CS5351 Software Engineering
2024/25 Semester B
Quiz
(150 minutes)

	Max	**Mark**
1	20	
2	20	
3	20	
4	20	
5	20	
Total	**100**	
Grade		

Student Name: ______________________

Student No.: ____________________ Seat No.:____________

This is an open-book quiz.

1. Software Process (20 points, 4 points for each pair of missing item and improvement)

Your team develops a software application using the following "Scrum" process. Your project manager has an impression that some key elements of Scrum are missing but cannot pinpoint what they are. Your task is to review the characteristics of the project mentioned below and present what has been missing to your project manager.

Point out **five** missing items and explain how the missing items, if they are in the process, will improve the current software process used by this project.

Case Study 1

- The project has a project backlog and a task board. They include all the user functional requirements to be implemented.
- Users and the project team prioritize the tasks in the project backlog.
- Each developer preferably picks a higher-priority task in the project backlog to achieve once he/she completes the current task.
- Each implemented task is reviewed and tested by the corresponding users.
- At the end of each month, the project manager picks the tasks users have approved to include in the upcoming release.
- To guard against mistakes, the project conducts requirement validation, design review (before the team commits to a design to be reviewed), modern code review (after the code has been developed), and testing.

You may or may not consider the following aspects:

- How do we know the overall project progress is on schedule?
- How does a developer know the size of requirements to deliver in a period?
- Can we forecast when the project will be completed?
- Do the project backlog items reflect the complete list of items to deliver?
- Are backlog items being delivered in the order reflecting a Scrum process?

Missing Item 1: Dividing the project into sprints, and each sprint has a sprint backlog. Improvement made: The progress on backlog can be tracked sprint-wise against the plan and become more transparent to developers.
Missing Item 2: The project does not estimate each backlog item's effort (or story point or time needed). Improvement made: With the estimation, individual developers can self-estimate how many story points one can deliver at the start of the current sprint. It will make progress more predictable.
Missing Item 3: The velocity of the development is unknown. Improvement made: Tracking the velocity and the remaining backlog items with effort estimation, the project can forecast the sprint (in, say, best, worst, average cases, for example) to complete all the backlog items.
Missing Item 4: The project does not track any technical debt (and non-functional requirements, which, strictly speaking, is not a technical debt.) Improvement made: Identify and review technical debts. Include them in the project backlog. The resulting project backlog provides a more realistic estimate of the project completion time, and backlog items may require re-prioritization to speed up the project velocity.
Missing Item 5: A higher-priority task is not enforced to be implemented in the current sprint as the notion of sprint is missing. Improvement made: Pick backlog items to be accomplished in the current sprint before picking backlog items outside.
[backup] Missing Item __: Improvement made:

(**Note to TA**: Students may split them into multiple missing items. Each aspect counted as a standalone missing item. The missing items for the process are open-ended.)

2. Software Requirement Engineering (20 points)

(a) Suppose **A** is a system-wide and major functional requirement, and **B** is a non-functional requirement associated with **A**. Give **one** example **for a pair of A and B in your course project**. Explain why **B** is a non-functional requirement of **A**. (3 points)

Example 1
Example of A:
Example of B:
Explanation:

(b) Is anything wrong with the operation of the software to be built if the non-functional requirements of the software are collected but not implemented? (3 points)

If non-functional requirements are not implemented, the software may not be suitable for users to use (perhaps, fail to support the intended business goal) or violate the software's business constraints.

(c) Requirement engineering involves elicitation, specification, validation, and negotiation. We need elicitation and specification to understand and describe the problem to be solved by the system to be built. (8 points)

(1) Why do we need requirement validation and negotiation?

requirement validation: Validation with the customers to uphold the validity of the requirement.
requirement negotiation Negotiation clarifies and reshapes the scope and balance between functional and non-functional requirements.

(2) Do we miss anything about these four process steps if we only listen to the stakeholders, record their requirements, and check our understanding of individual requirements with them? Also, give an example of a negative consequence if we follow the stated scenario to collect the requirement.

Only elicitation, specification, and **partial** validation are conducted in the given scenario.

We failed to include the following:

- a requirement negotiation process and
- the component for analysis of all requirements in the validation process.

