PARVATHANENI BRAHMAYYA
SIDDHARTHA COLLEGE OF ARTS & SCIENCE
Autonomous
Siddhartha Nagar, Vijayawada–520010
Re-accredited at 'A+' by the NAAC

Course Code				**23CSMAL234**			
Title of the Course				**SOFTWARE ENGINEERING**			
Offered to: (Programme/s)				**B.C.A Hons**			
L	**4**	**T**	**0**	**P**	**0**	**C**	**4**
Year of Introduction:	**2024-25**			**Semester:**			**3**
Course Category:	**Major**			**Course Relates to:**		Local, Regional, National, Global	
Year of Revision:				**Percentage:**			
Type of the Course:				Employability/ Skill development			
Crosscutting Issues of the Course :							
Pre-requisites, if any							

Legend:

Offered to: B.C.A Hons

Category: Major

Course Relates to: Local, Regional, National, Global

Type of the Course: Employability/ Skill development

Crosscutting Issues of the Course: Gender, Environment and Sustainability, Human Values and Professional Ethics

L: Lecture; T: Tutorial; P: Practicum/Practical/Project; C: Credits

Course Description:

The course is to assist the student in understanding the basic theory of software engineering, and to apply these basic theoretical principles to a group software development project.

Program Design Tools:

1. To draw dataflow diagrams using Microsoft Visio Software, SmartDraw,
2. To draw UML diagrams using Rational Rose Software, Star UML, etc.

Course Aims and Objectives:

S. NO	COURSE OBJECTIVES
1	Grasp fundamental software engineering concepts, methodologies, and principles
2	Known about ethical responsibilities of software engineers.
3	Gain the ability to design software systems that are modular, scalable, and maintainable.
4	Study the cognitive, physical, and social aspects of human interaction with technology.
5	Learn techniques for software testing and quality assurance and theoretical knowledge to real-world scenarios through case studies and practical exercises..

Course Outcomes

At the end of the course, the student will be able to…

CO NO	COURSE OUTCOME	BTL	PO	PSO
CO1	Understand the requirements of the software projects.	K2	PO5,PO7	PSO2
CO2	Ability to analyze software requirements with existing tools	K4	PO5,PO7	PSO1PSO,2
CO3	Apply different testing methodologies	K3	PO5,PO7	PSO1,PSO2
CO4	Understand and apply the basic project management practices in real life projects	K2,K4	PO5,PO7	PSO1,PSO2
CO5	Apply on software projects	K4	PO5,PO7	PSO1,PSO2

For BTL: K1: Remember; K2: Understand; K3: Apply; K4: Analyze; K5: Evaluate; K6: Create

CO-PO MATRIX									
CO NO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PSO1**	**PSO2**
CO1					3		2		1

CO2					3		2	3	2
CO3					3		2	2	
CO4					3		2	2	2
CO5					3		2	3	2

Use the codes 3, 2, 1 for High, Moderate and Low correlation Between CO-PO-PSO respectively

Course Structure:

Unit – 1 : (10Hrs)

Introduction to Software Engineering: The Software Engineering – Evolution and impact, Software Development Projects, Software Process and Project Metrics, Emergence of Software Engineering, Computer Systems Engineering,

Software Life cycle models: Need for life Cycle model, classical waterfall model, Iterative waterfall model, V-model, Prototyping model, Evolutionary model, Spiral model, Agile Development Models, Comparison of different life cycle models.

Description:

Software Engineering is the systematic application of engineering principles to the development, operation, maintenance, and retirement of software. It's a discipline focused on producing high-quality software that meets user needs within budget and on time.

Learning Outcomes:

- Understanding software project management principles.
- Proficiency in software development tools and environments.
- Knowledge of software design patterns and architectures.

