FF No. :654

Syllabus Template

CS2226:: Software Engineering

Course Prerequisites: Data Structures

Course Objectives:

1. To summarize capabilities and impact of Software Development Process Models and justify process maturity through application of Software Engineering principles and practices

2. To discriminate competing and feasible system requirements indicating correct real world problem scope and prepare stepwise system conceptual model using stakeholder analysis and requirement validation.

3. To formulate system specifications by analyzing User-level tasks and compose software artifacts using agile principles, practices and Scrum framework

4.To compose system analysis and design specifications indicating logical, physical, deployment, and concurrency viewpoints using object-oriented analysis and design principles and Model Driven Engineering practices using UML-supported modeling tools.

5. To comprehend the nature of design patterns by understanding a small number of examples from different pattern categories and apply these patterns in creating a correct design using design heuristics

6.To propose multi-faceted defendable solutions demonstrating team-skills accommodating design patterns reducing the potential cost and performance impedance in order to realize system artifacts with the help of Model Driven Development practices using, scheduling, estimation and risk management activities.

Credits:.5....... **Teaching Scheme Theory:…3 Hours/Week**

Tut: 1 Hours/Week

Lab:...2.Hours/Week

Course Relevance: Given that software engineering is built upon the foundations of both computer science and engineering, a software engineering curriculum can be approached from either a computer science-first or software engineering-first perspective; there clearly is merit in both approaches. Software engineering spans the entire software lifecycle - it involves creating high-quality, reliable programs in a systematic, controlled, and efficient manner using formal methods for specification, evaluation, analysis and design, implementation, testing and maintenance. any software products are among the most complex of man-made systems, requiring software development techniques and processes that successfully scale to large applications which satisfy timing, size, and security requirements all within acceptable timeframes and budgets. For these reasons, software engineering requires both the analytical and descriptive tools developed in computer science and the rigor that the engineering disciplines bring to the reliability and trustworthiness of the systems that software developers design and implement while working cohesively in a team environment.

SECTION-1
Topics and Contents **Software Engineering Paradigms**: Overview of Software Engineering, Software Process Framework, Traditional Process Models, Process Models: Code-and-Fix, Waterfall Model, Rapid Application Development, Incremental Models, Evolutionary Models, Iterative Development, The Unified Process, Cleanroom Methodology, Component-Based Software Engineering, CMMI, Software Engineering Principles and Practices, Requirements Engineering Tasks, Requirement Elicitation Techniques, Software Requirements: Functional, Non-Functional, Domain Engineering activities, Requirements Characteristics and Characterization, Eliminating Requirement Ambiguities, Conflict Identification and Resolution, Requirement Qualities, Requirement Specification, Requirement Traceability, Requirement Prioritization, Relationship of Requirement Engineering to other Framework Activities, System Scope Determination and Feasibility Study, Statement of Work Generation, Requirements Verification and Validation, Requirement Maturity, Technical Reviews, Stakeholder Management **Overview of Agile Methodology**: Introducing Agile in Practice, Landscape of Agile and Planned Methods, Agile Challenges in Practice, Composite Agile Method and Strategy (CAMS), Composite Agile and IT: Enablement, Development, and Maintenance, Collaborative-Agile Business Management, Business Analysis and Composite Agile, CAMS Project Management and ICT Governance, Agile Adoption in Organizations. Time-Boxing, Kanban, and Theory of Constraints, Lean IT, Pair Programming, Extreme Programming, DSDM, User Requirements in the context of Agile **The Scrum**: Scrum Origins: What Is Scrum? Scrum Origins, Why Scrum? Scrum Framework, Agile Principles, Overview, Variability and Uncertainty, Sprints., Requirements and User Stories, Product Backlog, Estimation and Velocity, Technical Debt, Roles: Product Owner, Scrum Master, Development Team, Scrum Team Structures, Managers, Planning: Scrum Planning Principles, Portfolio Planning, Envisioning (Product Planning), Release Planning (Longer-Term Planning), Sprinting: Sprint Planning, Sprint Execution, Sprint Review, Sprint Retrospective, Scrum and Service Industry

