SOFTWARE DEVELOPMENT METHODOLOGY

VII Semester: CSE

Course Code	Category	Hours / Week			Credits	Maximum Marks		
		L	T	P	C	CIA	SEE	Total
AIT508	Elective	3	-	-	4	30	70	100
Contact Classes: 45	Tutorial Classes: 0	Practical Classes: Nil				Total Classes: 60		

COURSE OBJECTIVES:

The course should enable the students to:

I. Learn how to elicitate requirements and develop software life cycles.
II. Understand the design considerations for enterprise integration and deployment.
III. Analyze quality assurance techniques and testing methodologies.
IV. Understand implementation issues such as modularity and coding standards.
V. Prepare a project plan for a software project that includes estimates of size and effort, a schedule, resource allocation, configuration control, and project risk.

COURSE OUTCOMES (COs):

CO 1: Identify the approach to risks management through risk identification, risk measurement and risk mitigation.
CO 2: Use the concept of classical analysis to determine the acceptance criteria part of specification
CO 3: Understand the principles of graphical user interface design.
CO 4: Identify the major differences between white box testing and black box testing.
CO 5: Identify the importance of earned value analysis related to project scheduling and also understand the Various process and project metric used to improve the quality of software.

COURSE LEARNING OUTCOMES (CLOs):

1. Understand the key concerns that are common to all software development processes.
2. Identify the appropriate process models, approaches and techniques to manage a given software development process.
3. Identify the approach to risks management through risk identification, risk measurement and risk mitigation.
4. Use the concept of Earned Value Analysis (EVA) to measure the projects progress at any given point in time, forecasting its completion date and final cost, and analyzing variances in the schedule and budget as the project proceeds.
5. Memorize project planning activities that accurately help in selection and initiation of individual projects and of portfolios of projects in the enterprise.
6. Identify dependability and security issues that affect a given software product.
7. Use the concept of classical analysis to determine the acceptance criteria as part of specification.
8. Memorize the importance of eliciting the requirements for a software product and translate these into a documented design.
9. Understand the concept of data dictionary in order to manage the details in large-scale systems, to locate errors and omissions in the system.
10. Understand the concept of petri nets that exhibit concurrency, synchronization and used as a visual communication aid to model the system behavior.
11. Memorize the design of object oriented software using with the aid of a formal system modelling notation.
12. Learn to model the structure and behavior of a software system.
13. Memorize different architectural styles, patterns and architectural mapping using data flow.
14. Understand the principles of graphical user interface design.

15. Understand the concept of component-level design used to define interface characteristics and communication mechanisms for each software component identified in the architectural design.
16. Understand the importance of testing with the performance of root cause analysis.
17. Memorize the concepts of software testing approaches such as unit testing and integration testing.
18. Understand the approaches to verification and validation including static analysis and reviews.
19. Identify the major differences between white box testing and black box testing.
20. Understand the importance of refactoring which improves the performance of non-functional attributes of the software.
21. Learn to manage time, processes and resources effectively by prioritizing competing demands to achieve personal and team goals.
22. Understand the concept of risk management through risk identification, risk measurement and mitigation.
23. Memorize the relationship between people and effort.
24. Identify the importance of earned value analysis related to project scheduling.
25. Use a proactive, structured risk assessment and analysis activity to identify and analyze root causes.
26. Possess the knowledge and skills for employability and to succeed in national and international level competitive exams.

UNIT-I	**INTRODUCTION, A GENERIC VIEW OF PROCESS AND PROCESS MODELS**	**Classes: 08**

Introduction to software engineering: The evolving role of software, changing nature of software, legacy software, software myths; A generic view of process: Software engineering , a layered technology, a process framework, the capability maturity model integration (CMMI), process patterns, process assessment, personal and team process models, process models: the waterfall model, incremental process models, the unified process.

UNIT-II	**SOFTWARE REQUIREMENTS AND REQUIREMENTS ENGINEERING PROCESS**	**Classes: 09**

Design engineering: Design process and design quality, design concepts, the design model, pattern based software design.

Creating an architectural design: Software architecture, data design, architectural styles and patterns, architectural design, assessing alternative architectural designs, mapping data flow into software architecture.

UNIT-III	**DESIGN ENGINEERING, CREATING AN ARCHITECTURAL DESIGN AND MODELING COMPONENT-LEVEL DESIGN**	**Classes: 09**

Design process: Design concepts, design mode, design heuristic, architectural design architectural styles, architectural design, and architectural mapping using data flow.

User interface design: Interface analysis, interface design; Component level design: Designing class based components, traditional components.

UNIT-IV	**TESTING STRATEGIES AND PRODUCT METRIC**	**Classes: 10**

Testing strategies: A strategic approach to software testing, test strategies for conventional software, black-box and white-box testing, validation testing, system testing, the art of debugging; Product metrics: Software quality, frame work for product metrics, metrics for analysis model, metrics for design model, metrics for source code, metrics for testing, metrics for maintenance..

UNIT-V	**RISK MANAGEMENT AND QUALITY MANAGEMENT**	**Classes: 09**

Risk management: Reactive vs proactive risk strategies, software risks, risk identification, risk projection, risk refinement, RMMM(Risk Mitigation, Monitoring and Management), RMMM plan; Quality Management: Quality concepts, software quality assurance, software Reviews, formal technical reviews, statistical software quality assurance, software reliability, The ISO 9000 quality standards.

Text Books:

1. Roger S. Pressman, –Software Engineering – A Practitioner‘s Approach, Tata McGraw-Hill International Edition, 7th Edition, 2010.
2. Ian Somerville, –Software Engineering, Pearson Education Asia, 9th Edition, 2011.

Reference Books:

1. Rajib Mall, –Fundamentals of Software Engineering‖, PHI Learning Private Limited, 3rd Edition, 2009.
2. Pankaj Jalote, –Software Engineering, A Precise Approach‖, Wiley India, 1st Edition, 2010.

Web References:

1. http://www.softwareengineerinsider.com/articles/what-is-software-engineering.html
2. https://www.udacity.com/courses/software-engineering
3. http://www.tutorialspoint.com/software_engineering
4. http://computingcareers.acm.org/?page_id=12

http://en.wikibooks.org/wiki/Introduction_to_Software_Engineering

E-Text Books:

1. http://www.acadmix.com/eBooks_Download
2. http://www.freetechbooks.com/software-engineering-f15.html

Course Home Page: