ME SOFTWARE ENGINEERING (2014-2016)

SEMESTER-I

S.No.	Course No.	Title	L	T	P	Cr
1	PSE101	Software Engineering Concepts and Methodologies	3	1	0	3.5
2	PSE102	Software Design and Construction	3	0	2	4.0
3	PCL105	Statistical Methods and Algorithms	3	0	2	4.0
4	PSE103	Software Architecture	3	1	0	3.5
5	PSE104	Software Project Management	3	0	2	4.0
6		Elective –I	3	0	2	4.0
		Total	**18**	**2**	**8**	**23**

SEMESTER-II

S.No.	Course No.	Title	L	T	P	Cr
1	PSE201	Software Quality Management	3	1	0	3.5
2	PSE202	Software Verification and Validation Testing	3	0	2	4.0
3	PSE203	Software Metrics	3	1	0	3.5
4	PSE204	Advanced Topics in Software Engineering	3	0	2	4.0
5		Elective-II	3	0	2	4.0
6		Elective-III	3	0	2	4.0
		Total	**18**	**2**	**8**	**23**

SEMESTER-III

S.No.	Course No.	Title	L	T	P	Cr
1	PSE391	Seminar	-	-	-	2.0
2	PSE392	Capstone Project	-	-	-	6.0
		Dissertation (Starts)	-	-	-	-
		Total	-	-	-	**8.0**

SEMESTER-IV

S.No.	Course No	Title	L	T	P	Cr
1	PSE091	Dissertation	-	-	-	12.0
		Total	**-**	-	-	**12.0**
		Grand Total of Four Semester Credits				**66.0**

LIST OF ELECTIVES

S.No.	Course No.	Title	L	T	P	Cr
1	PSE206	Agile Software Development Approaches	3	0	2	4.0
2	PSE207	Component Based Development	3	0	2	4.0
3	PSE208	Service Oriented Architecture	3	0	2	4.0
4	PCS104	Advance Data Structures and Algorithms	3	0	2	4.0
5	PCS106	Parallel and Distributed Computing	3	0	2	4.0
6	PCS204	Advance Information Management System	3	0	2	4.0
7	PCS205	Big Data and Business Intelligence	3	0	2	4.0
8	PCS206	Machine Learning	3	0	2	4.0
9	PCS211	Cloud Infrastructure and Services	3	0	2	4.0
10	PIS105	Secure Coding	3	0	2	4.0
11	PIS106	Advanced Computer Networks	3	0	2	4.0
12	PIS204	Network Security and Ethical Hacking	3	0	2	4.0

PSE101 SOFTWARE ENGINEERING CONCEPTS AND METHODOLOGIES

L	T	P	Cr
3	1	0	3.5

Course Objectives: To apply principles of software development and evolution. To specify, abstract, verify and validate solutions to large-size problems, to plan, develop and manage large software and learn emerging trends in software engineering.

Continuous Assessment Scheme : MST - 20 ; Sessional – 35 ; EST – 45

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

Software Development Methods: Formal, semi-formal and informal methods; Requirements elicitation, requirements specification; Data, function, and event-based modeling; Popular methodologies such as Yourdons SAD, SSADM; Managing the Software Projects

Software Engineering Tools and Environments: upper and lower CASE tools, evolution of CASE tools-classification, features, strengths and weaknesses; ICASE; CASE standards. Role of the repository for supporting incremental development, software reuse

Software Quality Assurance: SQA Tasks, Goals and Metrics, Software Review Techniques: Informal reviews-Formal Technical Reviews, Software Reliability, Software risk management, Case Studies. Real Time Systems

Configuration Management: Need, Configuration management functions and activities; Configuration management techniques; Case studies.

Software Testing Fundamentals: Basic Terminology, Testing Techniques and strategies. Brief introduction to various standards related to Software Engineering.

Recommended Books

1. Pressman, Roger, Software Engineering - A Practitioners Approach, McGraw Hill ,2014 8th ed.
2. Waman Jawadekar, Software Engineering: Principles & Practices, 1st edition 2004
3. Sommerville, Ian, Software Engineering, Addison-Wesley Publishing Company, 2006 8th ed.
4. Jalote, Pankaj, An integrated Approach to Software Engineering, Narosa, 2005.

