REVIEW OF THE CLASS

1. Software process models
   1. What is software process? What, who, when, why?
   2. What is software process model?
   3. Models
      1. Waterfall model: traditional approach
         1. Sequential
         2. Making sure one stage completed before doing the next
         3. Suitable for projects having stable requirements: it is eliminating the waste of projects.
         4. Stages: requirements definition, design & analysis, implementation, testing, deployment, maintenance & operation.
      2. RUP: traditional approach
         1. Iterative: doing things repeatedly, incrementally
         2. Phases: Inception, Elaboration, Construction, and Transition
         3. Iterations = cycle
            1. One iteration is like one cycle of waterfall
         4. Detailed process to do things
         5. Detailed roles of team members
         6. Many documents to be written
         7. Use-case driven development
      3. Agile methods
         1. Four value propositions of Agile methods
         2. Scrum
            1. Sprint, time-boxed iteration
            2. Backlog (product backlog, sprint backlog, impediment backlog)
            3. Roles
            4. Activities

Daily scrum

Sprint review

Sprint planning

Release planning

* + - 1. XP
         1. Philosophy: take down good practices into extreme
         2. Key practices

Pair programming

Small release

Coding standard

….

1. Software project management
   1. Goals
   2. Roles
      1. PM
      2. TA
      3. BA
      4. Tester
      5. Maintainer
   3. Activities
      1. Planning
         1. Estimation
         2. Scheduling
         3. Task assignment
      2. Team building
      3. Human management
      4. Controlling and monitoring
         1. Report
         2. Problem resolving
      5. Customer collaboration
      6. Risk management
2. Software requirements & requirements engineering
   1. Software requirements
      1. User requirements (concepts of operation) & system requirements
         1. User requirements
         2. System requirements
            1. Use-case
            2. User story
      2. Functional requirements & non-functional requirements
      3. Domain requirements (functional & non-functional)
   2. Requirements engineering
      1. Requirement gathering
         1. Techniques
            1. Interview
            2. Survey
            3. Observe
            4. Record
      2. Requirement analysis and documentation
         1. Use case model
      3. Requirement validation
         1. Inspection
         2. TDD – Test-driven development (test generation)
         3. Prototyping (PoC)
      4. Requirement management
      5. Question: why do we need to do requirements validation? Why requirement inspection/review is effective?
3. Software Analysis and Design
   1. Software architecture
      1. Question: why do we need to detail/form an architecture for the software? (Log4j)
      2. Scalability -> horizontal scale vs. vertical scale
      3. Architectural critical requirements
   2. Architectural design (architectural design)
      1. Based on requirements to form the architecture
      2. Question: how does architecture affect the performance of the software?
      3. Loosely coupled vs. tightly coupled?
      4. Fine-grained vs. coarse-grained components?
   3. Detailed design (low-level design)
      1. High-level design vs. detailed design?
      2. Class diagram
      3. Sequence diagram
4. Software Verification and Validation
   1. Verification vs. Validation
      1. Why do we need both?
   2. Techniques
      1. Static inspection
         1. Review (artifacts, work products)
         2. Analysis (code analysis)
      2. Dynamic
         1. Testing
         2. Analysis
      3. Test-driven development
      4. Model-driven development
      5. …
5. Software testing
   1. Levels of testing
      1. Unit testing
      2. System testing
         1. Integration
         2. Release testing
      3. Acceptance testing
   2. Types of testing
      1. Functional testing
      2. Non-functional
         1. Performance
         2. Load
         3. Usability
         4. Security
   3. Techniques
      1. Regression testing
      2. Ad-hoc testing
      3. Smoke testing
      4. ….
   4. Concepts
      1. Test case
      2. Test steps
      3. Test data
      4. Test results
      5. Defects
      6. Test coverage
         1. Code coverage
         2. Function coverage
         3. UI coverage
         4. Path coverage
   5. Techniques
      1. Requirement-based 🡪 test cases
      2. Equivalence partition 🡪 test cases
         1. Ex. Generating test cases for testing the registration function with username having at least 8 characters and a special character.
      3. Path testing
         1. Given a snippet of code, write test cases to cover all paths
      4. Ad-hoc testing
      5. Smoke testing
6. Test automation
   1. Goals
   2. Approaches of test automation
   3. Levels
      1. Unit testing
      2. Integration testing
      3. System testing
   4. Tools
      1. Selenium
      2. Appium
      3. Katalon
7. User interface design
   1. Principles of UI design
8. Software reuse
   1. Techniques

**FINAL EXAM**

* Online
* Open book
* Condition
  + No discussion, sharing allowed
  + No phrases, sentences taken from the Internet
* Time: 100 minutes