundamentals of Computer Algorithms by Ellis Horowitz, SartajSahni and SanguthevarRajasekaran, 2nd edition, University Press.

Reference Books:

1. Data Structures and Algorithms Using C++ by Ananda Rao Akepogu and RadhikaRaju Palagiri, Pearson Education, 2010.
2. Classic Data Structures by D. Samanta, 2005,PHI
3. Data Structures and Algorithms by G.A.V. Pai, 2009,TMH.
4. Design and Analysis of Computer Algorithms by Aho, Hopcraft, Ullman 1998,PEA.
5. Introduction to the Design and Analysis of Algorithms by Goodman, Hedetniemi,TMG
6. Design and Analysis of Algorithms by E. Horowitz, S. Sahani, 3rdEdition,Galgotia.
7. Data Structures and Algorithms in C++ by Drozdek 2nd Edition,Thomson.

Course Code		**Software Requirements and Estimation**	**L**	**T**	**P**	**C**
Semester	**I**		**3**	**0**	**0**	**3**

Course Objectives:

- To demonstrate knowledge of the distinction between critical and non- critical systems.
- To demonstrate the ability to manage a project including planning, scheduling and risk assessment/management.
- To author a software requirements document.
- To demonstrate an understanding of the proper contents of a software requirements document.

Course Outcomes (CO): Student will be able to

- To author a formal specification for a software system.
- To demonstrate an understanding of distributed system architectures and application architectures.
- To demonstrate an understanding of the differences between real-time and non-real time systems.
- To demonstrate proficiency in rapid software development techniques.
- To demonstrate proficiency in software development cost estimation
- To author a software testing plan.

UNIT – I		Lecture Hrs: 9 Hrs

Software Requirements: What And Why: Essential Software requirement, Good practices for requirements engineering, Improving requirements processes, Software requirements and risk management.

UNIT - II		Lecture Hrs: 10 Hrs

Software Requirements Engineering: Requirements elicitation, requirements analysis documentation, review, elicitation techniques, analysis models, Software quality attributes, risk reduction through prototyping, setting requirements priorities, verifying requirements quality.

UNIT - III		Lecture Hrs: 09 Hrs

Software Requirements Modeling: Use Case Modeling, Analysis Models, Dataflow diagram, state transition diagram, class diagrams, Object analysis, Problem Frames.

Software Requirements Management: Requirements management Principles and practices, Requirements attributes, Change Management Process, Requirements Traceability Matrix, Links in requirements chain.

UNIT - IV		Lecture Hrs: 09 Hrs

Software Estimation: Components of Software Estimations, Estimation methods, Problems associated with estimation, Key project factors that influence estimation.

Size Estimation: Two views of sizing, Function Point Analysis, Mark II FPA, Full Function Points, LOC Estimation, Conversion between size measures.

Effort, Schedule and Cost Estimation: What is Productivity? Estimation Factors, Approaches to Effort and Schedule Estimation, COCOMO II, Putnam Estimation Model, Algorithmic models, Cost Estimation.

UNIT - V		Lecture Hrs: 09 Hrs

Requirements Management Tools: commercial requirements management requirements management automation. Benefits of using a requirements management tool, tool, Rational Requisite pro, Caliber – RM, implementing

Software Estimation Tools: Desirable features in software estimation tools, IFPUG, USC's COCOMO II, SLIM (Software Life Cycle Management) Tools.

Textbooks:

1. Software Requirements by Karl E. Weigers,Microsoft Press.
2. Software Requirements and Estimation by Rajesh Naik and Swapna Kishore, Tata Mc Graw Hill.

Reference Books:

1. Managing Software Requirements, Dean Leffingwell& Don Widrig, Pearson Education,2003.
2. Mastering the requirements process, second edition, Suzanne Robertson & James Robertson, Pearson Education, 2006.
3. Estimating Software Costs, Second edition, Capers Jones, Tata McGraw-Hill, 2007.
4. Practical Software Estimation, M.A. Parthasarathy, Pearson Education, 2007.
5. Measuring the software process, William A. Florac& Anita D. Carleton, Pearson Education,1999.

Online Learning Resources:

1.https://books.google.co.in/books/about/Software_Requirements_And_Estimation.html?id=TaRGLO57vnUC

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR
JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) :: ANANTHAPURAMU
Ananthapuramu – 515 002, Andhra Pradesh, India

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		Advanced Python Programming	L	T	P	C
Semester	I		3	0	0	3

Course Objectives:

The main objective of this course is to help students learn, understand, and practice dataanalytics using python, which include the study of modern computingbig data technologies and scaling up machine learning techniques focusing on industryapplications. Mainly the course objectives are conceptualization and summarization of data

Course Outcomes (CO): Student will be able to

- Write relatively advanced, well structured, computer programs in Python
- Gain familiarity with principles and techniques for optimizing the performance of numeric applications
- Understand parallel computing and how parallel applications can be written in Python
- Experiment with developing GPU accelerated Python applications
- Learn the fundamentals of the most widely used Python packages; including NumPy, Pandas and Matplotlib

UNIT – I		Lecture Hrs: 9 Hrs

Introduction- Creating the Data Science Pipeline, Understanding Python's Role in Data Science, Learning to Use Python Fast, Setting Up Python for Data Science, Reviewing Basic Python

UNIT - II		Lecture Hrs: 10 Hrs

Uploading, Streaming, and Sampling Data, Accessing Data in Structured Flat File Form, Sending Data in Unstructured File Form, Managing Data from Relational Databases, Interacting with Data from NoSQL Databases, Accessing Data from the Web,NumPy and pandas, Validating Your Data, Manipulating Categorical Variables, Dealing with Dates in Your Data, Slicing and Dicing: Filtering and Selecting Data, Aggregating Data at Any Level

UNIT - III		Lecture Hrs: 09 Hrs

Working with HTML Pages, Working with Raw Text, Using the Bag of Words Model and Beyond, Working with Graph Data, Contextualizing Problems and Data, Considering the Art of Feature Creation, Performing Operations on Arrays

UNIT - IV		Lecture Hrs: 09 Hrs

Starting with a Graph, Setting the Axis, Ticks, Grids, Defining the Line Appearance, Using Labels, Annotations, and Legends, Choosing the Right Graph, Creating Advanced Scatterplots, Plotting Time Series, Plotting Geographical Data, Visualizing Graphs

UNIT - V		Lecture Hrs: 09 Hrs

Playing with Scikit-learn, Performing the Hashing Trick, Considering Timing and Performance, Running in Parallel, Counting for Categorical Data, Understanding Correlation, Modifying Data Distributions, Reducing Dimensionality, Clustering, Detecting Outliers in Data

Textbooks:

1. Python for Data Science for Dummies, 2ed, Luca Massaron John Paul Mueller, by ISBN: 978-1- 118-84418-2

Reference Books:

1. Introduction to Parallel Computing, AnanthGrama, Anshul Gupta, George Karypis, Vipin Kumar, Pearson; 2 edition (January 26, 2003), ISBN 978-0201648652
2. Big Data: Principles and best practices of scalable realtime data systems, 1st Edition, Nathan Marz, James Warren, ISBN 978-1617290343

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Program Elective Course – II

Course Code		Artificial intelligence for Software Engineering	L	T	P	C
Semester	I		3	0	0	3

Course Objectives:

1. Acquire knowledge on intelligent systems and agents.
2. Formalization of knowledge, reasoning with and without uncertainty machine learning and applications at a basic level.

Course Outcomes (CO): Student will be able to

- Familiarity about how AI is being used for addressing different problems in SE.
- In-depth understanding of use of AI models in one particular SE issue
- Understanding of how to apply AI models for a different domain (SE in this case)

UNIT - I		Lecture Hrs:8

Introduction to Computer Software: Computers and software systems, An introduction to software engineering, Bridges and buildings versus software systems, The software crisis, A demand for more software power, Responsiveness to human users, Software systems in new types of domains, Responsiveness to dynamic usage environments, Software systems with self-maintenance capabilities, A need for Al systems.

UNIT - II		Lecture Hrs:8

AI Problems and Conventional SE Problems: What is an AI problem?, Ill-defined specifications, Correct versus 'good enough' solutions, It's the HOW not the WHAT, The problem of dynamics, The quality of modular approximations, Context-free problems.
Software Engineering Methodology: Specify and verify—the SAV methodology, The myth of complete specification, What is verifiable?, Specify and test—the SAT methodology, Testing for reliability, The strengths, The weaknesses, What are the requirements for testing?, What's in a specification?, Prototyping as a link

UNIT - III		Lecture Hrs:8

An Incremental and Exploratory Methodology: Classical methodology and AI problems,The RUDE cycle , How do we start?, Malleable software, AI muscles on a conventional skeleton, How do we proceed? , How do we finish? , The question of hacking , Conventional paradigms.