Thus, non-functional requirements are missed, functional requirements are unclear to be consistent, and the balance between functional and non-functional requirements is not considered in the requirement engineering process. The specification becomes incomplete.

Study Case Study 2 and answer the following questions.

Case study 2:

John aims to determine a customized debugger's requirements for developing the program scripts for training deep-learning models and their inferences on samples for his company. He collects examples of these buggy program scripts and their bug-fixed versions from the developers of these scripts. He studies what bugs the program scripts contain, the errors produced from these scripts, and the information needed in the bug-fixed versions to quantify the code region that contains the errors. He can write a requirements specification for the debugger. He walks through the screen sketches of the debugger with each developer separately to show that the debugger will visualize the errors and show the code regions containing the bugs so the developer can easily fix the same buggy program scripts in the future. The screens in the specification are revised according to the developer's feedback.

(d) Is there any problem regarding the process steps for requirements engineering? Explain your answer. (4 points)

The requirement elicitation and validation processes could be problematic.

- The current requirements are produced by John rather than by the developers. They may not reflect the actual needs of the developers in debugging program scripts.
- The validation process only validates the screen but not the functionality of the debugger.

(e) Suppose John wants to take *social relativism* as the position in the negotiation process. Your task is to recommend an elicitation technique to use so that his process can be improved. Elaborate your answer. (2 points)

Recommended elicitation technique We need a consensus. So, techniques explicitly requesting developers to create a consensus will be good enough. Examples are (1) brainstorming session, (2) group storytelling, (3) focus group, and (4) facilitated workshop.
Justification of the recommendation See the points in the answer for "Recommended elicitation technique"

3. Software Architecture (20 points)

(a) In considering the software architecture of a system, we need to consider various factors in domain model, design, vocabulary, and how we present the software architecture. What do these factors refer to? (4 points)

Domain model • It describes the environmental constraints designers cannot control and the architecturally significant requirements.
Design • It describes the elements that designers can control (i.e., these elements and their arrangements designers can make design decisions).
Vocabulary • Software architecture is to communicate the non-functional aspect of the system to meet user non-functional requirements. We should choose the terminologies that our users use to improve the effectiveness of communications.
Presentation • As long as users find it effective, any presentation can be used.

(b) There are many quality attributes (QAs), too many to recall, consider, and apply for a typical project. How should we identify the major quality attributes of a system? (4 points)

We should first ensure the requirements capture the non-functional behavior of the system and the key functional requirements (e.g., key use cases). The non-functional aspects of these requirements should be measurable if possible. We can then assign them with the corresponding quality attributes and group requirements by quality attributes.

(c) Attribute-driven design (ADD) is a methodology to consider non-functionality quality attributes in the architectural design of a system. In the tutorial on software architecture, we chose a (family of) technology with good scores in the quality attributes we want the architectural element to attain. we rely on three subprocesses to complete the case study in the tutorial: identification of quality attributes for an architectural element (e.g., a Batch Layer in our tutorial in Iteration 3), the identification of the set of relevant families/technologies, and the evaluation of these families/technologies. (12 points)

(1) Suppose we can only identify one technological family for each architectural element and one technology under the selected family when applying the ADD methodology to our system. ADD can be meaningfully applied, do you agree with it or not?

☐ Agree ☐ Disagree

(2) What is the implication on the requirements and implementation if the "score" on a quality attribute we are concerned for an architectural element cannot be satisfactorily addressed by the single available family/technology for selection?

Implication on requirements Committing the design to this selected family/technology will threaten the system design to fulfill the system's requirements.
Implication on implementation The final implementation to fulfill the system's non-functional requirements becomes more difficult. (or the project cost will increase and take longer).

(3) Suppose we are not quite comfortable with the current design to address the quality attributes of the current architectural element. What should we do? Give one example is enough.

You may consider the following questions in evaluating your answer.

- Are the design choices available to us too limited for the current architectural element?
- Is it due to how we decompose an architectural element in a higher level of abstraction into the current set of architectural elements so that meeting the quality attribute by the current architectural element becomes difficult?

If the current design is unsuitable for the current architectural element, we should look for other choices. We may research other potential solutions for this architectural element if none is good enough. Alternatively, we may go up one level of architectural abstraction to determine whether any alternate system decomposition of the higher-level architectural element is feasible and whether the quality attributes for the requirements can be addressed in that alternate system decomposition.

