IQRA NATIONAL UNIVERSITY

Spring 2020 Final-Term Examination

Course Name	Course Code	Max. Marks	Max. Time	Date	Instructor
Software Design & Architecture	SEE-302	50	6 hrs. 9-3 PM	23rd June 2020	Aasma Khan

---

ID: 14646 **NAME:** Abdul Musawer

- **Attempt all questions.**
- **Marks will be given as per the DEPTH of the answer, not LENGTH.**

Question No: 01 (5+5)

a) What is Software Architecture? Why is software architecture design so important?

Answer: <u>SOFTWARE ARCHITECTURE:</u>

The software architecture of a system depicts the system's organization or structure, and provides an explanation of how it behaves. A system represents the collection of components that accomplish a specific function or set of functions. In other words, the software architecture provides a sturdy foundation on which software can be built.

<u>IMPORTANCE OF SOFTWARE ARCHITECTURE DESIGN:</u>

Communication among stakeholders Every stakeholder has different concerns; SW Architecture can be used as a basis for mutual understanding, negotiation, consensus, and communication among stakeholders.

Early design decisions

1. Software architecture manifests the earliest design decisions, these decision are
    a. the most difficult to get correct and
    b. the hardest to change later
2. The Architecture Defines Constraints on Implementation
    a. Implementation must conform to
    b. prescribed design decisions

    c. resource allocation decisions
3. Architectures are both prescriptive and descriptive
    a. The Architecture Dictates Organizational Structure.
4. work breakdown structure
    a. work assignments to teams
    b. plans, schedules, budgets
    c. communication channels among teams
    d. dependency and coupling important for work assignment

b) Explain any four tasks of the architect.

Answer: **FOUR TASKS OF THE SOFTWARE ARCHITECT:**

1. A software architect needs to **interact with clients**, product managers, and developers in order to **envision, model and provide** initial models and designs that can be built. This role also may cover the meeting potential or current customers.
2. A software architect has to constantly **review the code** to ensure the quality of the design by avoiding complexity, **advocating clarity** and to do this with the team. This usually requires hands-on work in terms of developing prototypes, contributing code or **evaluating technologies**.
3. The role of a software architect includes **collaborative working** with a degree of humility and providing **mentoring** as required. Such collaboration also allows the architect to become familiar with the skills and interests in the team and to share their knowledge with the rest of the team. Humility is required to ensure that all the team is listened to, as they may have more specific experience or knowledge for the problem at hand.
4. Ensure software meets all requirements of quality, security, modifiability, extensibility etc.

Question No: 02 (10)

Explain Architecture Business Cycle (ABC) in detail with figure.

Answer:

ARCHITECTURE BUSINESS CYCLE (ABC):

"Software architecture is a result of technical, business, and social influences. Its existence in turn affects the technical, business, and social environments that subsequently influence future architectures. We call this cycle of influences, from the environment to the architecture and back to the environment, the Architecture Business Cycle (ABC)."

The organization goals of **Architecture Business Cycle** are beget requirements, which beget an architecture, which begets a system. The architecture flows from the architect's experience and the technical environment of the day.

Three things required for ABC are as follows:

(i) **Case studies** of successful architectures crafted to satisfy demanding requirements, so as to help set the technical playing field of the day. (ii) **Methods** to assess an architecture before any system is built from it, so as to mitigate the risks associated with launching unprecedented designs. (iii)**Techniques** for incremental architecture-based development, so as to uncover design flaws before it is too late to correct them.

ABC Activities Include:

1. Create the business case.
2. Understand the requirement.
3. Create the architecture.
4. Document and communicate the architecture.
5. Analyze the architecture.
6. Implement the system based on Architecture.
7. conforms the implementation.

How the ABC Works :

1. The architecture affects the structure of the developing organization. An architecture prescribes a structure for a system; as we will see, it particularly prescribes the units of software that must be implemented (or otherwise obtained) and integrated to form the system. These units are the basis for the development project's structure. Teams are formed for individual software units; and the development, test, and integration activities all revolve around the units. Likewise, schedules and budgets allocate resources in chunks corresponding to the units. If a company becomes adept at building families of similar systems, it will tend to invest in each team by nurturing each area of expertise. Teams become embedded in the organization's structure. This is feedback from the architecture to the developing organization.

In the software product line case study, separate groups were given responsibility for building and maintaining individual portions of the organization's architecture for a family of products. In any design undertaken by the organization at large, these groups have a strong voice in the system's decomposition, pressuring for the continued existence of the portions they control.

2. The architecture can affect the goals of the developing organization. A successful system built from it can enable a company to establish a foothold in a particular market area. The architecture can provide opportunities for the efficient production and deployment of similar systems, and the

organization may adjust its goals to take advantage of its newfound expertise to plumb the market. This is feedback from the system to the developing organization and the systems it builds.