Examples:

Conduct workshops and training sessions to ensure all team members understand the principles and practices of software engineering. Create a repository of reference materials and best practices

Exercises

Analyze the size factors of a software project Building online stores, payment gateways

Specific Resources: (web)

Prof. Rajib Mall, Assistant Professor, Department of Computer Science and Engineering, IIT Kharagpur

https://youtu.be/Ln_LP7c23WM

UNIT – II (12Hrs)

Software Project Management: Responsibilities of a Software Project Manager, Project planning, Metrics for Project size estimation and scheduling.

Requirement Analysis: Requirements gathering and analysis, Software Requirements Specification ___ contents of the SRS document, Functional requirements, Traceability, Characteristics of good SRS DOCUMENT, Organization of the SRS document.

Description:

Software cost estimation is the process of predicting the financial resources required to develop and maintain a software system. Accurate cost estimation is crucial for project planning, budgeting, and resource allocation. Various techniques are employed to estimate software costs

Learning Outcomes:

- Understand the concept of software cost estimation
- Identify key factors
- Apply various cost estimation techniques
- Estimate staffing levels
- Determine the factors influencing software maintenance costs
- Create comprehensive software requirements specifications
- Utilize formal specification techniques
- Select appropriate languages and processors
- Evaluate the impact of software requirements on project costs

Examples:

- Detailed Explanation of the COCOMO Model

Exercises:

- The basic COCOMO formula is:

$Effort=a\times(KLOC)^b$

Where:

- **Effort** is the number of person-months.
- **KLOC** is the estimated number of thousands of lines of code.
- **a** and **b** are coefficients that vary depending on the project type (organic, semi-detached, or embedded).

For example, for an organic project: a=2.4, b=1.05a = 2.4, \, b = 1.05a=2.4,b=1.05

Suppose we estimate the size of the software to be 50 KLOC:

Specific Resources: (web)

Prof. Rajib Mall, Assistant Professor, Department of Computer Science and Engineering, IIT Kharagpur

https://youtu.be/ilHPCbkZLV4

UNIT – III (12Hrs)

Software design: Desirable characteristics of a good software design, Cohesion and coupling, Layer Arrangement of Modules, Function-oriented design and Object-oriented design.

Function-oriented software Design: Overview of SA/SD methodology, structured analysis, Data Flow Diagrams, Structured Design and Detailed Design.

Description:

Software design is the process of conceptualizing the software solution to a problem, transforming user requirements into a suitable form, and producing a design document based on the customer requirements. It's a crucial phase that bridges the gap between requirements analysis and software implementation.

Learning Outcomes:

- Define software design
- Transform user requirements
- Apply fundamental design concepts
- Utilize various design notations
- Employ different design techniques
- Conduct detailed design
- Design software for real-time and distributed systems
- Develop comprehensive test plans to ensure software quality.
- Participate in software design reviews

Examples:

- Software Design for an E-Commerce Platform

Exercises:

- Represent Design Notations for e-commeerce platform

UML Diagrams

- **Class Diagrams**: Represent classes and relationships in each module.
- **Sequence Diagrams**: Show interactions between objects during user registration, product search, order placement, and payment processing.
- **Data Flow Diagrams (DFDs)**

- Illustrate how data flows through the system, from user inputs to database storage and retrieval.

Specific Resources: (web)

Prof. Mythii Vutukuru, assistant Professor, IIT Bombay, software Design,

https://youtu.be/3fLahzQr8EI?list=PLDW872573QAZNlUzWVzoU8cCadXg1NUGK

UNIT IV (12Hrs)

User interface design: Characteristics of good user interface design, Basic concepts, Types of user interfaces, component-based GUI development, A user interface Design Methodology

Unified Modeling Language: Overview of Object-oriented concepts, Unified Modeling Language, UML diagrams, use case model class diagrams, Interaction diagrams, Activity diagrams, state chart diagrams

Description:

User Interface (UI) design is the process of creating effective interactions between humans and computer systems. It's about designing the look, feel, and behavior of software applications to ensure they are user-friendly, efficient, and enjoyable to use.