COURSE OUTCOMES (COs)

CO1	Identify the need for engineering approach to software development and various processes of requirements analysis for software engineering problems.
CO2	Apply various software engineering models and apply methods for design and development of software projects.
CO3	Apply various techniques, metrics and strategies for Testing software projects.
CO4	Identify and apply the principles, processes and main knowledge areas for Software Project Management
CO5	Apply standards, CASE tools and techniques for engineering software projects

PSE 102 SOFTWARE DESIGN AND CONSTRUCTION				
	L	T	P	Cr
	3	0	2	4

Course Objective: To gain knowledge of software construction fundamentals, managing construction and practical considerations related to the domain of software design and construction.

Continuous Assessment Scheme : MST – 20 ; Lab -20;Sessional – 20 ; EST – 40

Software Design: Design concepts, design model, software architecture, architectural design, data design, component level design, and user interface design.

Object Modeling and Design: OMT, visual odelling, UML, Rational Rose Tool, Classes, objects, relationships, key abstractions, common mechanisms, diagrams, class diagrams, advanced classes, advanced relationships, interfaces, types, roles, packages, instances, object diagrams, interactions, use cases, use case diagrams, interaction diagrams, activity diagrams, events and signals, state machines, processes, threads, state chart diagrams, components, deployment, collaborations, patterns and frameworks, component diagrams, systems and models, code generation and reverse engineering.

Software Construction: Object-oriented approach, object-oriented programming and languages, Scope of class members-public, private, protected. Class constructor, destructor, copy constructor, virtual destructor. Derived classes, scope of derivation-public, private, protected. Virtual functions, Function overloading. Friend functions and friend classes, Operator overloading, dynamic memory allocation to classes and class members, new and delete operators. Overloading new and delete operators. Explicit type conversion operators. Input output streams, Stream class hierarchies, standard I/O objects: cin, cout, cerr, overloading <<, >> operators, File Streams, opening, reading, writing to file. File pointers and their manipulators, Introduction to templates and container classes.

Laboratory Work : Design and Modeling with Rational Rose, implementation-using Object oriented programming.

Recommended Books

1. Object-Oriented Analysis and Design with Applications, Grady Booch 3rd Edition,2007
2. The Unified Modeling Language User Guide, Grady Booch, James Rumbaugh, Ivar Jacobson, Addison-Wesley Professional, 2nded, 2005

COURSE OUTCOMES (COs)

CO1	Specify various elements of object odelling to identify, analyze, visualize, specify, model and design
CO2	Apply analysis and design principles at various levels and various views in different domains of software systems.
CO3	Represent engineering problems graphically by drawing all UML diagrams.
CO4	Identify and apply concepts of software construction like Object Oriented Programming skills
CO5	Skillful use of Rational Rose tool for drawing all the UML diagrams in order to forward and reverse engineer the complex software engineering problems.

PSE 103 SOFTWARE ARCHITECTURE

L	T	P	Cr
3	1	0	3.5

Course Objective: To gain familiarity with the issues associated with large-scale software architectures, frameworks, patterns, components, tools and techniques that may be used for the automatic analysis, documentation and evaluation of software.

Continuous Assessment Scheme : MST - 20 ; Sessional – 35 ; EST - 45

Basics of Software Architecture: Architecture Business Cycle, Architecture Patterns, Reference Model and Reference Architecture, Architecture Structure and Views, Product Line Architecture, Functional and Non-functional Properties of Software Architectures.

Enabling Techniques for Software Architecture: Coupling and Cohesion, Sufficiency, Completeness and Primitiveness, Separation of Policy and Implementation, Separation of Interface and Implementation.

Architectural Styles: Pipes and Filters, Data Abstraction and Object-Orientation, Event-Based, Implicit Invocation, Layered Systems, Repositories, Interpreters, Process Control, Heterogeneous Architectures, Case studies based on architectural styles.

Understanding and Achieving Software Qualities: Changeability, Efficiency, Interoperability, Reliability, Testability, Reusability, Security, Usability, Fault tolerant software, Tactics to achieve software qualities.

Designing of Software Architecture: Function Oriented Design, Object Oriented Design, Attribute Driven Design of Software Architecture, Case Studies.

Documenting Software Architecture: Software Architecture Documentation Template, Use of Documentation, Creation of different views of Software Architecture with UML.

Reconstructing Software Architecture: Phases of Reconstruction, Uses of Reconstruction, Reconstruction of Software Architecture using tool.

