| SOFTWARE ENGINEERING | | | |
|---|---|---|---|
| **(Effective from the academic year 2018 -2019)** | | | |
| **SEMESTER – III** | | | |
| **Course Code** | **18CS35** | **CIE Marks** | 40 |
| **Number of Contact Hours/Week** | 3:0:0 | **SEE Marks** | 60 |
| **Total Number of Contact Hours** | 40 | **Exam Hours** | 03 |
| **CREDITS –3** | | | |

**Course Learning Objectives:** This course (18CS35) will enable students to:

- Outline software engineering principles and activities involved in building large software programs.Identify ethical and professional issues and explain why they are of concern to software engineers.
- Explain the fundamentals of object oriented concepts
- Describe the process of requirements gathering, requirements classification, requirements specification and requirements validation. Differentiate system models, use UML diagrams and apply design patterns.
- Discuss the distinctions between validation testing and defect testing.
- Recognize the importance of software maintenance and describe the intricacies involved in software evolution.Apply estimation techniques, schedule project activities and compute pricing.
- Identify software quality parameters and quantify software using measurements and metrics. List software quality standards and outline the practices involved.

| Module 1 | Contact Hours |
|---|---|
| **Introduction**: Software Crisis, Need for Software Engineering. Professional Software Development, Software Engineering Ethics. Case Studies.<br>**Software Processes**: Models: Waterfall Model **(Sec 2.1.1)**, Incremental Model **(Sec 2.1.2)** and Spiral Model **(Sec 2.1.3)**. Process activities.<br>**Requirements Engineering**: Requirements Engineering Processes **(Chap 4)**. Requirements Elicitation and Analysis **(Sec 4.5).** Functional and non-functional requirements **(Sec 4.1)**. The software Requirements Document **(Sec 4.2)**. Requirements Specification **(Sec 4.3)**. Requirements validation **(Sec 4.6)**. Requirements Management **(Sec 4.7)**.<br>**RBT: L1, L2, L3** | 08 |
| **Module 2** | |
| What is Object orientation? What is OO development? OO Themes; Evidence for usefulness of OO development; OO modelling history. Modelling as Design technique: Modelling; abstraction; The Three models. **Introduction, Modelling Concepts and Class Modelling:** What is Object orientation? What is OO development? OO Themes; Evidence for usefulness of OO development; OO modelling history. Modelling as Design technique: Modelling; abstraction; The Three models. Class Modelling: Object and Class Concept; Link and associations concepts; Generalization and Inheritance; A sample class model; Navigation of class models;<br>**Textbook 2: Ch 1,2,3.**<br>**RBT: L1, L2 L3** | 08 |
| **Module 3** | |
| **System Models**: Context models **(Sec 5.1)**. Interaction models **(Sec 5.2)**. Structural models **(Sec 5.3)**. Behavioral models **(Sec 5.4)**. Model-driven engineering **(Sec 5.5)**.<br>**Design and Implementation**: Introduction to RUP **(Sec 2.4)**, Design Principles **(Chap 7)**. Object-oriented design using the UML **(Sec 7.1)**. Design patterns **(Sec 7.2)**. Implementation issues **(Sec 7.3)**. Open source development **(Sec 7.4)**.<br>**RBT: L1, L2, L3** | 08 |

| | |
|---|---|
| | |
| **Module 4** | |
| **Software Testing**: Development testing **(Sec 8.1)**, Test-driven development **(Sec 8.2)**, Release testing **(Sec 8.3)**, User testing **(Sec 8.4)**. Test Automation **(Page no 212)**.<br>**Software Evolution**: Evolution processes **(Sec 9.1)**. Program evolution dynamics **(Sec 9.2)**. Software maintenance **(Sec 9.3)**. Legacy system management **(Sec 9.4)**.<br>**RBT: L1, L2, L3** | 08 |
| **Module 5** | |
| **Project Planning**: Software pricing **(Sec 23.1)**. Plan-driven development **(Sec 23.2)**. Project scheduling **(Sec 23.3)**: Estimation techniques **(Sec 23.5)**. **Quality management**: Software quality **(Sec 24.1)**. Reviews and inspections **(Sec 24.3)**. Software measurement and metrics **(Sec 24.4)**. Software standards **(Sec 24.2)**<br>**RBT: L1, L2, L3** | 08 |

**Course Outcomes:** The student will be able to :

- Design a software system, component, or process to meet desired needs within realistic constraints.
- Assess professional and ethical responsibility
- Function on multi-disciplinary teams
- Use the techniques, skills, and modern engineering tools necessary for engineering practice
- Analyze, design, implement, verify, validate, implement, apply, and maintain software systems or parts of software systems

**Question Paper Pattern:**

- The question paper will have ten questions.
- Each full Question consisting of 20 marks
- There will be 2 full questions (with a maximum of four sub questions) from each module.
- Each full question will have sub questions covering all the topics under a module.
- The students will have to answer 5 full questions, selecting one full question from each module.

**Textbooks:**

1. Ian Sommerville: Software Engineering, 9th Edition, Pearson Education, 2012. (Listed topics only from Chapters 1,2,3,4, 5, 7, 8, 9, 23, and 24)
2. Michael Blaha, James Rumbaugh: Object Oriented Modelling and Design with UML,2nd Edition, Pearson Education,2005.

