

## BCA 401: Software Engineering

### Teaching Scheme

Lectures: 4 hrs/Week

Tutorials: 2 hr/Week

Credits: 6

### Examination Scheme

Class Test -20Marks

Teachers Assessment - 10Marks

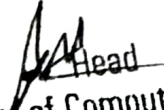
Attendance – 20 Marks


End Semester Exam – 100 marks

**Prerequisite:** - BCA 103 Computer Fundamental and Programming using C.

### Course Objectives:

1. To recognize basic software design principles, software engineering methods and practices, software cost estimation, testing approaches and their appropriate application.
2. To exemplify the critical understanding of software process models, project management and requirements, implementation issues, verification and validation.
3. To implement techniques, skills, and modern software engineering tools for designing a system and to apply the basic project management practices in real life projects.
4. To demonstrate development of a computing-based system in terms of design, verification, validation, implementation, and maintenance within realistic constraints.
5. To evaluate software design principles, software requirements with existing tools and to test the project with respect to effort and development time.

  
Head  
Department of Computer Applications  
Faculty of Computer Applications  
Lovely Professional University, Bareilly (UP)  
Bachelor of Computer Applications

  
Registrar  
Lovely Professional University  
Bareilly

  
Dean Academics  
Faculty of Computer Applications  
Lovely Professional University, Bareilly (UP)

## Detailed Syllabus

### Unit-1

**Introduction:** Introduction to Software Engineering, Software Characteristics, Software Engineering Processes, And Software Development Life Cycle (SDLC) Models: Water Fall Model, Prototype Model, Spiral Model, Evolutionary Development Models, and Iterative Enhancement Models.

### Unit-2

**Software Requirement Specifications (SRS):** Requirement Engineering Process: Elicitation, Analysis, Documentation, Review and Management of User Needs, Feasibility Study, Data Flow Diagrams, Entity Relationship Diagrams, Decision Tables, SRS Document.

### Unit-3

**Software Design:** Basic Concept of Software Design, Architectural Design, Low Level Design: Modularization, Design Structure Charts, Coupling and Cohesion, Top-Down and Bottom-Up Design Strategies: Function Oriented Design, Object Oriented Design.

### Unit-4

**Software Testing :** Testing Objectives, Test Data Suit Preparation, Unit Testing, Integration Testing, Acceptance Testing, Regression Testing, Top-Down and Bottom-Up testing. White Box Testing, Black Box Testing, Alpha and Beta Testing of Products. Formal Technical Reviews, Walk Through, Code Inspection, Compliance with Design and Coding Standards.

### Unit-5

**Software Maintenance:** Need for Maintenance, Preventive, Corrective and Perfective Maintenance Cost of Maintenance, Maintenance Models.

### Unit-6

**Software Project Management:** Estimation of Various Parameters such as Size, Cost, Efforts, Schedule/Duration, Constructive Cost Model (COCOMO), Resource Allocation Models, Software Risk Analysis and Management, Software Quality Attributes and Factors Software Configuration Management, CASE Tools.

### Text and Reference Books

1. Software Engineering: A Practitioners Approach, R. S. Pressman, McGraw Hill, 6<sup>th</sup> Edition.
2. Fundamentals of Software Engineering, Rajib Mall, PHI Publication, 2<sup>nd</sup> Edition.
3. K. K. Aggarwal and Yogesh Singh, Software Engineering, New Age International Publishers, 3<sup>rd</sup> Edition.
4. Software Engineering, Pankaj Jalote, Wiley, 5<sup>th</sup> Edition.
5. Ian Sommerville, Software Engineering, Addison Wesley, 7<sup>th</sup> Edition.

### Course Outcomes:

After completing the course, students will be able to:

1. Understand that how to apply the software engineering lifecycle by demonstrating competence in planning, analysis, design, testing and implementation.
2. Identify the best software model to develop a real-life software product.
3. Demonstrate an ability to use the techniques and tools necessary for engineering practice.
4. Work in one or more significant application domains.
5. Demonstrate an understanding of and apply current theories, models, and techniques that provide a basis for the software lifecycle.
6. Work as an individual and as part of a multidisciplinary team to develop and deliver quality software.