Recommended Books

1. Software Architecture in Practice, SEI Series in Software Engineering, Len Bass,Paul Clements, Rick Kazman,3rd Edition, 2012
2. Patterns of Enterprise Application Architecture, Martin Fowler, Addison-Wesley Signature Series, 1st Edition, 2012
3. The Unified Modeling Language User Guide, Booch G., Rumbaugh J., Jacobson

COURSE OUTCOMES (COs)

CO1	Comprehend about Architecture Business Cycle, Architecture Patterns, Reference Model
CO2	Able to differentiate between various architecture styles
CO3	To familiarize and gain knowledge of various software qualities
CO4	Designing and documenting and reconstructing software architecture

PSE104 SOFTWARE PROJECT MANAGEMENT				
	L	**T**	**P**	**Cr**
	3	**0**	**2**	**4.0**

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

Continuous Assessment Scheme : MST - 20 ; Lab – 20;Sessional-20 ; EST - 40

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

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

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

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

Measurement and Tracking Planning: Earned Value Analysis.

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

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

Project Monitoring and Control: Audits and Reviews.

Laboratory Work: Implementation of software project management concepts using tools like MS Project, Rational Suite (RequisitePro, Purify, etc.), Advanced Cost Estimation Models.

Recommended Books

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

COURSE OUTCOMES (COs)

CO1	Apply the basics of Software Project Management in order to manage and deliver qualified product.
CO2	Identify the Problem Effectively and Efficiently with proper documentation for the use in different software teams and organization.
CO3	Comprehend and be able to carry on Technical as well as Cost Benefit Analysis and plan the activities within time schedules with CPM and PERT Analysis.
CO4	Competent to design Communication Plans, Procurement of Resources and Human Resource Management.

PSE201 SOFTWARE QUALITY MANAGEMENT				
	L	**T**	**P**	**Cr**
	3	**1**	**0**	**3.5**

Course Objectives: This course aims to equip students with the knowledge and techniques of professional practices in software processes and activities. It prepares students to manage the development of high quality software using proven techniques and established standards in software quality management.

Continuous Assessment Scheme : MST - 20 ; Sessional – 35 ; EST - 45

Concepts and Overview: Concepts of software quality, quality attributes, software quality control and software quality assurance, evolution of SQA, major SQA activities, major SQA issues, zero defect software, Zero defect techniques.

Software Quality Assurance: The philosophy of assurance, the meaning of quality, the relationship of assurance to the software life cycle, SQA techniques. Tailoring the Software Quality Assurance Program: Management review process, technical review process, walkthrough, software inspection process, configuration audits, Document verification. Quality Assurance through Process maturity, CMM, PSP, TSP.

Evaluation: Quality evaluations of Software requirements, preliminary design, detailed design, coding and unit test, integration and testing, system testing, Clean Engineering.

Error Reporting: Identification of defect, analysis of defect, correction of defect, implementation of correction, regression testing; Categorization of defect, relationship of development phases.

Trend Analysis: Error quantity, error frequency, program unit complexity, compilation frequency.

Corrective action as to Cause: Identifying the requirement for corrective action, determining the action to be taken, implementing the corrective action, Documenting the corrective action, periodic review of actions taken.

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

Recommended Books

1. Practical Guide to Software Quality Management (Artech House Computing Library) 2nd edition, 2003
2. Quality Software Management, Volume 1: Systems Thinking, 2011, Dorset House Publishing

COURSE OUTCOMES (COs)

CO1	To have basic knowledge of Software quality models
CO2	Understand Quality measurement and metrics, Quality plan, implementation and documentation
CO3	Understand Quality tools including CASE tools
CO4	Understand and know Quality control and reliability of quality process and Quality management system models
CO5	Understand Complexity metrics and Customer Satisfaction and International quality standards – ISO, CMM

PSE 202 SOFTWARE VERIFICATION AND VALIDATION TESTING

L	T	P	Cr
3	0	2	4.0

Course Objectives: This course makes students understand the concepts and theory related to software testing. Understand different testing techniques used in designing test plans, developing test suites, and evaluating test suite coverage. Understand how software developers can integrate a testing framework into code development in order to incrementally develop and test code.

Continuous Assessment Scheme : MST - 20 ; Lab-20;Sessional – 20 ; EST - 40

Introduction: Terminology, evolving nature of area, Errors, Faults and Failures, Correctness and reliability, Testing and debugging, Static and dynamic testing, Exhaustive testing: Theoretical foundations: impracticality of testing all data, impracticality of testing all paths, no absolute proof of correctness.