UNIT - IV		Lecture Hrs:8

New Paradigms for System Engineering: Automatic programming ,Transformational implementation, The "new paradigm" of Blazer, Cheatham and Green, Operational requirements of Kowalski, The POLITE methodology ,Towards a Discipline of Exploratory Programming ,Reverse engineering , Reusable software ,Design knowledge, Stepwise abstraction, The problem of decompiling ,Controlled modification, Structured growth ,Machine Learning: Much Promise, Many Problems ,Self-adaptive software ,The promise of increased software power ,The threat of increased software problems , Page v The state of the art in machine learning, Practical machine learning examples,Multiversion inductive programming.

UNIT - V		Lecture Hrs:7

Expert Systems: The Success Story, Expert systems as Al software, Engineering expert systems, The lessons of expert systems for engineering Al software: Failure Modes and Effects Analysis.
AI into Practical Software: Support environments, Reduction of effective complexity, Moderately stupid assistance, An engineering toolbox, Self-reflectivesoftware, Over engineering software.

Textbooks:

1. "Artificial intelligence and software engineering, Understanding the Promise of the Future", Derek Partridge, Glenlake Publishing Company, Ltd.

Reference Books:

1. Adams, J.M. and Smartt, M. (1985) Software reliability through redundancy, Proc. 18th Hawaii International Conference on Systems Sciences.
2. Andrews, D. (1990) The Vienna Development Method, in D. Ince& D. Andrews (Eds.) The Software Life Cycle, London: Butterworths, pp. 221-259.

Bahrami, A. (1988) Designing Artificial Intelligence Based Software, Wilmslow, UK: Sigma Press.

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Program Elective Course - II

Course Code		**Software Architecture and Design Patterns**	**L**	**T**	**P**	**C**
Semester	**I**		**3**	**0**	**0**	**3**

Course Objectives:

To understand interrelationships, principles and guidelines governing architecture and evolution over time.
• To understand various architectural styles of software systems.
• To understand design patterns and their underlying object oriented concepts.
• To understand implementation of design patterns and providing solutions to real world software design problems.
• To understand patterns with each other and understanding the consequences of combining patterns on the overall quality of a system.

Course Outcomes (CO): Student will be able to

- Knows the underlying object oriented principles of design patterns.
- Understands the context in which the pattern can be applied.
- Understands how the application of a pattern affects the system quality and its tradeoffs.
- Design and motivate software architecture for large scale software systems
- 5.Recognize major software architectural styles, design patterns, and frameworks

UNIT – I		Lecture Hrs:8

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		Lecture Hrs:8

Analyzing 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		Lecture Hrs:8

Patterns Pattern Description, Organizing catalogs, role in solving design problems, Selection and usage.
Creational and Structural Patterns:
Abstract factory, builder, factory method, prototype, singleton, adapter, bridge, composite, façade, flyweight.

UNIT - IV		Lecture Hrs:8

Behavioural Patterns:
Chain of responsibility, command, Interpreter, iterator, mediator, memento, observer, state, strategy, template method, visitor.

UNIT - V		Lecture Hrs:7

Case Studies:
A-7E – A case study in utilizing architectural structures, The World Wide Web - a case study in interoperability, Air Traffic Control – a case study in designing for high availability, Celsius Tech – a case study in product line development.

A Case Study (Designing a Document Editor): Design Problems, Document Structure, Formatting, Embellishing the User Interface, Supporting Multiple Look-and-Feel Standards, Supporting Multiple Window Systems, User Operations, Spelling Checking and Hyphenation.
Textbooks:
1.Software Architecture in Practice, second edition, Len Bass, Paul Clements & Rick Kazman, Pearson Education, 2003. 2. Design Patterns, Erich Gamma, Pearson Education, 1995.
Reference Books:
1. Beyond Software architecture, Luke Hohmann, Addison wesley, 2003. 2. . Software architecture, David M. Dikel, David Kane and James R. Wilson, Prentice Hall PTR, 2001 3. Software Design, David Budgen, second edition, Pearson education, 2003 4. Head First Design patterns, Eric Freeman & Elisabeth Freeman, O'REILLY, 2007. 5. Design Patterns in Java, Steven John Metsker& William C. Wake, Pearson education, 2006 6. J2EE Patterns, Deepak Alur, John Crupi& Dan Malks, Pearson education, 2003. 7. Design Patterns in C#, Steven John metsker, Pearson education, 2004. 8. Pattern Oriented Software Architecture, F.Buschmann& others, John Wiley & Sons..

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Program Elective Course – II

Course Code		**Software Metrics and Reuse**	**L**	**T**	**P**	**C**
Semester	**I**		**3**	**0**	**0**	**3**

Course Objectives:

• To understand why measurement is important
• To know how to extract, when and where to apply relevant metrics
• To understand the importance of measurement in software engineering
• To describe and compare the different metrics that can be used for measuring software
• To understand the important factors that affect the measurement of software

Course Outcomes (CO): Student will be able to

- Acquired basic knowledge of Software quality models
- Exemplify Quality measurement and metrics, Quality plan and implementation
- Articulate Quality control and reliability of quality process and Quality management system models
- Articulate Complexity metrics and Customer Satisfaction and International quality standards – ISO, CMM
- Control and Manage the project and processes, apply configuration management on the basis of collected metrics.

UNIT – I		Lecture Hrs:10

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.

UNIT – II		Lecture Hrs:10

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.

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.

UNIT – III		Lecture Hrs:10

Measuring external product attributes: Modelling software quality, measuring aspects of software quality.

Metrics for object-oriented systems: The intent of object-oriented metrics, distinguishing characteristics of object-oriented metrics, various object-oriented metric suites – LK suite, CK suite and MOOD metrics.

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

UNIT – IV		Lecture Hrs:10

Introduction: Software Reuse and 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, Aspects of Software Reuse, Organizational Aspects, Technical Aspects and Economic Aspects.

Programming Paradigm and Reusability: Usability Attributes, Representation and Modeling Paradigms, Abstraction and Composition in development paradigm.

UNIT – V		Lecture Hrs:10

Object-Oriented Domain Engineering: Abstraction and Parameterization Techniques, Composition Techniques in Object Orientation.

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.

Textbooks:

1. Norman E. Fenton and Shari Lawrence Pfleeger; Software Metrics – A Rigorous and Practical Approach,Thomson Asia Pte., Ltd, Singapore.
2. Stephen H. Kan; Metrics and Models in Software Quality Engineering, Addison Wesley, New York.
3. Reuse Based Software Engineering Techniques, Organization and Measurement by Hafedh Mili, Ali Mili, SherifYacouband Edward Addy, John Wiley & Sons Inc
4. The Three Rs of Software Automation: Re-engineering, Repository, Reusability by Carma McClure, Prentice Hall NewJersey engineering, Repository, Reusability by Carma McClure, Prentice Hall New Jersey

Reference Books:

1. K. H. Möller and D. J. Paulish; Software Metrics - A Practitioner's Guide to Improved Product Development, Chapman and Hall, London.
2. Mark Lorenz and Jeff Kidd; Object-Oriented Software Metrics, Prentice Hall, New York.
3. McClure, Carma L. Software reuse techniques : adding reuse to the system development process / : Prentice Hall
4. Poulin, Jeffrey S. Measuring software reuse: principles, practices, and economic models / Jeffrey S. Poulin. Reading, Mass. : Addison-Wesley

Online Learning Resources:

1. https://www.imagix.com/links/software-metrics.html

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		**Software Engineering Lab**	**L**	**T**	**P**	**C**
Semester	**I**		**0**	**0**	**4**	**2**

Course Objectives:

- To have hands on experience in developing a software project by using various software engineering principles and methods in each of the phases of software development.