3. The architecture can affect customer requirements for the next system by giving the customer the opportunity to receive a system (based on the same architecture) in a more reliable, timely, and economical manner than if the subsequent system were to be built from scratch. The customer may be willing to relax some requirements to gain these economies. Shrink-wrapped software has clearly affected people's requirements by providing solutions that are not tailored to their precise needs but are instead inexpensive and (in the best of all possible worlds) of high quality. Product lines have the same effect on customers who cannot be so flexible with their requirements. A Case Study in Product Line Development will show how a product line architecture caused customers to happily compromise their requirements because they could get high-quality software that fit their basic needs quickly, reliably, and at lower cost.

4. The process of system building will affect the architect's experience with subsequent systems by adding to the corporate experience base. A system that was successfully built around a tool bus or .NET or encapsulated finite-state machines will engender similar systems built the same way in the future. On the other hand, architectures that fail are less likely to be chosen for future projects.

5. A few systems will influence and actually change the software engineering culture, that is, the technical environment in which system builders operate and learn. The first relational databases, compiler generators, and table-driven operating systems had this effect in the 1960s and early 1970s; the first spreadsheets and windowing systems, in the 1980s. The World Wide Web is the example for the 1990s. J2EE may be the example for the first decade of the twenty-first century. When such pathfinder systems are constructed, subsequent systems are affected by their legacy.

Question No: 03 (10)

Explain ABC Activities?

Answer:

ABC ACTIVITIES:

1. Creating the business case for the system
2. Understanding the requirements
3. Creating or selecting the architecture
4. Documenting and communicating the architecture
5. Analyzing or evaluating the architecture
6. Implementing the system based on the architecture

7. Ensuring that the implementation conforms to the architecture

1. Creating the Business Case for the System

Creating a business case is broader than simply assessing the market need for a system. It is an important step in creating and constraining any future requirements. How much should the product cost? What is its targeted market? What is its targeted time to market? Will it need to interface with other systems? Are there system limitations that it must work within?

These are all questions that must involve the system's architects. They cannot be decided solely by an architect, but if an architect is not consulted in the creation of the business case, it may be impossible to achieve the business goals.

2. Understanding the Requirements

There are a variety of techniques for eliciting requirements from the stakeholders. For example, object-oriented analysis uses scenarios, or "use cases" to embody requirements. Safety-critical systems use more rigorous approaches, such as finite-state-machine models or formal specification languages.

One fundamental decision with respect to the system being built is the extent to which it is a variation on other systems that have been constructed. Since it is a rare system these days that is not similar to other systems, requirements elicitation techniques extensively involve understanding these prior systems' characteristics.

Another technique that helps us understand requirements is the creation of prototypes. Prototypes may help to model desired behavior, design the user interface, or analyze resource utilization. This helps to make the system "real" in the eyes of its stakeholders and can quickly catalyze decisions on the system's design and the design of its user interface.

Regardless of the technique used to elicit the requirements, the desired qualities of the system to be constructed determine the shape of its architecture. Specific tactics have long been used by architects to achieve particular quality attributes. It is not until the architecture is created that some tradeoffs among requirements become apparent and force a decision on requirement priorities.

3. Creating or Selecting the Architecture

In the landmark book The Mythical Man-Month, Fred Brooks argues forcefully and eloquently that conceptual integrity is the key to sound system design and that conceptual integrity can only be had by a small number of minds coming together to design the system's architecture.

4. Communicating the Architecture

For the architecture to be effective as the backbone of the project's design, it must be communicated clearly and unambiguously to all of the stakeholders. Developers must understand the work assignments it requires of them, testers must understand the task structure it imposes on them, management must understand the scheduling implications it suggests, and so forth. Toward this end, the architecture's documentation should be informative, unambiguous, and readable by many people with varied backgrounds..

5. Analyzing or Evaluating the Architecture

In any design process there will be multiple candidate designs considered. Some will be rejected immediately. Others will contend for primacy. Choosing among these competing designs in a rational way is one of the architect's greatest challenges.

Evaluating an architecture for the qualities that it supports is essential to ensuring that the system constructed from that architecture satisfies its stakeholders' needs. Becoming more widespread are analysis techniques to evaluate the quality attributes that an architecture imparts to a system. Scenario-based techniques provide one of the most general and effective approaches for evaluating an architecture.

6. Implementing Based on the Architecture

This activity is concerned with keeping the developers faithful to the structures and interaction protocols constrained by the architecture. Having an explicit and well-communicated architecture is the first step toward ensuring architectural conformance. Having an environment or infrastructure that actively assists developers in creating and maintaining the architecture (as opposed to just the code) is better.

7. Ensuring Conformance to an Architecture

Finally, when an architecture is created and used, it goes into a maintenance phase. Constant vigilance is required to ensure that the actual architecture and its representation remain faithful to each other during this phase. Although work in this area is comparatively immature, there has been intense activity in recent years.

Question No 04: (20)

Pair programming is an agile software development technique in which two programmers work together at one workstation. One types in code while the other reviews each line of code as it is typed in. The person typing is called the driver. The person reviewing the code is called the observer. The two programmers switch roles frequently (possibly every 30 minutes or less).