Software Verification and Validation Approaches and their Applicability: Software technical reviews; Software testing: levels of testing - module, integration, system, regression; Testing techniques and their applicability-functional testing and analysis, structural testing and analysis, error-oriented testing and analysis, hybrid approaches, integration strategies, transaction flow analysis, stress analysis, failure analysis, concurrency analysis, performance analysis; Proof of correctness; simulation and prototyping; Requirement tracing.

Test Generation: Test generations from requirements, Test generation pats, Data flow analysis, Finite State Machines models for flow analysis, Regular expressions based testing, Test Selection, Minimizations and Prioritization, Regression Testing.

Program Mutation Testing: Introduction, Mutation and mutants, Mutation operators, Equivalent mutants, Fault detection using mutants, Types of mutants, Mutation operators for C and Java.

Laboratory Work: To Use various verification and validation testing tools and to apply these tools on few examples and case studies

Recommended Books

1. Software Verification and Validation: An Engineering and Scientific Approach, Marcus S. Fisher, Springer, 2007
2. Foundations of Software Testing, Aditya P. Mathur,Pearson Education, 2008
3. Software Testing: Principles and Practices, Srinivasan Desikan, Gopalaswamy Ramesh, Pearson Education India,2006

COURSE OUTCOMES (COs)

CO1	Capable to comprehend the concepts related to theoretical foundations of testing and debugging.
CO2	Competent to know and demonstrate software verification and validation approaches and their applicability.
CO3	Proficient to formulate and generate test cases from specifications
CO4	Able to exemplify program mutation testing strategies using programming language.
CO5	Proficient to formulate and generate test cases from finite state machine model etc.

PSE203 SOFTWARE METRICS				
	L	**T**	**P**	**Cr**
	3	**1**	**0**	**3.5**

Course Objectives: To teach students software metrics and issues related to that and provide students with skills needed to do independent research in the area.

Continuous Assessment Scheme : MST - 25 ; Sessional – 30 ; EST - 45

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

Software-metrics data collection and analysis: What is good data, how to define the data, how to collect the data, how to store and extract data, analyzing software-measurement data, frequency distributions, various statistical techniques.

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

Measuring external product attributes: Modeling software quality, measuring aspects of software quality, software reliability, basics of software reliability, software reliability problem, parametric reliability growth models, predictive accuracy, recalibration of software-reliability growth predictions, importance of operational environment, wider aspects of software reliability.

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

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

Metrics for component-based systems: The intent of component-based metrics, distinguishing characteristics of component-based metrics, various component-based metrics.
Resource measurement: Measuring productivity, teams, tools, and methods.

Recommended Books

1. Metrics and Models in Software Quality Engineering 2 Edition, Pearson, 2003.
2. Applied Software Measurement by Capers Jones, Tata McGraw Hill, 2008 3rded

COURSE OUTCOMES (COs)

CO1	Understand the basics of Software Measurement, its underlying objectivity using the quantifiable approach in order to control, manage and Improvise quality.
CO2	Able to apply appropriate software measurement scales according to the characteristic of the projects.
CO3	Learn to manage the project and processes, apply configuration management on the basis of collected metrics.
CO4	Able to design and conduct surveys, case studies and experimentation. And also analyse the collected data and draw conclusions and appropriate results graphically.
CO5	Able to define, design or redesign new direct or indirect metrics.

PSE 204 ADVANCED TOPICS IN SOFTWARE ENGINEERING				
	L	**T**	**P**	**Cr**
	3	**0**	**2**	**4.0**

Course Objectives: To apply advance topics in software engineering. To specify, abstract, verify and validate solutions to large-size problems, to plan, develop and manage large software using state-of-the-art methodologies and learn emerging trends in software engineering

Continuous Assessment Scheme : MST - 20 ; Lab-20;Sessional – 20 ; EST – 40

Formal Methods: Basic concepts, mathematical preliminaries, Applying mathematical notations for formal specification, formal specification languages, using Z to represent an example software component, the ten commandments of formal methods, formal methods- the road ahead.

Cleanroom Software Engineering: approach, functional specification, design and testing

Component-Based Software Engineering: CBSE process, domain engineering, component-based development, classifying and retrieving components, and economics of CBSE.

Client/Server Software Engineering: Structure of client/server systems, software engineering for Client/Server systems, analysis modeling issues, design for Client/Server systems, testing issues.
Web Engineering: Attributes of web-based applications, the WebE process, a framework for WebE, formulating, analyzing web-based systems, design and testing for web-based applications, Management issues.

Reengineering: Business process reengineering, software reengineering, reverse reengineering, restructuring, forward reengineering, Economics of reengineering.