SECTION-1I

Topics and Contents

System Behavior Specification: Static Behavior: Use Cases, Use Case Diagram Components, Use Case Diagram, Actor Generalization, Include and Extend, Template for Use Case Narrative, Building Domain Model, and capturing system behavior in use cases, Use cases and User Stories, Dynamic Behavior: Sequence diagrams, object lifelines and message types, Modeling collections multiobjects, Refining sequence diagrams, Collaboration diagrams, States, events and actions, Nested machines and concurrency, Modifying the object model to facilitate states, Modeling methods with activity diagrams, Activity Diagrams: Decisions and Merges, Synchronization, Iteration, Partitions, Parameters and Pins, Expansion Regions, Swimlanes, concurrency and synchronization, Communication Diagram, Interaction Overview Diagrams, Timing Diagrams

Software Architecture Design and Configuration Management: Analysis Concepts, Analysis Methods, The Design Model, Design Qualities, Characteristics of Design activities, Design Principles, Cohesion and Coupling, Software Architecture Vs Software Design, Software Reuse, Design Heuristics, User Interface Design: Rules, User Interface Analysis and Steps in Interface Design, Design Evaluation, Source Code Management,

Foundations of Software Architecture, Reference Architectures, Architectural Design: Software Architecture, Data Design and Architectural Design, Views, Viewpoints, Perspectives, Conceptual Architecture View, Module Architecture View, Execution Architecture View, Code Architecture View.

Architecture styles: data-flow, object oriented, layered, data-centered, call and return, Repository, Pipe-Filter, Peer-Peer, Publish-Subscribe, Client-Server, Two-Tier, Three-Tier, N-Tier, Heterogeneity in Architecture, Categorizing classes: entity, boundary and control , Modeling associations and collections, Preserving referential integrity, Achieving reusability, Reuse through delegation, Identifying and using service packages, Improving reuse with design Packages and interfaces: Distinguishing between classes/interfaces, Exposing class and package interfaces.

Project Management Principles and Design Patterns:, Design Patterns: Introduction to Design Pattern, Describing Design Patterns, Catalogue of Design Patterns Creational Patterns: Abstract Factory, Builder, Factory Method, Prototype, Singleton, Structural Patterns: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy, Behavioral Patterns: Chain of

Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor, Antipatterns, Applications of Design Patterns, Project Management Activities, Structures and Frameworks, Teamwork, Leadership, Project Planning, Project Scheduling, Risk Analysis, Introduction to Function Points, Empirical Estimation, COCOMO II model.

List of Tutorials: (Any Four)

1. Study of Requirement Engineering
2. Study on preparation of System Requirement Specification
3. Scrum Artifacts
4. User Stories and Use Case
5. Product Backlog Development
6. Burn-up and Burn-down chart development and management
7. Software System Analysis and Design: UML
8. Incorporation of Design patterns

List of Practical's: (Any Eight)

1. A real-world problem issue is required to be identified with manageable scope. The problem scenarios are required to be identified for target system to be developed. The scenarios are stated in the form of Statement-of-Work template. The SOW document shall address the vision, goals, and objectives of the project.
2. The initial requirements and feature set for the target system is required to be identified. The requirements are required to be synthesized with stakeholder participation. The project roles are assigned to the project team with clear indicator of responsibilities. The initial requirements summary document with adequate and minimal infrastructure is required to be developed using multiple iterations.
3. The product backlog for the project aimed at maintaining a prioritized queue of project requirements shall be created.
    a. It should be dynamic and should be continuously groomed as the project progresses. Agile projects generally use an iceberg strategy for grooming the product backlog.
    b. The items that are near the top of the iceberg and are closest to going into development should get the most attention.
    c. There should typically be about two to three sprints worth of stories at the top of the backlog that are well-groomed and ready to go into development in order to avoid a situation where the project team is waiting for work to do.
4. Sprint-level planning activity accommodating story points, planning poker shall be performed. The Sprint-plan and Sprint-design indicating detailed activity planner shall be developed.
5. To decompose and organize the problem domain area into broad subject areas and identify the boundaries of problem/system. Specify the behavior of the target system and map requirements to Use cases.
    a. The System Context Diagram depicts the overall System behavioral trace and Requirement Capture diagram depicts the hierarchical Use case Organization. The Use Case diagram should encompass
    b. Actors (External Users)
    c. Transactions (Use Cases)
    d. Event responses related to transactions with external agents.
    e. Detection of System boundaries indicating scope of system.
6. To depict the dynamic behavior of the target system using sequence diagram. The Sequence diagram should be based on the Scenarios generated by the inter-object Communication. The model should depict:
    a. Discrete, distinguishable entities (class).
    b. Events (Individual stimulus from one object to another).
    c. Conditional events and relationship representation.