4. Technical Debt (20 points)

This question relates to the case study in the tutorial for technical debt (developing a testing system).

(a) If we choose an adequate software architecture for our software system (such as the one in our tutorial for Technical Debt), that does not fit the overall design goal very well, do we agree that we will incur a technical debt based on this choice? (8 points)

☐ Agree ☐ Disagree

Suppose this is a technical debt. We know that a technical debt can be classified into four dimensions. Classify this technical debt. Explain your choice.

Type Design debt because this is an architectural design problem.
Intentionality Intentional debt because we adopt an architecture that is merely adequate but not a good fit.
Time Horizon It is a long-term debt. This is because adopting this decision results in many smaller debts appearing in the system, and the development team needs to handle the consequences over time.
Degree of Focus This is an unfocused debt. Architectural design is an abstract design decision. It limits many lower-level designs spanning over a few components to produce effective solutions.

(b) There are three typical patterns for increasing the technical debts in a project. Based on the case study in Tutorial 5 (for developing a testing system), give **one example** of "Schedule Pressure" and **one example** of "Duplication Code" from the case study. You only need to describe them to highlight the key points briefly. (4 points)

Schedule Pressure The background states the following. • To speed up the delivery of a workable system as soon as possible, the team starts developing the "workarounds" for the component.
Duplication Code In the description of the [Workaround] design choice for Oracle, • Decide to copy the rule implementations from several open-source projects implementing various security vulnerability detection rules, wrap them as a set of functions *F* (one for an open-source project), and call a subset of functions on a case-by-case basis.

(c) There are seven debts listed in the case study. How should we generally deal with these technical debts before developing the final solution (using the [Original] design choice)? Also, give three examples from a project backlog perspective. (8 points)

Strategy: We should manage the technical debts from the backlog perspective.
Example 1: • Identify debt (e.g., label them as Debt X). • Track them as backlog items in a project (place them on task board) • Monitor the amount of rework (e.g., Events triggered reworks). • Measure % of the budget spent on rework due to debt and its repayment. • Repay the debt (e.g., apply [Repay 1] or [Repay 2])
Example 2:
Example 3:

5. Modern Code Review (MCR) (20 points)

Case Study 3
Candy creates a branch on a code file in GitHub. She completes the debugging task on this code and uploads a revised file to that branch with brief comments on the bug in the code and how she repairs the bug in the changes. She then looks around the office and finds Tom and Betty available to review her change. So, she put down these two teammates as reviewers on GitHub to review her changes. Tom and Betty receive notifications from the system. Tom quickly studies the changes and the comments and approves the changes through the system. Betty is busy with her other tasks. She reviews the changes three days later and finally approves the changes. Betty also suggests a simpler solution to fix the bug with her reasons. After adopting Betty's suggested solution and getting approval from Betty about the change, Candy commits the changes to the main branch after a final check and approval by her supervisor, James.

(a) MCR is informal, tool-based, asynchronous, and focused on reviewing patches. In which way does Case Study 3 demonstrate these four aspects? Give an example in each aspect. (10 points)

Informal Candy selects the reviewer set informally by checking who is available in the office on spot.
Tool-based GitHub is used. The system notifies reviewers. Changes and approval are recorded in the system.
Asynchronous The two reviewers can choose their different timeslots to complete the review.
Focused on reviewing patches Reviewers review the changes.

(b) Betty has approved Candy's original solution as well. One possible alternate workflow is that Candy does not adopt Betty's suggestion and directly seeks James's approval on her original solution. Suppose James approves Candy's original solution upon observing that the two reviewers both have approved the changes. Which key purpose of MCR demonstrated in this scenario does James miss? What should James do so that the team can get a better solution? How could James know that an alternative solution probably exists? If the better solution is Betty's suggestion, what should Candy do? (10 points)

Key purpose Code improvement. Note: It is unclear which one of the two possible solutions is better.
James' action James should not approve the changes and request Candy to communicate with Betty to resolve the better solution. James should read the comments on GitHub as Betty has stated them as comments.
Candy's action If Betty's solution is the final choice, the workflow should send the revised solution to both reviewers (or at least to Betty) for approval.

- END -