Computer-Aided Software Engineering: Building blocks for CASE, taxonomy of CASE tools, integrated CASE environments, integration architecture, CASE repository, case Study of tools like TCS Robot.

Mobile Development Process: Model View Controller, Presentation Abstraction Control, UML based development, Use cases, Testing: Mobile infrastructure, Validating use cases, Effect of dimensions of mobility on testing, Case study: IT company, Requirements, Detailed design, Implementation.

Real Time Operating Systems: Real-time and non-real time applications. Classification of Real-Time Task scheduling algorithms, Event-driven scheduler- Simple priority-based, Rate Monotonic Analysis, Earliest Deadline First, The simplest of Task assignment and scheduling, priority scheduling, characteristics of tasks, task assignment and multi-tasking.

Software Engineering Issues in Embedded Systems: Characteristics of embedded systems I/O, Embedded systems/real time systems. Embedded software architecture, control loop, interrupts control system, co-operating multitasking, pre-emptive multitasking, Domain analysis, Software element analysis, requirement analysis, Specification, Software architecture, Software analysis design, implementation, testing, validation, verification and debugging of embedded systems.

Laboratory Work: To implement the advance concepts in the lab using related tools and to develop the project using related technology

Recommended Books

1. Software Engineering a Practitioners Approach, Roger S. Pressman, McGraw-Hill , 8th Edition, 2014
2. Formal Specification and Documentation using Z - A Case Study Approach, J.Bowan , International Thomson Computer Press, 2003

COURSE OUTCOMES (COs)

CO1	Acquire knowledge on the wider perspective of software engineering and architecture Issues
CO2	Implement the mathematical notation of the software systems through formal methods.
CO3	Design and construct the software systems using reusable software “components” by acquiring the knowledge about domain engineering and component based development
CO4	Learn to merge the conventional principles, concepts and methods in software engineering with the elements of object oriented and CBSE to create client/server systems.
CO5	To create high quality web applications by using software engineering concepts and principles like formulation, planning, analysis testing and evaluation.

PSE206 AGILE SOFTWARE DEVELOPMENT APPROACHES				
	L	**T**	**P**	**Cr**
	3	**0**	**2**	**4.0**

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

Continuous Assessment Scheme : MST - 20 ; Lab- 20;Sessional – 20 ; EST - 40

Agile Software Development: Basics and Fundamentals of Agile Process Methods, Values of Agile, Principles of Agile, stakeholders, Challenges

Lean Approach: Waste Management, Kaizen and Kanban, add process and products add value. roles related to the lifecycle, differences between Agile and traditional plans, differences between Agile plans at different lifecycle phases. Testing plan links between testing, roles and key techniques, principles, understand as a means of assessing the initial status of a project/ How Agile helps to build quality

Agile and Scrum Principles: Agile Manifesto, Twelve Practices of XP, Scrum Practices, Applying Scrum. Need of scrum, working of scrum, advanced Scrum Applications, Scrum and the Organization , scrum values

Agile Product Management: Communication, Planning, Estimation Managing the Agile approach Monitoring progress, Targeting and motivating the team, Managing business involvement, Escalating issue. Quality, Risk, Metrics and Measurements, Managing the Agile approach Monitoring progress, Targeting and motivating the team, Managing business involvement and Escalating issue

Agile Requirements: User Stories, Backlog Management. Agile Architecture**:** Feature-Driven Development. Agile Risk Management**:** Risk and Quality Assurance, Agile Tools
Agile Testing: Agile Testing Techniques, Test-Driven Development, User Acceptance Test
Agile Review: Agile Metrics and Measurements, The Agile approach to estimating and project variables, Agile Measurement, Agile Control: the 7 control parameters. Agile approach to Risk, The Agile approach to Configuration Management, The Atern Principles , Atern Philosophy ,The rationale for using Atern, Refactoring, Continuous integeration, Automated Build Tools

Scaling Agile for large projects: Scrum of Scrums, Team collaborations, Scrum, Estimate a Scrum Project, Track Scrum Projects, Communication in Scrum Projects, Best Practices to Manage Scrum.
Laboratory Work: exploring the tools related to Agile Development and approached and develop small projects in this technology.