Course Outcomes (CO): Student will be able to

- Ability to translate end-user requirements into system and software requirements
- Ability to generate a high-level design of the system from the software requirements
- Will have experience and/or awareness of testing problems and will be able to develop a simple testing report

List of Experiments:

Do the following 8 exercises for any two projects given in the list of sample projects or any other projects:

1. Identifying Requirements from ProblemStatements

   Requirements | Characteristics of Requirements | Categorization of Requirements | Functional Requirements | Identifying Functional Requirements | Preparing Software Requirements Specifications
2. Estimation of ProjectMetrics

   Project Estimation Techniques | COCOMO | Basic COCOMO Model | Intermediate COCOMO Model | Complete COCOMO Model | Advantages of COCOMO | Drawbacks of COCOMO | Halstead's ComplexityMetrics.
3. Modeling UML Use Case Diagrams and Capturing Use CaseScenarios

   Use case diagrams | Actor | Use Case | Subject | Graphical Representation | Association between Actors and Use Cases | Use Case Relationships | Include Relationship | Extend Relationship | Generalization Relationship | Identifying Actors | Identifying Use cases | Guidelines for drawing Use Case diagrams

4. E-R Modeling from the ProblemStatements

   Entity Relationship Model | Entity Set and Relationship Set | Attributes of Entity | Keys | Weak Entity | Entity Generalization and Specialization | Mapping Cardinalities | ER Diagram | Graphical Notations for ER Diagram | Importance of ER modeling
5. Identifying Domain Classes from the ProblemStatements

   Domain Class | Traditional Techniques for Identification of Classes | Grammatical Approach Using Nouns | Advantages | Disadvantages | Using Generalization | Using Subclasses | Steps to Identify Domain Classes from Problem Statement | Advanced Concepts.

6. Statechart and ActivityModeling

   Statechart Diagrams | Building Blocks of a Statechart Diagram | State | Transition | Action | Guidelines for drawing Statechart Diagrams | Activity Diagrams | Components of an Activity Diagram | Activity | Flow | Decision | Merge | Fork | Join | Note | Partition | A Simple Example | Guidelines for drawing an Activity Diagram

7. Modeling UML Class Diagrams and SequenceDiagrams

   Structural and Behavioral Aspects | Class diagram | Class | Relationships | Sequence diagram | Elements in sequence diagram | Object | Life-line bar | Messages

8. Modeling Data Flow Diagrams

   Data Flow Diagram | Graphical notations for Data Flow Diagram | Symbols used in DFD | Context diagram and levelingDFD.

9. Study and usage of any Design phase CASE tool. Performing the Design by using any Design phase CASE tools.

10. Estimation of Test Coverage Metrics and StructuralComplexity

    Control Flow Graph | Terminologies | McCabe's Cyclomatic Complexity | Computing Cyclomatic Complexity | Optimum Value of Cyclomatic Complexity | Merits | Demerits

11. Designing TestSuites

    Software Testing | Standards for Software Test Documentation | Testing Frameworks | Need for Software Testing | Test Cases and Test Suite | Types of Software Testing | Unit Testing | Integration Testing | System Testing | Example | Some Remarks.

Sample Projects:
1. Passport automation System
2. Library management System
3. Online Exam Registration
4. Stock Maintenance System
5. Online Course Reservation System
6. E-ticketing
7. Software Personnel Management System
8. Credit Card Processing
9. E-book management System.
10. Automated banking system
11. Airline reservation system

12. Employee management application 13.. Hospital management Application
Textbooks: 1. Software Engineering, A practitioner’s Approach- Roger S. Pressman, 8th edition, Mc Graw Hill International Edition. 2. Software Engineering- Sommerville, 7th edition, Pearson Education. 3. The unified modeling language user guide Grady Booch, James Rambaugh, Ivar Jacobson, Pearson Education.

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		Service Oriented Architecture Lab	L	T	P	C
Semester	I		0	0	4	2

Course Objectives:

- To gain understanding of the basic principles of service orientation
- To learn technology underlying the service design
- To learn service oriented analysis techniques
- To learn advanced concepts such as service composition, orchestration and
- Choreography To know about various WS- * specification standards

Course Outcomes (CO): Student will be able to

- Introduction To distributed Computing and SOA
- Web Services Fundamental and Standard
- Principles of Service-Oriented Architecture
- SOA and WS-* Extension
- Principle of Service Oriented Computing
- SOA Platforms

List of Experiments:

1. Develop at least 5 components such as Order Processing, Payment Processing, etc., using .NET component technology.
2. Develop at least 5 components such as Order Processing, Payment Processing, etc., using EJB Component Technology.
3. Invoke .NET components as web services.
4. Invoke EJB components as web services.
5. Develop a Service Orchestration Engine (workflow) using WS-BPEL and Implement Service Composition. For Example, a business process for planning business travels will invoke several services. This process will invoke several airline companies (such as American Airlines, Delta Airlines etc.) to check the airfare price and buy at the lowest price.
6. Develop a J2EE client to access a .NET web service.
7. Develop a .NET client to access a J2EE web service.

Write problem definition, overall description, specific requirements, front – end description, back – end description and draw the data flow diagrams & UML diagram for following CASE Studies.

1. Library Management System
2. Automated banking system
3. Airline reservation system
4. Employee management application
5. Hospital management Application

References:

Online learning resources/Virtual labs

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		Machine Learning Applications for Software Engineering	L	T	P	C
Semester	II		3	0	0	3

Course Objectives:

- To learn the purpose of ML in Software Engineering.
- To understand the role of ML in prediction and Estimation.
- To recognize the ML applications in Property and Model Discovery.
- To intricate the Usage of ML in Requirements Acquisition and development of Knowledge.

Course Outcomes (CO): Student will be able to

CO1: Understand the purpose of ML in Software Engineering.
CO2: Identify the role of ML in prediction and Estimation.
CO3: Recognize the ML applications in Property and Model Discovery.
CO4: Usage of ML in Requirements Acquisition and development of Knowledge.

UNIT - I		Lecture Hrs:10

INTRODUCTION TO ML AND SOFTWARE ENGINEERING: Overview Of ML, Learning Approaches, SE Tasks For ML Applications, State Of The Practice In ML And SE, Property And Model Discovery, Transformation, Generation And Synthesis, Reuse Library Construction And Maintenance,Requirementacquisition,Capture Development Knowledge

Unit - Ii		Lecture Hrs:10

Machine Learning Applications In Prediction And Estimation:Bayesian Analysis Of Empirical Software Engineering Cost Models, Machine Learning Approaches To Estimating Software Development Effort, Estimating Software Project Effort Using Analogies, A Critique Of Software Defect Prediction Models, Using Regression Trees To Classify Fault-Prone Software Modules, Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation, Optimal Software Release Scheduling Based On Artificial Neural Networks.

Unit - Iii		Lecture Hrs:10

Ml Applications In Property And Model Discovery: Identifying Objects In Procedural Programs Using Clustering Neural Networks, Bayesian-Learning Based Guidelines To Determine Equivalent Mutants. Ml Applications In Reuse: On The Reuse Of Software: A Case-Based Approach Employing A Repository.

Unit – Iv		Lecture Hrs:10

ML Applications In Requirements Acquisition: Inductive Specification Recovery: Understanding Software By Learning From Example Behaviors, Explanation-Based Scenario Generation For Reactive System Models.

Unit – V		Lecture Hrs:9

ML Applications In Management Of Development Knowledge: Case-Based Knowledge Management Tools For Software Development.

Textbooks:

1.Machine Learning Applications In Software Engineering- Edited By: Du Zhang (California State University, USA) And Jeffrey J P Tsai (University Of Illinois, Chicago, USA)Feb 2005.
2. Applied Software Development With Python & Machine Learning By Wearable & Wireless Systems For Movement Disorder Treatment Via Deep Brain Stimulation By By (Author): Robert Lemoyne (Northern Arizona University, USA) And Timothy Mastroianni.

Reference Books:

1. HandBook on Machine Learning- Volume 1: Foundation of Artificial Intelligence by TshilidziMarwala.

Online Learning Resources:

https://www.worldscientific.com/worldscibooks/10.1142/5700#t=toc

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		Program Elective Course – III **Software Evolution and Maintenance**	**L**	**T**	**P**	**C**
Semester	**II**		**3**	**0**	**0**	**3**

Course Objectives:

- To learn the purpose of ML in Software Engineering.
- To understand the role of ML in prediction and Estimation.
- To recognize the ML applications in Property and Model Discovery.
- To intricate the Usage of ML in Requirements Acquisition and development of Knowledge.

Course Outcomes (CO): Student will be able to

CO1: Understand the purpose of ML in Software Engineering.
CO2: Identify the role of ML in prediction and Estimation.
CO3: Recognize the ML applications in Property and Model Discovery.
CO4: Usage of ML in Requirements Acquisition and development of Knowledge.

