SOFTWARE ARCHITECTURE

Semester II (Computer Engineering)
SUB CODE: MECE202

TEACHING SCHEME (Credits and Hours):

Teaching scheme				Total	Evaluation Scheme					
L	T	P	Total	Credit	Theory		Mid Sem Exam	CIA	Pract.	Total
Hrs	Hrs	Hrs	Hrs		Hrs	Marks	Marks	Marks	Marks	Marks
04	00	02	06	05	3	70	30	20	30	150

LEARNING OBJECTIVES:

Complex software systems require abstraction and analysis at an architectural level of abstraction. In this course we study, typical software system structures (architectural styles), techniques for designing and implementing these structures, models for characterizing and reasoning about architectures, and tools for architectural modeling. Role of architecture in Software engineering; Enterprise Architectures, Zachman's Framework; Architectural Styles, Design Patterns; Architecture Description Languages; Product-line architectures; Component based development.

OUTLINE OF THE COURSE:

Unit No	Topics
1	Introduction
2	Architectural Styles and Patterns
3	Models for characterizing and reasoning about architectures
4	Design of Software system structures
5	Modeling
6	Visualization and Architecture Description Languages
7	Implementation and Evaluation of the Architecture
8	Domain-Specific Software Architectures
9	Product-line architectures and Component based development
10	Enterprise Architectures

Total hours (Theory): 60

Total hours (Practical): 30

Total hours: 90

DETAILED SYLLABUS:

Sr. No	**Topic**	**Lecture Hours**	**Weight age (%)**
1	**Introduction** • The Architecture Business Cycle: Where do architectures come from? • Software processes and the architecture business cycle; • What makes a "good" architecture? • What software architecture is and what it is not; • Other points of view; Architectural patterns, reference models and reference architectures; • Importance of software architecture; Architectural structures and views. • Software Architecture Elements: Components, Connectors and Configuration	06	10
2	**Architectural Styles and Patterns** • Introduction • Architectural Patterns and Architectural styles • From mud to structure: Layers, Pipes and Filters, Blackboard, Data abstraction and object-oriented organization etc	08	15
3	**Models for characterizing and reasoning about architectures** • Formal Models • View and Viewpoints • Specification	06	05
4	**Design of Software system structures** • Design Strategy • The Attribute-Driven Design Method • The steps of ADD	08	15

5	**Modeling** • Basic Concepts • Ambiguity, Accuracy and Precision • Specific Modeling Techniques: Generic, ADL's etc **Tools for architectural modeling** • ER/Studio Software Architect • IBM Rational Software Architect etc	06	05
6	**Visualization and Architecture Description Languages** • Basic Concepts • Issues in Visualization • Techniques: Textual Visualization, UML, ADL etc • ADL's today • Capturing Architectural Information in an ADL • Application of ADL's in system Development • Choosing an ADL • Example of ADL	06	15
7	**Implementation and Evaluation of the Architecture** • Architecture and its Implementation • Architecture Tradeoff Analysis Method • Participants, Outputs and Phases in ATAM	04	05
8	**Domain-Specific Software Architectures** • What are DSSA; • How DSSA help development; • Components of DSSA; • DSSA based Software development;	05	10
9	**Product-line architectures** • Creating Products • Role of a Product Line Architecture • Evaluating a Product Line Architecture **Component based development** • Component Based Systems	06	10

10	**Enterprise Architectures** • J2EE • .NET, • Model DrivenArchitecture • Zachman's Framework	05	10

INSTRUCTIONAL METHOD AND PEDAGOGY (Continuous Internal Assessment (CIA) Scheme)

- At the start of course, the course delivery pattern, prerequisite of the subject will be discussed.
- Lectures will be conducted with the aid of multi-media projector, white/black board, OHP etc.
- Attendance is compulsory in lecture and laboratory which carries 10 marks in overall evaluation.
- One internal exam will be conducted as a part of internal theory evaluation.
- Assignments based on the course content will be given to the students for each unit and will be evaluated at regular interval evaluation.
- Surprise tests/Quizzes/Seminar/tutorial will be conducted having a share of five marks in the overall internal evaluation.
- The course includes a laboratory, where students have an opportunity to build an appreciation for the concepts being taught in lectures.
- Experiments shall be performed in the laboratory related to course contents.

STUDENTS LEARNING OUTCOMES:

On successful completion of the course, the student will:

- Be familiar with software architecture, its foundation, principles, and elements.
- Examine the useful abstractions and paradigms of system design as well as key notations and tools. They will learn the introduction to software architecture that illustrates the current state of the discipline and examines ways in which architectural issues can impact software design.
- Learn Software System Organization.
- Be able to develop a repertoire of useful techniques that allows them to approach systems from an architectural point of view.
- Learn architectural techniques (e.g., architectural recovery, architectural styles, domain specific software architectures) to design and implement a real-world software system.
- Understand important concepts, methods, languages, and tools.
- Be exposed to the concepts, principles, and state-of- the-art methods in software architectures, including domain-specific software architectures (DSSA), architectural styles, architecture description languages (ADL), software connectors, dynamism in architectures, and architecture-based testing and analysis.

- Understand the explicit boundaries of the field and its relationship to other areas of software engineering, specifically requirements, design (including object-oriented design and related notations, such as UML), and implementation.
- Learn the practical applicability of architecture research, specifically its relationship to the work in software reuse and component interoperability platforms (such as CORBA, JavaBeans, and COM/DCOM).
- Be introduced to the state-of-the-art in software architecture research, future trends and state-of-the-practice.

REFERENCE BOOKS:

1. R. N. Taylor, N. Medvidovic, and E. M. Dashofy. ***Software Architecture: Foundations, Theory, and Practice***, John Wiley & Sons, 2009.
2. Mary Shaw and David Garlan: ***Software Architecture- Perspectives on an Emerging Discipline***, Prentice-Hall of India, 2007.
3. Len Bass, Paul Clements, Rick Katzman, Ken Bass. ***Software Architecture in Practice*.**
4. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Miachel Stal, Douglas Schmidt. ***Pattern Oriented Sofware Architecture***, Volumes 1 &2
5. George T. Heineman, William T. Councill. ***Component Based Software Engineering.***
6. Kurt Wallnau, Scott Hissam and Robert Seacord. ***Building Systems from Commercial Components.***

LIST OF PRACTICALS:

Sr. No	**Name of Experiment**
1	Develop a simple application using CORBA (eg a Calculator).
2	Develop a simple application using Javabeans.
3	Develop a simple application using COM/DCOMs.
4	Provide an architectural breakdown for a simple software system.
5	Choose software architecture for a particular problem and identify strengths and weaknesses of your own and other people's solutions.
6	Provide an architectural description (using the discussed ADL) for the same software system as in Practical 4.
7	Implement the architectural description of Practical 4.
8	Implement any one design pattern in Java, and analyze the costs and benefits of applying the design pattern.
9	Develop a simple application in MTSA.
10	A group assignment that provides hand-on experience with an advanced software architecture topic.