605.601 - Foundations of Software Engineering

Final Examination

Sabbir Ahmed

December 14, 2020

Please answer any five (5) questions. Please type your answers below the question under "Answer".

1. (a) What is a Use Case? Please provide an example.

   A Use Case is a written list of actions or event steps that illustrate the interactions between a role and a system to achieve a goal. For example, a use case of a web application for a restaurant would be processing orders for the customers. The customers' orders would include selecting available items from the restaurant menu, adding up their prices, calculating the overall tax and other charges and finally providing the users with options to process their payments.

   (b) Please elaborate on how you describe a Use Case.

   I would describe a Use Case as a list of steps that describe how a user would utilize my software product. For example, if I developed a mobile application that keeps track of a user's workout routine, a Use Case would be the client selecting from a list of available exercises to curate their routine. A simpler Use Case would be the client saving their routines in the application using local storage or a cloud database.

   (c) Do you think there is a difference between a Use Case and a User Story? Do you have any preference in using any of them?

   Although both of the approaches are similar in concept, they both serve different purposes. A User Story is usually more superficial than a Use Case. A User Story focuses more on the general requirements rather than the design implementation of the system. I personally do not favor one approach over the other since I believe they are both essential to good software development methodology.

   (d) Since you may have used Use Cases in your group projects, can you list one or two benefits or limitations in using the Use Cases?

One limitation of Use Cases in the course projects I noticed was that there were no standard definitions for the design. Some portions of the Use Case language and graphics followed the course instructions, but the other portions had to be customized and interpreted specifically for the project. Another limitation I noticed was that it was not possible to fit in non-interactive requirements in the system, such as built-in functions and algorithms.

2. (a) What does design mean to you? What are the major steps in a software design process?

In the software development process, design is the process of transforming project requirements into software components. The major steps in the software design process are:

i. representing the system architecture

ii. modeling the interfaces interacting with users, other systems, and internal components

iii. designing the individual components

(b) What does separation of concerns mean to you?

Separation of concerns is a design principle for separating a convoluted design problem into modular pieces to be handled independently.

(c) Are you familiar with the fundamental principles of design? Please list any three principles that you know.

Yes, I am familiar with the fundamental principles of design. Three example principles are:

i. the quality of the design should be emphasized during implementation and not afterwards

ii. the design should be prepared to accommodate changes in future updates

iii. the design should exhibit uniformity and integration

(d) What does architecture mean to you? Can you describe the relationship between design and architecture?

Architecture in software design refers to the software system structures and how they interact independently and with the system as a whole. Software architecture and

design represent different perspectives of the software system. Architecture is more abstract and emphasizes strategy, structure and purpose, while design focuses on the implementation.

(e) Why do we need architecture? Can you describe a basic form of architecture? Please elaborate.

Software architecture is essential because it is used to analyze the effectiveness of the design in meeting its constraints and requirements and to reduce the risks associated with the construction of the software. A basic form of architecture deals with structures and the behavior between them.

3. (a) What is a software component (from a design perspective)?

A software component is a modular portion of a system that covers a set of its implementation and exposes a set of its interfaces.

(b) What are the different views that a component can be presented as?

The different views are:

i. conventional - a component is a functional element that encompasses processing logic
ii. control - a component coordinates interaction among domain components
iii. domain - a component implements required functionality
iv. infrastructure - a component supports processing
v. object oriented - a component comprises a set of collaborating classes

(c) What does 'refactoring' mean to you? Does the concept of refactoring have any value to you as a software engineer? And why?

Refactoring is the process of updating components in the software system in order to improve implementation functionality while maintaining the exterior behavior. This process is very valuable to a software engineer since maintenance is essential for a system to thrive. If a software product has a feature that can be upgraded due to new third party releases, the system should undergo a refactor to obtain the benefits of the upgrade.

(d) What is a design pattern?

Design patterns are general established solutions to commonly occurring problems in software design.

(e) How are the design patterns used in software engineering?

Design patterns can speed up the development process by providing proven development methodologies. If a design pattern is used, it helps to prevent subtle issues that can cause major problems. Design patterns also improve code readability for developers who are familiar with the paradigms.

4. (a) What is software testing? Describe fault, error, and failure.

Software testing is the discovery process to prevent failures in a software system. A fault is a physical defect within a hardware or software component. An error is any deviation from accuracy or correctness in behavior of a system. A failure is is an externally visible event representing a deviation from the system's required behavior.

(b) Why do you think testing is important for software engineering?

Testing is essential for software engineering because it exposes any weaknesses or flaws in the system before being introduced to the customers.

(c) Can you provide an example of software testing? How does software testing impact software development?

An example of software testing is unit testing, where a small modular component of the system is tested for its validity. Software testing is a major step in the software development methodology. Oftentimes, the results of the testing step moves the development phase back into the implementation stages to improve the quality.

(d) Can you describe how software testing can help an entire software development team?

Software testing can expose any flaws made by the development team and gives them an opportunity to revise their implementation before the product is finalized.

(e) Please define what software tester means to you. What are the different types of software tester you know of? Can a software tester help improve the quality of a software?

A software tester is usually an individual or a team, oftentimes external to the core development team, who are responsible for validating their assigned software product

through rigorous tests. Software testers can come in many different disciplines, such as penetration testing teams who attempt to infiltrate the security of a software. Software testers are effective in delivering a final product to the clients with minimal issues and flaws.

5. (a) What does 'Agility' mean to you? Do you believe 'Agility' must be a common characteristic of any software today?

"Agility" refers to the characteristics borrowed from the Agile methodology that promotes rapid and adaptive development. It is a common feature of many software today since the rate of technological improvement and customer demand increases with time.

(b) What is 'Agile' methodology? Do you agree with the Manifesto of Agile software development?

An Agile methodology is a software development methodology that advocates for frequent collaboration between the development teams and their end-users through iterative development. Several derivations of the method exist to account for company or product constraints; however, the general principles of adaptive planning, evolutionary development, and continuous integration are maintained. I agree with the Manifesto of Agile software development because I use portions of its principles at my work during development.

(c) How do you define DevOps? Do you see the difference between Agile and DevOps? If so, why? If not, why not?

DevOps derives several aspects of Agile methodologies. The practice advocates for rapid collaboration and delivery. The two practices are often viewed as complementary. There are several key differences between the two concepts. Agile methodologies attempt to address communication gaps between the clients and developers, while DevOps focuses on the development and operations teams.

(d) If you had an opportunity to proceed with your group project, which software development approach would you use – Agile or DevOps? Please elaborate.

If I had an opportunity to proceed with your group project, I would pick an Agile methodology because I am more familiar with the practice at work. Also, choosing DevOps for a small scale project such as our group project would not be appropriate

since the separation of concerns did not require group members to distinguish between development and operational tasks.

(e) In a real practice, how would you decide to use waterfall, iterative and incremental, Agile or DevOps? Do you have any preference?

In a real practice, I would prefer Agile methodologies simply because I am most familiar with the practice than the others. I have attempted to develop using the waterfall method once, but the project ended up incomplete due to the lack of constant verification from the clients.