UNIT - I		Lecture Hrs:10

Basic Concepts and Preliminaries: Evolution Versus Maintenance, Software Evolution, Software Maintenance, Software Evolution Models and Processes, Reengineering, Legacy Systems, Impact Analysis, Refactoring, Program Comprehension, Software Reuse.

Taxonomy of Software Maintenance and Evolution: General Idea, Categories of Maintenance Concepts, Evolution of Software Systems, Maintenance of Cots-Based Systems.

Unit - II		Lecture Hrs:10

Evolution and Maintenance Models: General Idea, Reuse-Oriented Model, The Staged Model for Closed Source Software, The Staged Model for Free, Libre, Open Source Software, Change Mini-Cycle Model, IEEE/EIA Maintenance Process, ISO/IEC 14764 Maintenance Process, Software Configuration Management, Brief History, SCM Spectrum of Functionality, 1SCM Process, CR Workflow,

Unit - III		Lecture Hrs:10

Reengineering General Idea, Reengineering Concepts, A General Model for Software Reengineering, Types of Changes, Software Reengineering Strategies, Reengineering Variations, Reengineering Process, Reengineering Approaches, Source Code Reengineering Reference Model, Phase Reengineering Model, Code Reverse Engineering, Techniques Used for Reverse Engineering, Decompilation Versus Reverse Engineering, Data Reverse Engineering, Reverse Engineering Tools.

Legacy Information Systems: General Idea, Wrapping, Migration, Migration Planning, Migration Methods.

Unit –IV		Lecture Hrs:10

Impact Analysis : General Idea, Impact Analysis Process, Identifying the SIS, Analysis of Traceability Graph, Identifying the Candidate Impact Set, Dependency-Based Impact Analysis, Call Graph, Program Dependency Graph, Ripple Effect, Computing Ripple Effect, Change Propagation Model, Recall and Precision of Change Propagation Heuristics, Heuristics for Change Propagation, Empirical Studies.

Refactoring: General Idea, Activities in a Refactoring Process, Formalisms for Refactoring, More Examples of Refactoring, and Initial Work on Software Restructuring.

Unit – V		Lecture Hrs:9
Program Comprehension: General Idea, Basic Terms, Cognition Models for Program Understanding, Protocol Analysis, Visualization for Comprehension. **Reuse and Domain Engineering:** General Idea, Domain Engineering, Reuse Capability, Maturity Models, Economic Models of Software Reuse.		
Textbooks:		
1. Software Evolution and Maintenance: A Practitioner's Approach, PriyadarshiTripathy, KshirasagarNaik		

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Semester	II	Software Quality Assurance & Testing	3	0	0	3

Course Objectives:

- Understand software testing and quality assurance as a fundamental component of software life cycle
- Define the scope of software testing & quality assurance projects
- Efficiently perform testing & quality assurance activities using modern software tools
- Estimate cost of a testing & quality assurance project and manage budgets
- Prepare test plans and schedules for a testing & quality assurance project
- Develop testing & quality assurance project staffing requirements
- Effectively manage a testing & quality assurance project

Course Outcomes (CO): Student will be able to

CO1	Understand fundamental concepts of software automation
CO2	Apply Selenium automation tool for testing web based application
CO3	Demonstrate the quality management, assurance, and quality standard to software system
CO4	Demonstrate Software Quality Tools and analyze their effectiveness.
CO5	Describe fundamental concepts of software quality assurance

UNIT - I — Lecture Hrs:10

Introduction to software quality: Challenges, Objectives, Quality Factors, Components of SQA, Contract review, Development and quality Plans, SQA Components in Project Life Cycle, SQA Defect Removal Policies, Reviews.

UNIT - II — Lecture Hrs:10

Software Testing Strategy and Environment: Minimizing Risks, Writing a Policy for Software Testing, Economics of Testing, Testing-an organizational issue, Management Support for Software Testing, Building a Structured Approach to Software Testing, Developing a Test Strategy. Building Software Testing Process: Software Testing Guidelines, Workbench Concept, Customizing the Software Testing Process, Process Preparation Checklist.

UNIT - III — Lecture Hrs:10

Software Testing Techniques: Dynamic Testing – Black Box Testing Techniques, White Box Testing Techniques, Static Testing, Validation Activities, Regression Testing. Software Testing Tools: Selecting and Installing Software Testing tools Automation and Testing Tools: Load Runner, Win runner and Rational Testing Tools, Silk test, Java Testing Tools, JMetra, JUNIT and Cactus.

UNIT - IV — Lecture Hrs:10

Seven Step Testing Process–I: Overview of the Software Testing Process, Organizing of Testing, Developing the Test Plan, Verification Testing, Validation Testing.

UNIT - V — Lecture Hrs:10

Seven Step Testing Process-II: Analyzing and Reporting Test results, Acceptance and Operational Testing, Post-Implementation Analysis Specialized Testing Responsibilities: Software Development Methodologies, Testing Client/Server Systems.

Textbooks:

1.Effective Methods for Software Testing, Third edition, William E. Perry, Wiley India, 2009.

2.Software Testing – Principles and Practices, Naresh Chauhan, Oxford University Press, 2010.

3. Software Quality Assurance – From Theory to Implementation, Daniel Galin, Pearson Education, 2009.

Reference Books:

1. Testing Computer Software, CemKaner, Jack Falk, Hung Quoc Nguyen, Wiley India,

rp2012. 2. Software Testing – Principles, Techniques and Tools, M.G.Limaye, Tata McGraw-Hill, 2009. 3. Software Testing - A Craftsman's approach, Paul C. Jorgensen, Third edition, Auerbach Publications, 2010. 4. Software Quality Assurance, MilindLimaye, Tata McGraw-Hill, 2011.
Online Learning Resources:

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR
JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS) :: ANANTHAPURAMU
Ananthapuramu – 515 002, Andhra Pradesh, India

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Semester	II	SOFTWARE RELIABILITY	3	0	0	3

Course Objectives:

- Apply engineering knowledge and specialist techniques to prevent or to reduce the likelihood or frequency of failures.
- To identify and correct the causes of failures that do occur despite the efforts to prevent them.
- To determine ways of coping with failures that do occur, if their causes have not been corrected
- To apply methods for estimating the likely reliability of new designs, and for analyzing reliability data.

Course Outcomes (CO): Student will be able to

CO1	Knows the process and basic activities of software reliability engineering, causes of failure appearance, software reliability metrics and models, methods for ensuring, evaluation and enhancing of software reliability.
CO2	Is able to detect, to analyze and to evaluate software faults, failures and errors using appropriate CASE tools
CO3	Is able to implement different software reliability models and to evaluate the reliability of developed tool using different methods and tools
CO4	Is able to select an appropriate reliability model, to collect necessary data during testing, to perform an evaluation of software reliability and in case of necessity to enhance reliability.

UNIT - I | Lecture Hrs:10

Introduction and Operational Profile: The Need for Reliable Software, Software Reliability Engineering Concepts, Basic definitions, Software practitioners biggest problem, software reliability engineering approach, software reliability engineering process, defining the product, Reliability concepts, software reliability and hardware reliability, developing operational profiles, applying operational profiles, learning operations and run concepts.

UNIT - II | Lecture Hrs:10

Software Reliability Concepts Defining failure for the product, common measure for all associated systems, setting system failure intensity objectives, determining develop software failure intensity objectives, software reliability strategies, failures, faults and errors, availability, system and component reliabilities and failure intensities, predicting basic failure intensity.

UNIT - III | Lecture Hrs:10

Software Reliability Modeling Survey Introduction, Historical Perspective and Implementation, Exponential Failure Time Class of Models, Weibull and Gamma Failure Time Class of Models, Infinite Failure Category Models, Bayesian Models, Model Relationship, Software Reliability Prediction in Early Phases of the Life Cycle, software reliability growth modeling.

UNIT - IV | Lecture Hrs:10

Software Metrics for Reliability Assessment Introduction, Static Program Complexity, Dynamic Program Complexity, Software Complexity and Software Quality, Software Reliability Modeling.

UNIT - V | Lecture Hrs:10

Software Testing and Reliability Introduction, Overview of Software Testing, Operational profiles, Time/Structure Based Software Reliability Estimation, Benefits and approaches of SRE, SRE during requirements phase, SRE during implementation phase, SRE during Maintenance phase.

