|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 13CS2204 | - | SOFTWARE ENGINEERING | | | | | | | | |
|  |  | |  | | | | |
| Hours / Week | : | 4 | |  | Sessional Marks | : | 40 |
| Credits | : | 4 | |  | End Examination Marks | : | 60 |

|  |
| --- |
| **UNIT – I** |
| **Introduction to Software Engineering**: Software evolution, Legacy software, Software myths.  **A Generic View of Process**: Software engineering layers, Process frame work, Capability Maturity Model Integration (CMMI), Process patterns, and assessment, Personal software process (PSP), Team software process (TSP) models.  **Process Models:** Prescriptive models, Waterfall model, Incremental model, RAD model, Spiral model, Concurrent development model, Formal methods model, Unified process. |
|  |
| **UNIT – II** |
| **Software Engineering Practice**: Principles, Communication practices, Planning practices, Analysis modeling principles, Design modeling principles, Coding principle and practice, Testing principles, Deployment.  **Requirements Engineering**: Requirements engineering tasks, Initiation, Eliciting requirements, developing use-cases, Building the analysis model, Negotiating and validating requirements. |
|  |
| **UNIT – III** |
| **Building the Analysis Model**: Requirements analysis, Analysis modeling approaches, Data modeling concepts, Object-oriented analysis, Scenario-based modeling, Flow-oriented modeling, Class-based modeling, Creating a behavioral model.  **Design Engineering**: Design process, Design quality, Design concepts, Design model, Pattern-based software design. |
|  |
| **UNIT – IV** |
| **Creating an Architectural Design**: Software architecture, Data design, Architectural styles and patterns, Architectural design, Assessing alternative architectural designs, Mapping data flow into a software architecture.  **Modeling Component-level Design**: Nature of component, Designing class-based components, Conducting component level design, Object constraint language, Designing conventional components. |
|  |
| **UNIT – V** |
| **Testing strategies**: A strategic approach to software testing, Test strategies for conventional software, Test strategies for object-oriented software, Validation testing, System testing, Art of debugging.  **Testing Tactics**: Software testing fundamentals, Black-box and white-box testing, Basis path testing, Control structure testing, Object-oriented testing methods, Class level testing methods, Testing patterns. |
|  |
|  |
| TEXT BOOKS |
| 1. Pressman R S, Software Engineering-A Practitioner’s Approach, 6th edition, McGraw-Hill, 2005. |
|  |
| REFERENCE BOOKS |
| 1. Sommerville I, Software Engineering, 5th edition, Pearson Education, 1996. 2. Jawadekar W S, Software Engineering – Principles and Practice, Tata McGraw-Hill, 2004. 3. Behforooz A, and Hudson F J, Software Engineering Fundamentals, Oxford University Press, 1996 |