Recommended Books

1. Agile Software Development, Principles, Patterns, and Practices (Alan Apt Series) Robert C. Martin (Author) ,2011
2. Succeeding with Agile : Software Development Using Scrum, Pearson 2010

COURSE OUTCOMES (COs)

CO1	Analyze existing problems with the team, development process and wider organization
CO2	Apply a thorough understanding of Agile principles and specific practices
CO3	Select the most appropriate way to improve results for a specific circumstance or need
CO4	Judge and craft appropriate adaptations to existing practices or processes depending upon analysis of typical problems
CO5	Evaluate likely successes and formulate plans to manage likely risks or problems

PSE207 COMPONENT BASED DEVELOPMENT

L	T	P	Cr
3	0	2	4.0

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

Continuous Assessment Scheme : MST - 20 ; Lab-20;Sessional – 20 ; EST - 40

Component Definition: Definition of Software Component and its Elements. Component Models and Component Services: Concepts and Principles, COTS Myths and Other Lessons Learned in Component-Based Software Development, Roles for Component-Based Development, Common High Risk Mistakes in Component-Based Software Engineering, CBSE Success Factors: Integrating Architecture, Process, and Organization.

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

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

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

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

Legal and Regulatory: CBSE as a Unique Engineering Discipline, The Future of Software Components: Standards and Certification, Commercial Law Applicable to Component-Based Software, The Effects of UCITA on Software Component Development and Marketing, Future of CBSE.

Laboratory Work: Practice, Implementation and working of Component Based Development tools and technologies

Recommended Books

1. Component-Based Development: Principles and Planning for Business Systems, Addison Wilsey, 2010
2. Essential COM, Don Box, Dorling Kingsley, 2006.

COURSE OUTCOMES (COs)

CO1	Familiarization with Component Based Systems, their Purpose and Scope.
CO2	To understand Software Engineering Practices related to CBD.
CO3	To Understand design Of Software Component Infrastructures
CO4	To know Component Based Development Technologies
CO5	To understand the concept of Legal and regulatory framework related to CBD

PSE 208 SERVICE OIRENTED ARCHITECTURE				
	L	**T**	**P**	**Cr**
	3	**0**	**2**	**4.0**
Course Objectives: To introduce the concepts and design principles of SOA, Non-technical aspects such as governance, impact on culture and organization, as well as the various interoperability standards, technology infrastructure and security considerations associated with SOA implementations.				
Continuous Assessment Scheme : MST - 20 ; Sessional – 35 ; EST - 45				

Introduction: Roots, Characteristics and Anatomy of SOA, Comparing SOA to client- server and distributed internet architectures, SOA component interrelation, Principles of service orientation

Service Oriented Architecture: Major components of the architecture SOAP, XML, HTTP, Cookies, WSDL, XML schema, UDDI, Interactions between components.

Introduction to Web services : Service descriptions , Messaging with SOAP ,Message exchange Patterns , Coordination ,Atomic Transactions , Business activities , Orchestratiory,Choreography ,Service layer abstraction , Application Service Layer , Business Service Layer , Orchestration Service Layer

Analysis: Service oriented analysis ,Business-centric SOA , Deriving business services-service modeling ,Service Oriented Design , WSDL basics , SOAP basics , SOA compositionguidelines ,Entity-centric business service design ,Application service design , Task centric business service design

SOA platform basics: SOA support in J2EE ,Java API for XML-based web services (JAX-WS), Java architecture for XML binding (JAXB) ,Java API for XML Registries (JAXR) ,Java API for XML based RPC (JAX-RPC),Web Services Interoperability Technologies (WSIT) , SOA support in .NET , Common Language Runtime , ASP.NET web forms , ASP.NET web services , Web Services Enhancements (WSE)

Security: WS-BPEL basics , WS-Coordination overview , WS-Choreography, WS-Policy, WS-Security

Laboratory work: Installing and configuring web servers, building and implementing Web services using the latest tools (.NET, J2EE)

Recommended Books

1. Service-Oriented Architecture: Concepts, Technology, and Design, Thomas Erl, Pearson Education, 2005
2. Achieving Service-Oriented Architecture: Applying an Enterprise Architecture Approach, Rick Sweeney, 2010

COURSE LEARNING OUTCOMES (COs)

CLO1	Analyze functions of Service Oriented Architecture and identify the ways in which they can benefit organizations and study the comparison of web services with other technologies.
CLO2	Evaluate the design of SOA, Major components of the architecture SOAP, XML, HTTP, Cookies, WSDL, XML schema, UDDI and Interactions between various components.
CLO3	Learn some of Semantic Web technologies and applications with knowledge of XML's, Grammar rules, namespace schema.
CLO4	Create web services and web services clients with state-of-the-art tools along
CLO5	Exemplify the web service interoperability, security, and future of web services with the implementation of cloud computing