SOFTWARE ENGINEERING

COURSE CODE: 23IT4113 CREDITS 3

L T P S Sessional Marks: 40

3 0 0 0 End Exam: 3 Hours

End Exam Marks: 60

Prerequisites: Computer Fundamentals, Any Programming Language

Course Objectives:

- The aim of the course is to provide an understanding of the working knowledge of the techniques for estimation, design, testing and quality management of large software development projects.
- Topics include process models, software requirements, software design, software testing, software process/product metrics, risk management, quality management and Agile Development.

Course Outcomes:

After course completion, the students will be able to:

1. Translate end-user requirements into system and software requirements, using UML, and structure the requirements into a Software Requirements Document (SRD).
2. Identify and apply appropriate software architectures and patterns to carry out high level design of a system and be able to critically compare alternative choices.
3. Develop a simple testing report, and to manage time, processes and resources effectively by prioritising competing demands toachieve personal and team goals
4. Demonstrate Agile Development & Testing Techniques

Mapping of Course Outcomes with POs and PSOs

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
CO1	3	2	3	3	1	1	1				2	2	3	3
CO2	2	2	2	2	1					2	2	2	3	3
CO3	2	2	3	3	3						1	1	3	3
CO4	2	2	3	3	3						1	1	3	3

Correlation levels 1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High)

UNIT – I 10 Periods

Introduction to Software Engineering: The evolving role of software, change in nature of software, software myths

Software Requirements: Functional and non-functional requirements, user requirements, system requirements, interface specification, the software requirements document.

Process Models: Waterfall model, Incremental model, Evolutionary models, Unified models

Learning outcomes:

1. Process of analysing user requirements

2. designing software application which will satisfy that requirements

UNIT – II 10 Periods

Requirements engineering process: Feasibility studies, requirements elicitation and analysis,requirements validation, requirements management.

System models: Context models, behavioral models, object models, structured methods.

UNIT – III 10 Periods

Design Engineering: Design process and design quality, design concepts, the design model. **Creating an Architectural Design:** Software architecture, data design, architectural styles andpatterns, architectural design.

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

Learning outcomes:

1. To produce efficient, reliable, robust and cost-effective software solutions.
2. Ability to perform independent research and analysis.

UNIT – IV 9 Periods

Metrics for Process and Products: Software measurement, metrics for software quality, Metrics for Process and Product

Quality Management: Quality concepts, software quality assurance, software reviews, formal technical reviews, statistical software quality assurance, software reliability, the ISO standards

Learning outcomes:

1. check whether the actual software product matches expected requirements
2. making it efficient and effective as per the quality standards defined for software products
3. Ability to understand and meet ethical standards and legal responsibilities
4. Ability to develop, maintain and evaluate large-scale software systems

UNIT-V 9 Periods

Agile Methodology: Theories for Agile management – agile software development – traditional model vs. agile model - classification of agile methods – agile manifesto and principles – agile project management – agile team interactions – ethics in agile teams - agility in design, testing – agile documentations – agile drivers, capabilities and values.

Learning outcomes:

1. Importance of interacting with business stakeholders in determining the requirements for a software system

TEXT BOOKS:

1. Software Engineering, A practitioner’s Approach- Roger S. Pressman, 6th edition, Mc Graw Hill International Edition.
2. The unified modeling language user guide Grady Booch, James Rambaugh, Ivar Jacobson, Pearson Education.

REFERENCE BOOKS:

1. Software Engineering, an Engineering approach- James F. Peters, Witold Pedrycz, John Wiley.
2. Software Engineering principles and practice- Waman S Jawadekar, The Mc Graw-Hill Companies.
3. Fundamentals of object-oriented design using UML Meiler page-Jones: Pearson Education.
4. Hazza& Dubinsky, ―Agile Software Engineering, Series: Undergraduate Topics in Computer Science‖, Springer, VIII edition, 2009
5. Craig Larman, ―Agile and Iterative Development: A manager_s Guide, Addison-Wesley, 2004
6. Software Engineering- Sommerville, 7th edition, Pearson Education.
7. Dingsoyr, Torgeir, Dyba, Tore, Moe, Nils Brede (Eds.), ―Agile Software Development, Current Research and Future Directions‖, Springer-Verlag Berlin Heidelberg, 2010
8. David J. Anderson; Eli Schragenheim, ―Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results‖, Prentice Hall, 2003

CHANGE OF SYLLABUS

Unit No	**Changes Incorporated**
Unit – 5	**Agile Methodology**
Change of Syllabus: 20%	

1.1.3 of NAAC

Name of the Course	**Course Code**	**Year of Introduction**	**Activities/Content with a direct bearing on Employability/ Entrepreneurship/ Skill development**	**Mapping with Employability/Skill development/Entrepreneurship**
Software Engineering	23IT4113	R23 (2024)	Agile Methodology Product metrics	Employability