COURSE OUTLINE

Course Code: MCC1093
Course Name : Advanced Software Engineering
Total Contact Hours: xx hours
Course Pre-requisite: None

SYNOPSIS

This course will expose students to the concepts, principles, and state-of-the-art methods and use of UML in object-oriented analysis, software design, software pattern and software architecture, including domain-specific software architectures, architectural styles, their properties and the types of problems for which they are most appropriate, and architecture-based testing and analysis. The course also examines the practical applicability of architecture research, specifically its relationship to work in architectural frameworks and component interoperability platforms such as CORBA, J2EE, and .NET. Particular emphasis will be given on adopting object oriented analysis and design in software engineering.

LEARNING OUTCOMES

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

Upon successful completion of this course, the student will be able to:

- Understand the roles of key elements of software engineering such as requirement analysis, software design, and software testing.
- Understand the principles of modern software process.
- Understand and apply principles of object-oriented analysis and design, and the corresponding UML notation including:
  - Think in objects
  - Analyze requirements with use cases
  - Create domain model
  - Apply UP
  - Relate analysis and design artifacts
  - Assign responsibility to objects
  - Design collaborations
  - Design with patterns
  - Design with architectural layers
  - Understand OOP mapping issues
- Understand and apply principles and patterns in software design
- Understand and use standard techniques for software testing

STUDENT LEARNING TIME

Evaluation Scheme:

*Assignments and projects	:	25%
*Seminar	:	10%
*Tests	:	30%
*Final exams	:	35%

*AP&S1 - 17%, AP&S2 –9% & AP&S3 – 9%

Assignments and projects:
Each student is required to select one project of a reasonable size at the beginning of semester to be used as basis for assignments and class project. The collection of assignments from the beginning of semester will be compiled and consolidated as a complete project to be submitted as project at the end of semester.

Seminar
Each student is also required to select papers from a collection of journal papers, summarize it and present to the class at specified date.

Tests and final exams :
The tests and exams will be in the forms of closed books and closed notes.

WEEKLY SCHEDULE

Week	Topics	Activities/hours
1	Registration & Introduction	
2	SE Preview	
3	Introduction, OOA & OOD, Unified Process	
4	Requirements, use cases, Inception phase	
5	Elaboration phase and domain models	
6	System Sequence Diagram, operation contracts	
7	Logical architecture and UML Package Diagram & Design Pattern – Introduction	
8	Interaction diagram, Collaboration diagram, Class Diagram	
9	Visible Design & Test-Driven Development	
10	Semester Break	
11	GRASP	
12	GoF	
13	Architecture Analysis & Refinement	
14	Package Design, UML deployment & component diagrams	

REFERENCES :

Main Text:

C. Larman, *Applying UML and Patterns,* Third edition, Prentice-Hall, 2005.

Other References:

1. C. Hofmeister, R. Nord, and D. Soni, *Applied Software Architecture,* Addison Wesley, 2000 (ISBN: 0-201-32571-3).
2. E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns – Elements of Reuseable Object-Oriented Software,* Eddison-Wesley Pubs, 1995.
3. L. Bass, P. Clements and R. Kazman, *Software Architecture in Practice*, Addison Wesley, 1998 (ISBN: 0-201-19930-0).
4. M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996 (ISBN: 0-13-182957-2).
5. J. Bosch, *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*, Addison Wesley, 2000 (ISBN: 0-201-67494-7).
6. D. E. Perry and A. L. Wolf. *Foundations for the Study of Software Architectures. ACM SIGSOFT Software Engineering Notes*, October 1992.
7. P. Kruchten, *Architectural Blueprints – The "4+1" View Model of Software Architecture*. *IEEE Software,* 12 (6), November 1995, pp. 42-50.
8. M. Moriconi et al. *Correct Architecture Refinement*. *IEEE Transactions on Software Engineering*, April 1995.
9. C. Gacek and B. W. Boehm. *Composing Components: How Does One Detect Potential Architectural Mismatches*? *Workshop on Compositional Software Architectures,* Monterey, CA, January 1998. (html)
10. D. Garlan, R. Allen, and J. Ockerbloom, *Architectural Mismatch or Why it's hard to build systems out of existing parts*, *IEEE Software*, November 1995, pp. 17-26.
11. R. Natarajan and D. S. Rosenblum. *Merging Component Models and Architectural Styles*. *Third International Software Architecture Workshop,* November 1998.
12. R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson and J. Carriere, *The Architecture Tradeoff Analysis Method,* in Proceedings of ICECCS, August 1998.
13. D. Soni,R.L. Nord and C. Hofmeister, *Software Architecture in Industrial Applications, Proceedings of ICSE 17.* New York: ACM Press 1995: 196-207.
14. L. Dobrica and E. Niemela, *A Survey on Software Architecture Analysis Methods.* IEEE Transactions on Software Engineering, Vol. 28, No. 7, July 2002, pp. 638-653.
15. Thomas Erl, Service Oriented Architecture: Concept, Technology and Design, Prentice Hall, 2005.
16. Thomas Erl, SOA – Principles of Service Design, Prentice Hall, 2008.