**Reference Books:**

1. Roger S. Pressman: Software Engineering-A Practitioners approach, 7th Edition, Tata McGraw Hill.
2. Pankaj Jalote: An Integrated Approach to Software Engineering, Wiley India

<table>
<tr><td colspan="5" align="center"><strong>DESIGN AND ANALYSIS OF ALGORITHMS</strong><br><strong>(Effective from the academic year 2018 -2019)</strong><br><strong>SEMESTER – IV</strong></td></tr>
<tr><td><strong>Course Code</strong></td><td><strong>18CS42</strong></td><td><strong>CIE Marks</strong></td><td>40</td></tr>
<tr><td><strong>Number of Contact Hours/Week</strong></td><td>3:2:0</td><td><strong>SEE Marks</strong></td><td>60</td></tr>
<tr><td><strong>Total Number of Contact Hours</strong></td><td>50</td><td><strong>Exam Hours</strong></td><td>03</td></tr>
<tr><td colspan="4" align="center"><strong>CREDITS –4</strong></td></tr>
</table>

**Course Learning Objectives:** This course (18CS42) will enable students to:

- Explain various computational problem solving techniques.
- Apply appropriate method to solve a given problem.
- Describe various methods of algorithm analysis.

| Module 1 | Contact Hours |
|---|---|
| **Introduction:** What is an Algorithm? **(T2:1.1),** Algorithm Specification **(T2:1.2),** Analysis Framework **(T1:2.1), Performance Analysis**: Space complexity, Time complexity **(T2:1.3)**. **Asymptotic Notations:** Big-Oh notation ($O$), Omega notation ($\Omega$), Theta notation ($\Theta$), and Little-oh notation ($o$), Mathematical analysis of Non-Recursive and recursive Algorithms with Examples **(T1:2.2, 2.3, 2.4). Important Problem Types:** Sorting, Searching, String processing, Graph Problems, Combinatorial Problems. **Fundamental Data Structures:** Stacks, Queues, Graphs, Trees, Sets and Dictionaries. **(T1:1.3,1.4).** <br><br>**RBT: L1, L2, L3** | 10 |
| **Module 2** | |
| **Divide and Conquer**: General method, Binary search, Recurrence equation for divide and conquer, Finding the maximum and minimum **(T2:3.1, 3.3, 3.4)**, Merge sort, Quick sort **(T1:4.1, 4.2)**, Strassen's matrix multiplication **(T2:3.8)**, Advantages and Disadvantages of divide and conquer. **Decrease and Conquer Approach:** Topological Sort. **(T1:5.3).** <br><br>**RBT: L1, L2, L3** | 10 |
| **Module 3** | |
| **Greedy Method:** General method, Coin Change Problem, Knapsack Problem, Job sequencing with deadlines **(T2:4.1, 4.3, 4.5). Minimum cost spanning trees:** Prim's Algorithm, Kruskal's Algorithm **(T1:9.1, 9.2)**. **Single source shortest paths:** Dijkstra's Algorithm **(T1:9.3)**. **Optimal Tree problem:** Huffman Trees and Codes **(T1:9.4)**. **Transform and Conquer Approach:** Heaps and Heap Sort **(T1:6.4)**. <br><br>**RBT: L1, L2, L3** | 10 |
| **Module 4** | |
| **Dynamic Programming:** General method with Examples, Multistage Graphs **(T2:5.1, 5.2)**. **Transitive Closure:** Warshall's Algorithm, **All Pairs Shortest Paths:** Floyd's Algorithm, Optimal Binary Search Trees, Knapsack problem ((**T1:8.2, 8.3, 8.4**), Bellman-Ford Algorithm **(T2:5.4)**, Travelling Sales Person problem **(T2:5.9)**, Reliability design **(T2:5.8)**. <br><br>**RBT: L1, L2, L3** | 10 |
| **Module 5** | |
| **Backtracking:** General method **(T2:7.1)**, N-Queens problem **(T1:12.1)**, Sum of subsets problem **(T1:12.1)**, Graph coloring **(T2:7.4)**, Hamiltonian cycles **(T2:7.5)**. **Programme and Bound:** Assignment Problem, Travelling Sales Person problem **(T1:12.2)**, **0/1 Knapsack problem (T2:8.2, T1:12.2):** LC Programme and Bound solution **(T2:8.2)**, FIFO Programme and Bound solution **(T2:8.2)**. **NP-Complete and NP-Hard problems:** Basic concepts, non- | 10 |

| |
|---|
| deterministic algorithms, P, NP, NP-Complete, and NP-Hard classes **(T2:11.1)**. |
| **RBT: L1, L2, L3** |

**Course Outcomes:** The student will be able to :

- Describe computational solution to well known problems like searching, sorting etc.
- Estimate the computational complexity of different algorithms.
- Devise an algorithm using appropriate design strategies for problem solving.

**Question Paper Pattern:**

- The question paper will have ten questions.
- Each full Question consisting of 20 marks
- There will be 2 full questions (with a maximum of four sub questions) from each module.
- Each full question will have sub questions covering all the topics under a module.
- The students will have to answer 5 full questions, selecting one full question from each module.

**Textbooks:**

1. Introduction to the Design and Analysis of Algorithms, Anany Levitin:, 2rd Edition, 2009. Pearson.
2. Computer Algorithms/C++, Ellis Horowitz, Satraj Sahni and Rajasekaran, 2nd Edition, 2014, Universities Press

**Reference Books:**

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronal L. Rivest, Clifford Stein, 3rd Edition, PHI.
2. Design and Analysis of Algorithms , S. Sridhar, Oxford (Higher Education).