Textbooks:

1. Handbook of Software Reliability Engineering Edited by Michael R. Lyu, published by IEEE Computer Society Press and McGraw-Hill Book Company 2. Software Reliability Engineering John D. Musa, second edition Tata McGraw-Hill.
Reference Books:
1. Practical Reliability Engineering, Patric D. T. O connor 4th Edition, John Wesley & Sons, 2003. 2. Fault tolerance principles and Practice, Anderson and PA Lee, PHI, 1981. 3. Fault tolerant computing-Theory and Techniques, Pradhan D K (Ed.): Vol 1 and Vol 2, Prentice hall, 1986. 4. Reliability Engineering E. Balagurusamy, Tata McGrawHill, 1994.
Online Learning Resources:

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Semester	**II**	Program Elective Course – III **Agile Methodologies**	**3**	**0**	**0**	**3**

Course Objectives:

- To provide students with a theoretical as well as practical understanding of agile software development practices and how small teams can apply them to create high-quality software.
- To provide a good understanding of software design and a set of software technologies and APIs.
- To do a detailed examination and demonstration of agile development and testing techniques.
- To understand the benefits and pitfalls of working in an agile team.
- To understand agile development and testing.

Course Outcomes (CO): Student will be able to

CO1	Understand The XP Lifecycle, XP Concepts, Adopting XP.
CO2	Work on Pair Programming, Root-Cause Analysis, Retrospectives, Planning, Incremental Requirements, Customer Tests.
CO3	Implement Concepts to Eliminate Waste.
CO4	Appreciate and focus on the most important aspects of project development and sprints.

UNIT - I — Lecture Hrs:10

Why Agile?
Understanding Success, Beyond Deadlines, The Importance of Organizational Success, Enter Agility, How to Be Agile?: Agile Methods, Don’t Make Your Own Method, The Road to Mastery, Find a Mentor.

UNIT - II — Lecture Hrs:10

Understanding XP:
The XP Lifecycle, The XP Team, XP Concepts, Adopting XP: Is XP Right for Us? Go!, Assess Your Agility. **Practicing XP:** Thinking: Pair Programming, Energized Work, Informative Workspace, Root-Cause Analysis, Retrospectives, Collaborating: Trust, Sit Together, Real Customer Involvement, Ubiquitous Language, Stand-Up Meetings, Coding Standards, Iteration Demo, Reporting

UNIT - III — Lecture Hrs:10

Releasing:
“Done Done”, No Bugs, Version Control, Ten-Minute Build, Continuous Integration, Collective Code Ownership, Documentation. **Planning:** Vision, Release Planning, The Planning Game, Risk Management, Iteration Planning, Slack, Stories, Estimating. **Developing:** Incremental requirements, Customer Tests, Test-Driven Development, Refactoring, Simple Design ,Incremental Design and Architecture, Spike Solutions, Performance Optimization, Exploratory.

UNIT - IV — Lecture Hrs:10

Mastering Agility Values and Principles:
Commonalities, About Values, Principles, and Practices, Further Reading, Improve the Process: Understand Your Project, Tune and Adapt, Break the Rules, Rely on People: Build Effective Relationships, Let the Right People Do the Right Things, Build the Process for the People, Eliminate Waste: Work in Small, Reversible Steps, Fail Fast, Maximize Work Not Done, Pursue Throughput.

UNIT - V — Lecture Hrs:10

Deliver Value:
Exploit Your Agility, Only Releasable Code Has Value, Deliver Business Results, Deliver Frequently, and Seek Technical Excellence: Software Doesn’t Exist, Design Is for Understanding, Design Trade-offs, Quality with a Name, Great Design, Universal Design Principles, Principles in Practice, Pursue Mastery.

Textbooks:
1. The Art of Agile Development (Pragmatic guide to agile software development), James shore, Chromatic, O'Reilly Media, Shroff Publishers & Distributors, 2007 2. Agile and Iterative Development A Manger's Guide, Craig Larman , First Edition, India, Pearson Education, 2004
Reference Books:
1. The Good, the Hype and the Ugly, Meyer, B., Agile!:, 1st Edition, Springer, 2014, ISBN 978-3-319-05155-0 2. Essential Scrum: A Practical Guide to the Most Popular Agile Process (Addison-Wesley Signature Series (Cohn)), Kenneth S. Rubin , 1stEdition .
Online Learning Resources:

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		Program Elective Course – III	**L**	**T**	**P**	**C**
Semester	**II**	**Data Science**	**3**	**0**	**0**	**3**

Course Objectives:

- Focuses on developing relevant programming abilities.
- Focuses on Computational Methods.
- To develop the understanding of the mathematical and logical basis to many modern techniques in information technology like machine learning, programming language design, and concurrency.
- To study various sampling and classification problems.

Course Outcomes (CO): Student will be able to

CO1: Students will demonstrate proficiency with Statistical Analysis of Data.
CO2: Students will develop the ability to build and assess data-based models
CO3: Students will apply data science concepts and methods to solve problems.
CO4: To understand the basic notions of discrete and continuous probability.
CO5: To understand the methods of statistical inference, and the role that sampling distributions play in those methods.

UNIT - I		Lecture Hrs:10

Introduction: What Is Statistical Learning?, Why Estimate f?, How Do We Estimate f?, The nTrade-Off Between Prediction Accuracy and Model Interpretability, Supervised Versus Unsupervised Learning, Regression Versus Classification Problems, Assessing Model Accuracy, Measuring the Quality of Fit, The Bias-Variance Trade-of, The Classification Setting, Introduction to R, Basic Commands, Graphics, Indexing Data, Loading Data, Additional Graphical and Numerical Summaries.

Unit - II		Lecture Hrs:10

Linear Regression: Simple Linear Regression, Multiple Linear Regression, Other Considerations in the Regression Model, Comparison of Linear Regression with K-Nearest Neighbours, Linear Regression.
Classification: Logistic Regression, Linear Discriminate Analysis, A Comparison of Classification Methods, Logistic Regression, LDA, QDA, and KNN.

Unit - III		Lecture Hrs:10

Probability mass, density, and cumulative distribution functions, Parametric families of distributions, Expected value, variance, conditional expectation, Applications of the univariate and multivariate Central Limit Theorem, Probabilistic inequalities, Markov chains.

Unit –IV		Lecture Hrs:10

Random samples: sampling distributions of estimators, Methods of Moments and Maximum Likelihood, Recent Trends in various distribution functions in mathematical field of computer science for varying fields like bio informatics.

Unit – V		Lecture Hrs:9

Statistical inference: Introduction to multivariate statistical models: regression and classification problems, principal components analysis, the problem of over fitting model assessment.
Graph Theory: Isomorphism, Planar graphs, graph coloring, Hamilton circuits and Euler cycles, Permutations and Combinations with and without repetition, specialized techniques to solve combinatorial enumeration problems.

Textbooks:

1. Gareth James Daniela Witten Trevor Hastie, Robert Tibshirani, An Introduction to Statistical

Learning with Applications in R, February 11, 2013, web link: www.statlearning.com. 2.K. Trivedi, Probability and Statistics with Reliability, Queuing, and Computer Science Applications. Wiley. 3. M. Mitzenmacher and E. Upfal.Probability and Computing: Randomized Algorithms and Probabilistic Analysis.
Reference Books:
1. Sinan Ozdemir, Principles of Data Science, Packt Publishing Ltd Dec 2016. 2. Alan Tucker, Applied Combinatorics, Wiley.
Online Learning Resources:

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		Program Elective Course - IV	**L**	**T**	**P**	**C**
Semester	**II**	Secure Software Engineering	**3**	**0**	**0**	**3**

Course Objectives:

- To evaluate secure software engineering problems.
- To analyze and elicit security requirements using SRS.
- To design and plan software solutions to security problems using various paradigms.
- To model the secure software systems using Unified Modeling Language.

Course Outcomes (CO): Student will be able to

CO1 : Evaluate secure software engineering problems, including the specification, design, implementation, and testing of software systems
CO2 : Elicit, analyse and specify security requirements through SRS
CO3 : Design and Plan software solutions to security problems using various paradigms
CO4 : Model the secure software systems using Unified Modelling Language Sec(UMLSec)
CO5 : Develop and apply testing strategies for Secure software applications

UNIT - I	Lecture Hrs:9

Software assurance and software security, threats to software security, sources of software insecurity, benefits of detecting software security, managing secure software development

UNIT - II	Lecture Hrs:9

Defining properties of secure software, how to influence the security properties of software, how to assert and specify desired security properties

UNIT - III	Lecture Hrs:9

Secure software Architecture and Design: Software security practices for architecture and design: Architectural risk analysis, software security knowledge for Architecture and Design: security principles, security guidelines, and attack pattens, secure design through threat modeling

UNIT - IV	Lecture Hrs:9

Writing secure software code: Secure coding techniques, Secure Programming: Data validation, Secure Programming: Using Cryptography Securely, Creating a Software Security Programs.

