CS425 – Software Engineering

Midterm Exam: Structure and Review Points

The Midterm Exam will consist of questions given in 2 parts, as follows:

**Part 1: Theory** – This part will be composed of Knowledge-based questions, including Short answer questions, True/False questions (some + rationale), Multiple-Choice questions and some short/small coding questions.

**Part 2: Software Engineering skills – Requirements elicitation & specification, Analysis, Design, Architecture / Coding / Problem solving skills** – This part will be composed of questions requiring Object-oriented Software analysis and design, including diagrams.

The Exam will be all computer-based. i.e. You will use computers for both parts 1 and 2. However, for diagrams, you are free to draw using Pencil and Paper; take a photo of your diagram and upload to submit.

The exam duration will be 2 hours (timed from 10 am to 12 noon).

The following are the relevant lessons/topics and sample questions:

Lesson 1: Introduction to Software Engineering and Development Methodologies

1. What is Software Engineering?

...

1. Software Development methodologies:
   1. Waterfall
      1. Document heavy
      2. Needs Requirement freezing
      3. Not Iterative
      4. Good for small/simple project – Easy to use
   2. RUP
      1. Document heavy
      2. Iterative
      3. Not light-weight
      4. Component-based development
      5. Specify requirements as Use-case
      6. 4 Phases: P19
         1. Inception
         2. Elaboration
         3. Construction
         4. Transition
   3. Agile/Scrum
      1. Iterative
      2. User Stories + Acceptance Criteria
      3. Good for change management
      4. Light-weight – low documentation
      5. Cross-functional
      6. Good for complex project
      7. Customer Interaction
      8. Greater success-rate
   4. Scrum:
      1. Scrum team including roles: ...
         1. Product Owner
         2. Scrum Master
         3. Developers, Designer, Testers, Architects, Ops etc.
      2. Daily Scrum standup meeting / Sprint execution
      3. Artifacts:
         1. Product backlog
         2. Sprint backlog
         3. Burndown chart
         4. Impediments list
         5. Software (Release)
         6. …
         7. User story map
      4. Ceremonies:
         1. Sprint planning
         2. Spring Review & Sprint Retrospective
         3. Backlog grooming or refinement
      5. Advantages:
         1. …
         2. ...
      6. Challenges:
         1. …
         2. …
         3. …
   5. XP
   6. A picture containing table

      Description automatically generated

Diagram

Description automatically generated

* + 1. Extreme Programming
    2. Pair programming
    3. TDD
  1. Lean process model
  2. Text

     Description automatically generated with low confidence
     1. Emphasize business value to customer with no waste
     2. Toyota, Ford
  3. Kanban
     1. Similar to Agile/Scrum but without the Sprints
     2. …
  4. DevOps
     1. Close collab btw dev and ops
     2. CI/CD - Continuous Integration / Continuous Delivery
     3. ...

1. Question: What is the difference between RUP and Agile/Scrum?
   1. …
   2. …
2. Question: What does RUP and Agile/Scrum have in common?
   1. …
   2. …

Lesson 2: Scrum

1. (See points about Scrum above)
2. …

Lesson 2: Working with Git & Github

1. What is Git?
   1. VCS - Distributed
   2. CLI tool obtainable git-scm.com
   3. …
2. What is the difference between Git and Github?
   1. Github is git hosted online
   2. ...
   3. Essential Git commands and their syntax/purpose:
      1. Git init
      2. Git add
      3. Git status
      4. Git commit
      5. Git log
      6. Git push
      7. Git branch
      8. Git checkout
      9. Git switch
      10. Git config
      11. Git merge
      12. Git clone
      13. Git fetch
      14. Git pull
      15. Git diff
      16. git fork???
3. Describe/Perform the process/steps for initializing a new git project repository and sharing/inviting team member(s) as collaborators/contributors/committers.
4. Describe/Perform the process/steps for contributing to a 3rd party repository.
5. …

Lesson3: Requirements 1

1. What are the types/categories of Software Requirements?
   1. Functional Reqs
   2. Non-function requirements
   3. Business reqs
2. Agile requirements
   1. In Agile, we can Specify software system requirements using:
      1. Text or statement
      2. Context diagram
      3. Use-cases
      4. User stories
3. ...

Lesson4a/b: Requirements 2 – Agile practices for gathering/specifying requirements

1. User story. What is the standard format for specifying:
   1. As a <role>
   2. I want to <goal>
   3. So that <outcome>
2. Characteristics of Good User Stories (INVEST);
   1. Independent
   2. Negotiable
   3. Valuable
   4. Estimable
   5. Small
   6. Testable
3. User story map
4. Main scenario
5. What is an Acceptance Criteria?
6. UI
7. Splitting of User stories
8. Estimating the time for User story
9. …

~~Lesson4c: Requirements 3~~

1. How do we communicate requirements?
   1. ~~Activity diagram~~
   2. ~~State machine diagram~~
   3. ~~Question: Which one of the following will be useful for Communicating Software requirements:~~
      1. ~~Deployment diagram~~
      2. **~~Activity diagram~~**
      3. ~~Class Component diagram~~
2. Question: Name some NFRs
   1. **Security**  …
   2. **Capacity**
   3. **Compatibility**
   4. **Reliability and Availability**
   5. **Maintainability  + Manageability**
   6. **Scalability**
   7. **Usability**
   8. **Performance**
   9. **Regulatory**
   10. **Environmental**
3. …

Lesson5: Analysis

1. What is Domain Driven Design? What is Domain modeling?
   1. An approach to software development where the focus is on the core Domain. ♣ We create a domain model to communicate the domain ♣ Everything we do (discussions, design, coding, testing, documenting, etc.) is based on the domain model.
2. Artifacts produced from Domain modeling?
   1. Class diagram with the attributes and multiplicities …
3. (25 points) Given a Problem Statement, perform Analysis and produce/create a domain (solution) model – e.g. see the problem statement for the Fairfield Library system
4. …

Lesson6: Architecture

1. What is Software Architecture?

**exposes the structure of a system while hiding the implementation details**.

1. Architectural styles

Diagram

Description automatically generated

* 1. …
  2. …
  3. …
  4. …
  5. What is the difference between Loose-Coupling versus Tight Coupling? Give example(s).

Timeline

Description automatically generated

* 1. What is SOA?

Diagram

Description automatically generated

* 1. …

1. …

Diagram

Description automatically generated

Lesson7: Design

1. Domain entities
2. Value objects

Timeline

Description automatically generated

1. Domain Services

Interface is defined in terms of other domain objects

Timeline

Description automatically generated

1. Data Access obects
2. Domain events

Timeline

Description automatically generated

1. In Design, what does the Principle of High Cohesion, Loose Coupling mean?
2. What are the SOLID principles for OO software Design?

SOLID is an acronym that stands for five key design principles: **single responsibility principle, open-closed principle, Liskov substitution principle, interface segregation principle, and dependency inversion principle**. All five are commonly used by software engineers and provide some important benefits for developers.

S

1. …

Lesson8: Clean Code

1. What do we mean by “Clean code”?
   1. ...
2. Clean Code best practices for:
   1. Classes, Variable
   2. Methods
   3. Commenting
3. …

**Note:** *For some of the questions/tasks in the part2 of the Exam, you will be expected to take screenshot(s) of your work/result(s), and save them to an image file (.png or .jpg or .jpeg only) and include these in the zip file, which you will submit.*

***Also, when you take the screenshots, it should be of the entire computer screen (NOT a snippet or a window***

***MacOS: Cmd + Shift + 3***

***Windows: Windows key + PrintScreen key***

**//-- The End --//**