7. To depict the state transition with the life history of objects of a given class model. The model should depict:
    a. Possible ways the object can respond to events from other objects.
    b. Determine of start, end, and transition states.
8. To depict the dynamic behavior using detailed Activity diagram. Activity is a parameterized behavior represented as coordinated flow of actions. The flow of execution is modeled as activity nodes connected by activity edges.
    a. A node can be the execution of a subordinate behavior, such as an arithmetic computation, a call to an operation, or manipulation of object contents.
    b. Activities may form invocation hierarchies invoking other activities, ultimately resolving to individual actions.
9. To develop logical static structure of target system with Software Class diagram. To prepare Class Collaboration-Responsibility (CRC) cards for the Conceptual classes traced from System analysis phase. The design model should depict
    a. Relationship between classes: inheritance, Assertion, Aggregation, Instantiation
    b. Identification of objects and their purpose.
    c. Roles / responsibilities entities that determine system behavior.
10. To enhance Software Class diagram to Architecture diagram with appropriate design patterns. The patterns selected shall be justifiable and applied to individual and distinct hierarchies. Suitable Architectural Styles shall be selected and the structural elements shall be well-documented.

To represent physical module that provides occurrence of classes or other logical elements identified during analysis and design of system using Component diagram. The model should depict allocation of classes to modules. To narrate precise Program Design Language constructs separating computation from interface. To represent deployment view of the system through Architecture Diagram.

List of Projects:

1. Automated Parking lot identifier
2. Health Care Software's
3. Financial Domain
4. Appraisal Systems

5. Automate Project Administration System
6. Translator for Agriculture System
7. Development of applications manageable by Agile
8. Development of SMART applications

List of Course Seminar Topics:

1. Agile software development
2. AI and software engineering
3. Apps and app store analysis
4. Automated reasoning techniques
5. Autonomic and (self-)adaptive systems
6. Big data
7. Cloud computing
8. Component-based software engineering
9. Computer-supported cooperative work
10. Configuration management and deployment
11. Crowd sourced software engineering
12. Cyber physical systems
13. Data-driven software engineering
14. Debugging
15. Dependability, safety, and reliability

List of Course Group Discussion Topics:

1. Distributed and collaborative software engineering
2. Domain modelling and meta-modelling
3. Education
4. Embedded software
5. Emerging domains of software
6. Empirical software engineering
7. End-user software engineering
8. Fault localization
9. Formal methods
10. Green and sustainable technologies
11. Human and social aspects of software engineering
12. Human-computer interaction
13. Knowledge acquisition and management
14. Machine learning for software engineering
15. Middleware, frameworks, and API

List of Home Assignments:

Design:

1. Software visualization
2. Specification and modeling languages
3. Tools and environments
4. Traceability
5. Ubiquitous and pervasive software systems
6. Validation and verification

Case Study:

1. Software economics and metrics
2. Software engineering for machine learning
3. Software evolution and maintenance
4. Software modeling and design
5. Software process
6. Software product lines

Blog

1. Mining software engineering repositories
2. Mobile applications
3. Model-driven engineering
4. Parallel, distributed, and concurrent systems
5. Performance
6. Program analysis
7. Program comprehension
8. Program repair
9. Program synthesis
10. Programming languages
11. Recommendation systems
12. Refactoring

Surveys

1. Requirements engineering
2. Reverse engineering
3. Safety-critical systems
4. Scientific computing
5. Search-based software engineering
6. Security, privacy and trust
7. Software architecture
8. Software reuse
9. Software services
10. Software testing

Suggest an assessment Scheme:

Suggest an Assessment scheme that is best suited for the course. Ensure 360 degree assessment and check if it covers all aspects of Blooms Taxonomy.