UNIT - V	Lecture Hrs:9

Secure Coding and Testing: code analysis- source code review, coding practices, static analysis, software security testing, security testing consideration through SDLC

Textbooks:

1.Julia H Allen, Sean J Barnum, Robert J Ellison, Gary McGraw, Nancy R Mead, Software Security Engineering: A Guide for Project Managers, Addison Wesley, 2008
2.Ross J Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd Edition, Wiley, 2008

Reference Books:

Howard, M. and LeBlanc, D., Writing Secure Code, 2nd Edition, Microsoft Press, 20032.

Online Learning Resources:

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		**Machine Learning Lab**	**L**	**T**	**P**	**C**
Semester	**II**		**0**	**0**	**4**	**2**

Course Objectives:

This course will enable students to
1. Make use of Data sets in implementing the machine learning algorithms
2. Implement the machine learning concepts and algorithms in any suitable language of choice.

Course Outcomes (CO): Student will be able to

1. Understand the implementation procedures for the machine learning algorithms.
2. Design Java/Python programs for various Learning algorithms.
3. Apply appropriate data sets to the Machine Learning algorithms.
4. Identify and apply Machine Learning algorithms to solve real world problems.

Lab Experiments:

1. Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.
2. For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.
3. Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.
4. Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets.
5. Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.
6. Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set.
7. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.
8. Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.
9. Write a program to implement k-Nearest Neighbor algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.
10. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs

Textbooks:
1.John Anderson, Hands On Machine Learning with Python 1st Edition, AI Sciences Publisher, 2018
Reference Books:
1. Michael Bowles, Machine Learning in Python: Essential Techniques for Predictive Analysis 1st Edition, John Wiley, 2015.
Online Learning Resources:
[1] Evaluating a hypothesis, Stanford University, https://www.coursera.org/learn/machine-learning/lecture/yfbJY/evaluating-ahypothesis , Last accessed on 26-8-2019 [2] BalaramanRavindran, NPTEL Lecture 1 - Introduction to Machine Learning, https://www.youtube.com/watch?v=fC7V8QsPBec, Last accessed on 26-8- 2019 [3] Benchmarking Neural Networks on Oracle Cloud Infrastructure with Mapr, https://mapr.com/whitepapers/benchmarking-neural-networks-on-oracle-cloudinfrastructure-with-mapr/ Last accessed on 26-8-2019 [4] George Crump, Dealing with The AI and Analytics Data Explosionhttps://mapr.com/whitepapers/dealing-with-the-ai-and-analytics-dataexplosion/ Last accessed on 26-8-2019

Course Code		Software Testing Lab	L	T	P	C
Semester	II		3	0	0	3

1. Write programs in C Language to demonstrate the working of the following constructs: i) do...while ii) while….do iii) if…else iv) switch v) for
2. A program written in C language for Matrix Multiplication fails. Introspect the causes for its failure and write down the possible reasons for its failure.
3. Consider ATM System and Study its system specifications and report the various bugs.
4. Write the test cases for Banking application.
5. Create test plan document for Library Management System.
6. Create test cases for Railway Reservation.
7. Create test plan document for Online Shopping.

Working with Tool's: Understand the Automation Testing Approach, Benefits, Workflow, Commands and Perform Testing on one application using the following Tool's.

1. Win runner Tool for Testing.
2. Load runner Tool for Performance Testing.
3. Selenium Tool for Web Testing.
4. Bugzilla Tool for Bug Tracking.
5. Test Director Tool for Test Management.
6. Test Link Tool for Open Source Testing.

Reference Books:

Howard, M. and LeBlanc, D., Writing Secure Code, 2nd Edition, Microsoft Press, 20032.

Online Learning Resources:

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		Program Elective Course – V	**L**	**T**	**P**	**C**
Semester	**III**	Blockchain Technologies	**3**	**0**	**0**	**3**

Course Objectives:

- Understand basic crypto currency concepts.
- Understand the working and transactions of bit coin.
- To analyze the function of Blockchain technique.

Course Outcomes (CO): Student will be able to

- Understand crypto currency concepts.
- Should be able to understand the working and transactions of bit coin.
- Should know the different advanced transactions and scripting techniques.
- Knowledge on analyzing the function of Blockchain

UNIT - I		Lecture Hrs:10

Introduction: Bitcoin - History of Bitcoin - Uses, Users, Choosing a Bitcoin Wallet - Quick Start - Getting Your First Bitcoin - Finding the Current Price of Bitcoin - Sending and Receiving Bitcoin - .Transaction Inputs and Outputs - Transaction Chains - Making Change - Common Transaction Forms - Constructing a Transaction - Getting the Right Inputs - Creating the Outputs - Adding the Transaction to the Ledger - Bitcoin Mining - Mining Transactions in Blocks - Spending the Transaction

UNIT – II		Lecture Hrs:10

Bitcoin Core: The Reference Implementation - Bitcoin Development Environment - Compiling Bitcoin Core from the Source Code - Selecting a Bitcoin Core Release - Configuring the Bitcoin Core Build - Building the Bitcoin Core Executables - Running a Bitcoin Core Node - Running Bitcoin Core for the First Time - Configuring the Bitcoin Core Node - Bitcoin Core Application Programming Interface (API) - Getting Information on the Bitcoin Core Client Status - Exploring and Decoding Transactions - Exploring Blocks - Using Bitcoin Core

UNIT – III		Lecture Hrs:10

Wallets and Transactions: Wallet Technology - Overview Nondeterministic (Random) Wallets - Deterministic (Seeded) Wallets - HD Wallets (BIP-32/BIP-44) - Seeds and Mnemonic Codes (BIP-39) - Wallet Best Practices - Using a Bitcoin Wallet - Wallet Technology Details - Mnemonic Code Words (BIP-39) - Creating an HD Wallet from the Seed - Using an Extended Public Key on a Web Store Transactions - Transactions in Detail – Transactions Behind the Scenes - Transaction Outputs and Inputs - Transaction Outputs - Transaction Inputs - Transaction Fees - Adding Fees to Transactions Transaction Scripts 59 and Script

UNIT - IV		Lecture Hrs:10

Advanced Transactions and Scripting: Multisignature -Pay-to-Script-Hash (P2SH) -P2SH Addresses -Benefits of P2SH -Redeem Script and Validation -Data Recording Output (RETURN) -Time locks -Transaction Lock time (nLocktime) -Check Lock Time Verify (CLTV) -Relative time locks -Relative time locks with nSequence -Relative time locks with -The Extended Bitcoin Network , Bloom Filters -How Bloom Filters Work -How SPV Nodes Use Bloom Filters -SPV Nodes and Privacy - Encrypted and Authenticated Connections -Tor Transport -Peer-to-Peer Authentication and Encryption -Transaction Pools

UNIT - V		Lecture Hrs:10

Block chain : The Blockchain Structure of a Block -Block Header -Block Identifiers: Block Header Hash and Block Height -The Genesis Block -Linking Blocks in the Blockchain -Merkle Trees -Merkle Trees and Simplified Payment Verification (SPV) -Bitcoin Test Blockchains - Testing Playground -The Segregated Witness Testnet -The Local Blockchain -

Using Test Blockchains for Development.
Textbooks:
1. Mastering Bitcoin: Programming the Open Block chain, Andreas M. Antonopoulos, Shroff/O'Reilly; Second edition, 2017. 2. Imran BashirMasteringBlockchainPack Publishing Limited ,2016.
Reference Books:
ArshdeepBahga ,Blockchain Applications: A Hands-On Approach , 2017.
Online Learning Resources:

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		Program Elective Course - V	**L**	**T**	**P**	**C**
Semester	**III**	**Software Project Planning &Management**	**3**	**0**	**0**	**3**

Course Objectives:

- Describe and determine the purpose and importance of project management from the perspectives of planning, tracking and completion of project.
- Compare and differentiate organization structures and project structures.
- To discuss the various aspects of project management
- To understand the tasks in software project management
- To describe the requirements of a project plan

Course Outcomes (CO): Student will be able to

CO1 Identify and explore the advantages of agents and design the architecture for an agent
CO2 Analyze the agent in details in a view for the implementation
CO3 Analyze communicative actions with agents.
CO4 Analyze typical agents using a tool for different types of applications.