Learning Outcomes:

- Define user interface design.
- Apply human factors principles.
- Understand the fundamentals of human-computer interaction.
- Develop effective user interfaces.
- Create visually appealing and intuitive user interfaces.
- Evaluate user interface designs.
- Design user interfaces for specific user groups.
- Apply user interface design principles to real-time systems .
- Stay updated on emerging trends and technologies.

Examples:

- User Interface Design for a Real-Time Patient Monitoring System

Exercises:

Example Interface Design:

Dashboard:

- **Patient List**: A list of all monitored patients with summary information (name, room number, key vital signs).
- **Critical Alerts**: A section for critical alerts, sorted by severity and time.

- **Navigation**: Easy access to patient detail views, settings, and system logs.

Patient Detail View:

- **Vital Signs Graphs**: Real-time graphs showing trends for heart rate, blood pressure, temperature, etc.
- **Alerts History**: A log of all alerts for the patient, with timestamps and statuses.
- **Actions**: Buttons for common actions, such as acknowledging alerts, adding notes, or calling for assistance.

Settings and Customization:

- **Alert Thresholds**: Interface for setting and adjusting alert thresholds for different vital signs.
- **Display Options**: Options for customizing the layout, themes, and data visibility.

Specific Resources: (web)

Dr. Samit Bhattacharya, Assistant Professor, Computer Science and Engineering, IIT GUWAHATI, Design & Implementation of Human-Computer Interfaces

https://youtu.be/uFYuHHglC6U?list=PLwdnzlV3ogoVKbbd4bwgSoga7EEuX5kFf

UNIT V (14Hrs)

Software quality and testing: Software Quality Assurance - Quality metrics - Software Reliability - Software testing - Path testing – Control Structures testing - Black Box testing - Integration, Validation and system testing - Reverse Engineering and Reengineering.

CASE Tools: Projects management, tools - analysis and design tools – programming tools - integration and testing tool - Case studies.

Description:

software quality and testing are critical for producing reliable and efficient software. CASE tools can significantly enhance the software development process by automating tasks and improving productivity

Learning Outcomes:

- Define software quality assurance.
- Identify and apply quality metric.
- Explain the concept of software reliabilit.
- Design and execute various software testing techniques.
- Conduct integration, validation, and system testing.
- Apply reverse engineering and reengineering techniques to analyze and modify existing software systems.
- Utilize CASE tools to support software quality assurance and testing activities.

- Create and manage software test plans and test cases.
- Analyze test results and generate test reports to identify defects and recommend corrective actions.
- Understand the importance of software quality

Examples:

- Implementing Test Automation for a E-commerce Platform

Exercises:

- **Client:** A large online retailer looking to improve software quality and reduce testing time.
- **Project:** Implement a test automation framework for functional, regression, and performance testing of their e-commerce platform.

Specific Resources: (web)

Prof. Rajib Mall, Assistant Professor, Department of Computer Science and Engineering, IIT Kharagpur

https://youtu.be/ilHPCbkZLV4

Choose any two of above case studies and do the following exercises for that Case Study

1. Write the software requirements specification document

2. Draw the entity relationship diagram

3. Draw the data flow diagrams

4. Draw use case diagrams

5. Draw activity diagrams for all use cases

6. Draw sequence diagrams for all use cases

7. Draw collaboration diagram

8. Assign objects in sequence diagrams to classes and make class diagram.

Student Activity:

1. Visit any financial organization nearby and prepare requirement analysis report

2. Visit any industrial organization and prepare risk chart

Text Books:

1. Fundamentals of Software Engineering, Fourth Edition, Rajib Mall, PHI

References:

1. R.Fairley, Software Engineering Concepts, Tata McGraw-Hill, 1997.
2. Software Engineering, H. Sommervill Ian , Addition Wesley Pub. Co.
3. Software Engineering: An object Oriented Perspective by Braude, E.J., Willey, 2001

***