MSE+ESE+HA+LAB+CP+CVV+SEMINAR+GD

Text Books: ***(As per IEEE format)***

1. *Ian Sommerville, 'Software Engineering', Addison-Wesley, 9th Edition, 2010, ISBN-13: 978-0137035151.*
2. *Kenneth S. Rubin, Essential SCRUM: A Practical Guide To The Most Popular Agile Process, Addison-Wesley, ISBN-13: 978-0-13-704329-3, 2012*
3. *Tom Pender, "UML Bible", John Wiley & sons, ISBN – 0764526049*

Reference Books: ***(As per IEEE format)***

1. *SorenLauesen, Software requirements: Styles and techniques, Addison Wesley, ISBN 0201745704, 2002*
2. *Dean Leffingwell, Agile Software Requirements, Addison-Wesley, ISBN-13: 978-0-321-63584-6, 2011*
3. *Charles G. Cobb, The Project Manager's Guide To Mastering Agile: Principles and Practices for an Adaptive Approach, Wiley Publications, ISBN: 978-1-118-99104-6 (paperback), ISBN 978-1-118-99177-0 (epdf), 2015*
4. *Grady Booch, James Rambaugh, Ivar Jacobson, "Unified Modeling Language Users Guide", 2nd Edition, Addison- Wesley, ISBN – 0321267974*
5. *Erich Gamma, Richard Helm, Ralph Johnson, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley Professional, ISBN-10: 0201633612 ISBN-13: 978-0201633610*
6. *Paul Clements, Felix Bachmann, Len Bass, David Garlan, Documenting Software Architectures: Views and Beyond Addison-Wesley Professional 2003, ISBN-10:0201703726, ISBN-13: 9780201703726*

Moocs Links and additional reading material: www.nptelvideos.in

www.nptelvideos.in
www.coursera.com

www.udemy.com

Course Outcomes:

1. Summarize capabilities and impact of Software Development Process Models and justify process maturity through application of Software Engineering principles and practices focusing tailored processes that best fit the technical and market demands of a modern software project.
2. Discriminate competing and feasible system requirements indicating correct real world problem scope and prepare stepwise system conceptual model using stakeholder analysis and requirement validation.
3. Formulate system specifications by analyzing User-level tasks and compose software artifacts using agile principles, practices and Scrum framework along with Propose and demonstrate realistic solutions supported by well-formed documentation with application of agile roles, sprint management, and agile architecture focusing project backlogs and velocity monitoring.
4. Compose system analysis and design specifications indicating logical, physical, deployment, and concurrency viewpoints using object-oriented analysis and design principles and Model Driven Engineering practices using UML-supported modeling tools.
5. Comprehend the nature of design patterns by understanding a small number of examples from different pattern categories and apply these patterns in creating a correct design using design heuristics, published guidance, applicability, reasonableness, and relation to other design criteria resulting in well-documented system profiles to the engineering and social community.
6. Propose multi-faceted defendable solutions demonstrating team-skills accommodating design patterns reducing the potential cost and performance impedance in order to realize system artifacts with the help of Model Driven Development practices using, scheduling, estimation and risk management activities.

CO PO Map

CO1	CO2	CO3	CO4	CO5	CO6
PO2	PO3	PO4	PO8	PO11	PSO3
2	3	3	2	1	3

CO attainment levels

CO1	CO2	CO3	CO4	CO5	CO6
PO2	PO3	PO4	PO8	PO11	PSO3
1	5	2	3	3	4

Future Courses Mapping:

Software testing and Quality Assurance, Service-oriented Software

Job Mapping:

Application Architect, Project Designer, SCRUM Role Players