UNIT - I		Lecture Hrs:10

Manage Your People: Managing project culture, Managing Good People, Making Good People Better, Leading Good People.
Implement Your Process: Putting a process in place, implementing a Process, Adopting a Process.
Leverage Your Tools: Choosing Tools, Training to Use Tools, leveraging Tools. Use Your Measurements: Selecting Measurements, Planning Measurements, Leveraging Measurements.

UNIT - II		Lecture Hrs:10

Form Your Vision: Analyzing Stakeholders, Balancing Project Needs, Ascending Project Risks, Specifying Project Payoffs, Specifying and Communicating a Project Vision. Organize Your Resources: Identifying Hardware, Identifying Software, Identifying Support. Sketch Your Schedule: Estimating Project Size and Effort, Scheduling Immovable Milestones, Scheduling Synchronization Points, Facilitating Communication. Write Your Plan: Organizing the Plan, Covering all the bases, Reviewing the Plan.

UNIT - III		Lecture Hrs:10

Roll Out Your Roles : Identifying Roles, Matching People to Roles, Highlighting Commitments and Dependencies. Schedule Your Schedule: Identifying and Scheduling Tasks, Assigning Tasks to Roles, Creating a Backup Plan, Examining a Case Study. Leaving the Starting Line: Directing the Team, Implementing the Technology, Capturing the Measurements.

UNIT - IV		Lecture Hrs:10

Monitor Your Project: Gathering Information, Understanding the Information, Avoiding Problems, Finding Solutions. Reschedule Your Schedule: Making the Schedule Important, Knowing when the Schedule Slipped, Rescheduling Correctly, Examining a Case Study. Engineer a Great Product: Requiring Your Requirements, Designing in Quality, Implementing Smartly, Testing Effectively.

UNIT - V		Lecture Hrs:10

Deliver Your System: Planning to Finish, Finishing a Plan Supporting a Product Examining a Case Study. Assess your Project: Planning a Project Assessment, Analyzing

Measurements, Presenting the Assessments Results, Examining a Case Study.
Textbooks:
1.Joel Henry, "Software Project Management A Real Word to Guide to Success", Pearson Education, 2004.
Reference Books:
1.Walker Royce, "Software Project Management", Pearson Education, 1998 2. Pankaj Jalote, "Software Project Management in Practice", Addison-Wesley Professional, 2002. 3. Bob Hughes & Mike Cotterell, "Software Project Management", fourth edition,Tata McGraw Hill,2006
Online Learning Resources:

Course Code		Program Elective Course - IV	**L**	**T**	**P**	**C**
Semester	**II**	Software Agents	**3**	**0**	**0**	**3**

Course Objectives:
- To introduce the concept of agents, theirdesign and manipulation.
- To study the various aspects related toagent architecture and communication.
- To understand the concept of agents, theirarchitecture.
- To understand agent communication andtheir role in information sharing.
- To be able to apply the knowledge gainedto implement a software agent.

Course Outcomes (CO): Student will be able to

CO1 Identify and explore the advantages of agents and design the architecture for an agent
CO2 Analyze the agent in details in a view for the implementation
CO3 Analyze communicative actions with agents.
CO4 Analyze typical agents using a tool for different types of applications.

UNIT - I — Lecture Hrs:10

An introduction to Software Agents, Incorporating Agents as Resource Managers, Overcoming user Interface Problems, Toward Agent-Enabled System Architectures. Agents, Artificial Intelligence, Decentralization, Why Linking works, The Theatrical Metaphor, Direct. Interfaces Agents Metaphors with Character: Introduction, Objections to Agents, In Defense of Anthropomorphism, Key Characteristics of Interface Agents, Agency, Responsiveness, Competence, Accessibility, Design and Dramatic Character, An R & D Agenda.

UNIT - II — Lecture Hrs:10

Designing Agents as if People Mattered, The Agent Metaphor, Direct Manipulation versus Agents, Agents for Information Sharing and Coordination, Semiformal Systems an d Radical Tailorability, Oval: A Radically Tailorable Tool for Information Management and Cooperative Work, Examples of Application and Agents in Oval, Conclusions: An Addendum: The Relationship between Oval and Objects Lens

UNIT - III — Lecture Hrs:10

Agents that Reduce Work and Information Overload Introduction, Approaches to Building Agents, Training a Personal Digital Assistant, Some Example of Existing Agents, Acknowledgements Software Agents for Cooperative Learning: Computer-Supported Cooperative Learning, Examples of Software Agents for Cooperative Learning, Examples of Software Agents for Cooperative Learning, Developing an Example, Discussion and Perspectives.

UNIT - IV — Lecture Hrs:10

An Overview of Agent-Oriented Programming: Agent-Oriented Programming, AGENT-0: A Simple Language and its Interpreter, KQML as an Agent Communication Language: The approach of knowledge sharing effort(KSE), The Solution of the knowledge sharing efforts, knowledge Query Manipulation Language (KQML),Implementation, Application of KQML , Other Communication Language, The Approach of Knowledge-Sharing Effect,(KSE),The Solutions of the Sharing Effect.

UNIT - V — Lecture Hrs:10

Agent for Information Gathering: Agent Organization, The Knowledge of an Agent, The Domain Model of an Agent, Modeling other Agent, communication language and protocol, query processing, Mobile Agents: Enabling Mobile Agents, Programming Mobile Agents, Using Mobile Agents.

Textbooks:

1	Software Agents	Jeffrey M. Bradshaw	PHI(MIT Press)	2012

Reference Books:

1		*Lin Padgham and*	*John Wiley & sons*	*2004*

	Developing In telligent Agent Systems: A Practical Guide	*Michael Winikoff*	*Publication*	
2	*Agent-Based and Individual Based modeling: A Practical Introduction*	*Steven F. RailsBack and Volker Grimm*	*Princeton University Press*	*2012*
3	*Disappearing Cryptography – Information Hiding: Steganography & Watermarking*	*Peter Wayner*	*Morgan Kaufmann Publishers*	*2002*
4	*Multimedia Secuirty, Watermarking, Steganography and Forensics*	*Frank Y. Shih*	*CRC Press*	*2012*

Online Learning Resources:

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code		Program Elective Course - IV	**L**	**T**	**P**	**C**
Semester	**II**	Software Agents	**3**	**0**	**0**	**3**

Course Objectives:

- To introduce the concept of agents, theirdesign and manipulation.
- To study the various aspects related toagent architecture and communication.
- To understand the concept of agents, theirarchitecture.
- To understand agent communication andtheir role in information sharing.
- To be able to apply the knowledge gainedto implement a software agent.

Course Outcomes (CO): Student will be able to

CO1 Identify and explore the advantages of agents and design the architecture for an agent
CO2 Analyze the agent in details in a view for the implementation
CO3 Analyze communicative actions with agents.
CO4 Analyze typical agents using a tool for different types of applications.

UNIT - I — Lecture Hrs:10

An introduction to Software Agents, Incorporating Agents as Resource Managers, Overcoming user Interface Problems, Toward Agent-Enabled System Architectures. Agents, Artificial Intelligence, Decentralization, Why Linking works, The Theatrical Metaphor, Direct. Interfaces Agents Metaphors with Character: Introduction, Objections to Agents, In Defense of Anthropomorphism, Key Characteristics of Interface Agents, Agency, Responsiveness, Competence, Accessibility, Design and Dramatic Character, An R & D Agenda.

UNIT - II — Lecture Hrs:10

Designing Agents as if People Mattered, The Agent Metaphor, Direct Manipulation versus Agents, Agents for Information Sharing and Coordination, Semiformal Systems an d Radical Tailorability, Oval: A Radically Tailorable Tool for Information Management and Cooperative Work, Examples of Application and Agents in Oval, Conclusions: An Addendum: The Relationship between Oval and Objects Lens

UNIT - III — Lecture Hrs:10

Agents that Reduce Work and Information Overload Introduction, Approaches to Building Agents, Training a Personal Digital Assistant, Some Example of Existing Agents, Acknowledgements Software Agents for Cooperative Learning: Computer-Supported Cooperative Learning, Examples of Software Agents for Cooperative Learning, Examples of Software Agents for Cooperative Learning, Developing an Example, Discussion and Perspectives.

UNIT - IV — Lecture Hrs:10

An Overview of Agent-Oriented Programming: Agent-Oriented Programming, AGENT-0: A Simple Language and its Interpreter, KQML as an Agent Communication Language: The approach of knowledge sharing effort(KSE), The Solution of the knowledge sharing efforts, knowledge Query Manipulation Language (KQML),Implementation, Application of KQML , Other Communication Language, The Approach of Knowledge-Sharing Effect,(KSE),The Solutions of the Sharing Effect.