Suppose that you are asked to build a system that allows Remote Pair Programming. That is, the system should allow the driver and the observer to be in remote locations, but both can view a single desktop in real-time. The driver should be able to edit code and the observer

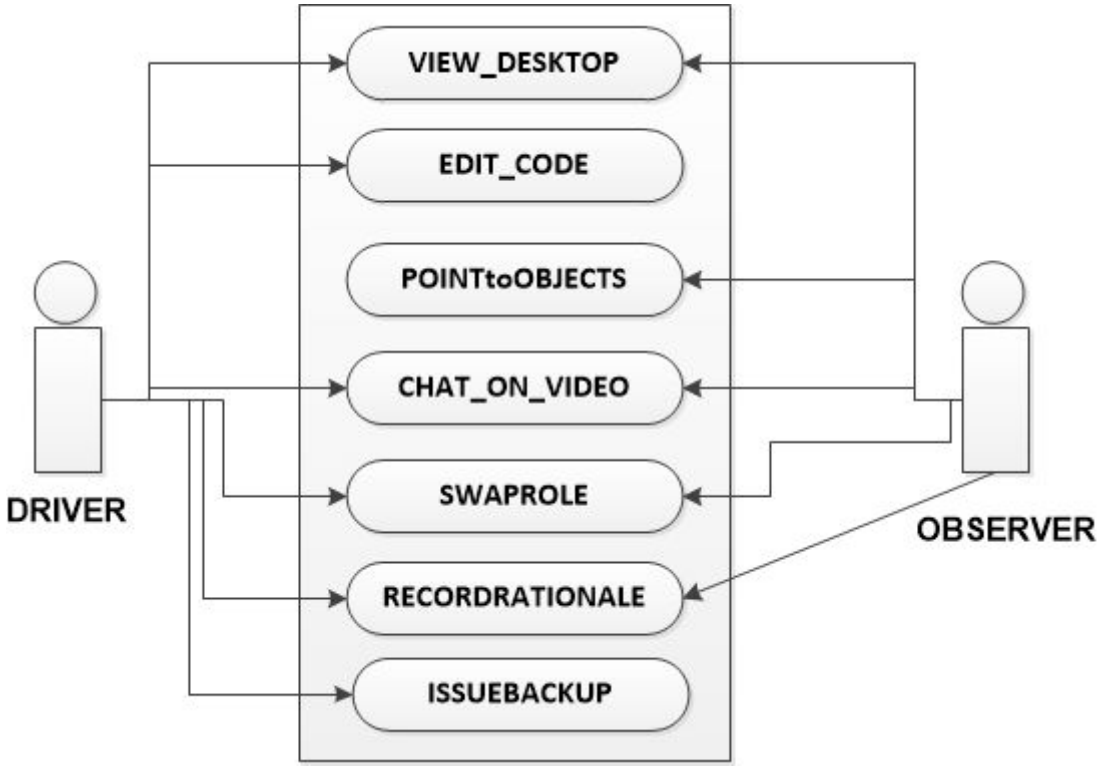
should be able to “point” to objects on the driver’s desktop. In addition, there should be a video chat facility to allow the programmers to communicate. The system should allow the programmers to easily swap roles and record rationale in the form of video chats. In addition, the driver should be able to issue the system to backup old work.

- Draw a use case diagram to show all the functionality of the system.
- Describe in detail four non-functional requirements for the system.
- Give a prioritized list of design constraints for the system and justify your list and the ordering.
- Propose a set of classes that could be used in your system and present them in a class diagram

ANSWER:

- **Draw a use case diagram to show all the functionality of the system.**

Answer:



- **Describe in detail four non-functional requirements for the system.**

Answer:

1. Ease of use - the front-end interface must be simple and easy to use.
2. Real-time performance - the Observer should be able to see the changes made by the Driver immediately without delay; the video chat should be smooth without delay.
3. Availability - the system should be available to both programmers all the time.
4. Portability - the programmers should be able to use the system regardless of what computer and operating system used by the programmers.

5. Security - the backup code should be kept securely and be protected from unauthorized access.
6. Cost - users should not pay for this above 100$ per month as a membership.
7. Reliability - the system should be reliable, i.e., it should not crash when the internet speed is slow and when the internet connection is suddenly down the user should be able to resume the session at a later time.

- **Give a prioritized list of design constraints for the system and justify your list and the ordering.**

Answer:

Example 1:

"the system should be portable" is a NFR. This NFR may lead to a constraint on the programming language used for the implementation of the system (e.g., the programming language Java (rather than C and C++) might be preferred in order to meet this NFR).

Example 2:

"security - the system must be secured" is a NFR. The design constraints could be a user authentication must be in place, the communication protocol must be encrypted, and/or the data must be stored on a server behind the firewall.

- **Propose a set of classes that could be used in your system and present them in a class diagram**

Answer: