SOFTWARE ENGINEERING

IV Semester: **IT**

Course Code	Category	Hours / Week			Credits	Maximum Marks		
		L	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
ACS008	**Core**	3	1	-	4	30	70	100
Contact Classes: 45	**Tutorial Classes: 15**	**Practical Classes: Nil**				**Total Classes: 60**		

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. 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 LEARNING OUTCOMES:

Students, who complete the course, will have demonstrated the ability to do the following:

I. Understand the key concerns that are common to all software development processes.
II. Identify the appropriate process models, approaches and techniques to manage a given software development process.
III. Identify the approach to risks management through risk identification, risk measurement and risk mitigation.
IV. 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.
V. Memorize project planning activities that accurately help in selection and initiation of individual projects and of portfolios of projects in the enterprise.
VI. Identify dependability and security issues that affect a given software product.
VII. Use the concept of classical analysis to determine the acceptance criteria as part of specification.
VIII. Memorize the importance of eliciting the requirements for a software product and translate these into a documented design.
IX. Understand the concept of data dictionary in order to manage the details in large-scale systems, to locate errors and omissions in the system.
X. Understand the concept of petri nets that exhibit concurrency, synchronization and used as a visual communication aid to model the system behavior.
XI. Memorize the design of object oriented software using with the aid of a formal system modelling notation.
XII. Learn to model the structure and behavior of a software system.
XIII. Memorize different architectural styles, patterns and architectural mapping using data flow.
XIV. Understand the principles of graphical user interface design.
XV. Understand the concept of component-level design used to define interface characteristics and communication mechanisms for each software component identified in the architectural design.
XVI. Understand the importance of testing with the performance of root cause analysis.

XVII. Memorize the concepts of software testing approaches such as unit testing and integration testing.
XVIII. Understand the approaches to verification and validation including static analysis and reviews.
XIX. Identify the major differences between white box testing and black box testing.
XX. Understand the importance of refactoring which improves the performance of non functional attributes of the software.
XXI. Learn to manage time, processes and resources effectively by prioritizing competing demands to achieve personal and team goals.
XXII. Use a proactive, structured risk assessment and analysis activity to identify and analyze root causes.
XXIII. Understand the concept of risk management through risk identification, risk measurement and mitigation.
XXIV. Memorize the relationship between people and effort.
XXV. 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.
XXVI. Possess the knowledge and skills for employability and to succeed in national and international level competitive exams.

UNIT-I	**SOFTWARE PROCESS AND PROJECT MANAGEMENT**	**Classes: 08**
Introduction to software engineering, software process, perspective and specialized process models; Software project management: Estimation: LOC and FP based estimation, COCOMO model; Project scheduling: Scheduling, earned value analysis, risk management		
UNIT-II	**REQUIREMENTS ANALYSIS AND SPECIFICATION**	**Classes: 09**
Software requirements: Functional and nonfunctional, user requirements, system requirements, software requirements document; Requirement engineering process: Feasibility studies, requirements elicitation and analysis, requirements validation, requirements management; Classical analysis: Structured system analysis, petri nets, data dictionary.		
UNIT-III	**SOFTWARE 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 AND IMPLEMENTATION**	**Classes: 10**
Software testing fundamentals: Internal and external views of testing, white box testing, basis path testing, control structure testing, black box testing, regression testing, unit testing, integration testing, validation testing, system testing and debugging; Software implementation techniques: Coding practices, refactoring.		
UNIT-V	**PROJECT MANAGEMENT**	**Classes: 09**
Estimation: FP based, LOC based, make/buy decision; COCOMO II: Planning, project plan, planning process, RFP risk management, identification, projection; RMMM: Scheduling and tracking, relationship between people and effort, task set and network, scheduling; EVA: Process and project metrics.		

Text Books:

1. Roger S. Pressman, “Software Engineering – A Practitioner’s Approach”, 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. PankajJalote, “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
5. 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