UNIT - V — Lecture Hrs:10

Agent for Information Gathering: Agent Organization, The Knowledge of an Agent, The Domain Model of an Agent, Modeling other Agent, communication language and protocol, query processing, Mobile Agents: Enabling Mobile Agents, Programming Mobile Agents, Using Mobile Agents.

Textbooks:

1	Software Agents	Jeffrey M. Bradshaw	PHI(MIT Press)	2012

Reference Books:

1	*Developing In telligent Agent Systems: A Practical Guide*	*Lin Padgham and Michael Winikoff*	*John Wiley & sons Publication*	*2004*
2	*Agent-Based and Individual Based modeling: A Practical Introduction*	*Steven F. RailsBack and Volker Grimm*	*Princeton University Press*	*2012*
3	*Disappearing Cryptography – Information Hiding: Steganography & Watermarking*	*Peter Wayner*	*Morgan Kaufmann Publishers*	*2002*
4	*Multimedia Secuirty, Watermarking, Steganography and Forensics*	*Frank Y. Shih*	*CRC Press*	*2012*

Online Learning Resources:

Course Code		Program Elective Course - IV	**L**	**T**	**P**	**C**
Semester	**II**	**Software Development and IT Services**	**3**	**0**	**0**	**3**

Course Objectives:

- To introduce the concept of agents, theirdesign and manipulation.
- To study the various aspects related toagent architecture and communication.
- To understand the concept of agents, theirarchitecture.
- To understand agent communication andtheir role in information sharing.
- To be able to apply the knowledge gainedto implement a software agent.

Course Outcomes (CO): Student will be able to

CO1 Identify and explore the advantages of agents and design the architecture for an agent
CO2 Analyze the agent in details in a view for the implementation
CO3 Analyze communicative actions with agents.
CO4 Analyze typical agents using a tool for different types of applications.

UNIT - I		Lecture Hrs:10

Unit – I :**The Big Picture:** A Snapshot of Devops Culture, The Evolution of Culture, The Value of the Story, Illustrating Devops with Stories, What is Devops? The Devops equation, A History of Devops, Developer as Operator , The Advent of Software Engineering , The Advent of Proprietary Software and Standardization , The Age of the Network, The Beginnings of a Global Community ,The Age of Applications and the Web , The Growth of Software Development Methodologies , Open Source Software, Proprietary Services , Agile Infrastructure , The Beginning of devopsdays , The Current State of Devops .
Foundational Terminology and Concepts: Software Development Methodologies, Operations Methodologies, Systems Methodologies, Development, Release, and Deployment Concepts, Infrastructure Concepts, Cultural Concepts
Devops Misconceptions and Anti-Patterns: Common Devops Misconceptions, Devops Anti-Patterns,The Four Pillars of Effective Devops

UNIT - II		Lecture Hrs:10

Collaboration: Individuals Working Together, Defining Collaboration, Individual Differences and Backgrounds, Opportunities for Competitive Advantage, Mentorship, Introducing Mindsets, Mindsets and Learning Organizations, the Role of Feedback, Reviews and Rankings, Communication and Conflict Resolution Styles, Communication Context and Power Differentials, Empathy and Trust, Humane Staffing and Resources, Effective Collaboration with Sparkle Corp.
Collaboration: Misconceptions and Troubleshooting: Collaboration Misconceptions, Collaboration Troubleshooting.

UNIT - III		Lecture Hrs:10

Affinity: From Individuals to Teams, What Makes a Team, Teams and Organizational Structure, Finding Common Ground Between Teams, Improving Team Communication, Case Study: United States Patent and Trademark Office, Bene‡ts of Improved Affinity, Requirements for Affinity, Measuring Affinity

Misconceptions and Troubleshooting: Affinity Misconceptions, Affinity Troubleshooting**.**

UNIT - IV		Lecture Hrs:10

Tools: Ecosystem Overview.SoftwareDevelopment,Automation,Monitoring,Metrics,Logging,Alerting,Events,Evolution of the Ecosystem.
Tools: Accelerators of Culture, What Are Tools? Irrelevance of Tools, Selection of Tools, Auditing Your Tool Ecosystem, Case Studies, Examining Etsy, Motivations and Decision-Making Challenges.

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

UNIT - V		Lecture Hrs:10
Scaling: Inflection Points, Understanding Scaling, Organizational Structure, Team Flexibility, Organizational Lifecycle, Complexity and Change, Scaling for Teams. **Case Studies**: Growing and Scaling Teams, Job Postings and Recruitment Issues, Developing Individuals and Teams, Team Scaling and Growth Strategies, Managing Conflict, Scaling for Organizations. **Misconceptions and Troubleshooting:**ScalingMisconceptions, ScalingTroubleshooting.		
Textbooks:		
1. Effective DevOps Building a Culture of Collaboration, A�nity, and Tooling at Scale, Jennifer Davis and Ryn Daniels **2.** 2.DevOpsfor Developers, Michael Hüttermann		
Reference Books:		
Online Learning Resources:		

R21 COURSE STRUCTURE &SYLLABUS FOR M.TECH COURSES
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(SOFTWARE ENGINEERING)

Course Code	21D50301	SOFTWARE DEVELOPMENT AND IT SERVICES-ORDER (OPEN ELECTIVE)	L	T	P	C
Semester	III		3	0	0	3

Course objectives:
- Take user stories and translate them into functioning web applications using HTML, CSS, and JavaScript
- Evaluate alternative approaches to software implementations
- Work through coding issues with analytical debugging techniques

Course Outcomes:

UNIT – I:
The Big Picture: A Snapshot of Devops Culture, The Evolution of Culture, The Value of the Story, Illustrating Devops with Stories, What is Devops? The Devops equation, A History of Devops, Developer as Operator, The Advent of Software Engineering, The Advent of Proprietary Software and Standardization, The Age of the Network, The Beginnings of a Global Community,The Age of Applications and the Web, The Growth of Software Development Methodologies, Open Source Software, Proprietary Services, Agile Infrastructure, The Beginning of devopsdays, The Current State of Devops.
Foundational Terminology and Concepts: Software Development Methodologies, Operations Methodologies, Systems Methodologies, Development, Release, and Deployment Concepts, Infrastructure Concepts, Cultural Concepts
Devops Misconceptions and Anti-Patterns: Common Devops Misconceptions, Devops Anti-Patterns, The Four Pillars of Effective Devops

UNIT – II:
Collaboration: Individuals Working Together, Defining Collaboration, Individual Differences and Backgrounds, Opportunities for Competitive Advantage, Mentorship, Introducing Mindsets, Mindsets and Learning Organizations, the Role of Feedback, Reviews and Rankings, Communication and Conflict Resolution Styles, Communication Context and Power Differentials, Empathy and Trust, Humane Staffing and Resources, Effective Collaboration with Sparkle Corp.
Collaboration: Misconceptions and Troubleshooting: Collaboration Misconceptions, Collaboration Troubleshooting.

UNIT – III:
Affinity: From Individuals to Teams, What Makes a Team, Teams and Organizational Structure, Finding Common Ground Between Teams, Improving Team Communication, Case Study: United States Patent and Trademark Office, Bene‡ts of Improved Affinity, Requirements for Affinity, Measuring Affinity
Misconceptions and Troubleshooting: Affinity Misconceptions, Affinity Troubleshooting**.**

UNIT – IV:
Overview of Software, Automation, Monitoring, Metrics, Logging, Alerting, Events, Evolution of the Ecosystem.
Tools: Accelerators of Culture, What Are Tools? Irrelevance of Tools, Selection of Tools, Auditing Your Tool Ecosystem, Case Studies, Examining Etsy, Motivations and Decision-Making Challenges.

UNIT – V:
Scaling: Inflection Points, Understanding Scaling, Organizational Structure, Team Flexibility, Organizational Lifecycle, Complexity and Change, Scaling for Teams.
Case Studies: Growing and Scaling Teams, Job Postings and Recruitment Issues, Developing Individuals and Teams, Team Scaling and Growth Strategies, Managing Conflict, Scaling for Organizations.
Misconceptions and Troubleshooting: ScalingMisconceptions, ScalingTroubleshooting.

TEXT BOOKS:

1. Effective DevOps Building a Culture of Collaboration, Affinity, and Tooling at Scale, Jennifer Davis and Ryn Daniels
2. DevOpsfor Developers, Michael Hüttermann