NAME MUHAMMAD SOHAIL

ID # 14071

SUBJECT SOFTWARE

DESING ARCHITECTURE

SUBMITTED TO ; AASMA

KHAN

SEMESTER 6TH

DATE 23/06/2020

Q NO 1 : What is Software Architecture?

Ans: When we talk about the architecture of a software, we talk about a plan that describes a set of aspects and decisions that are important to a software. This implies taking into consideration all kinds of requirements (performance, security, etc.), the organisation of the system, how do the the system parts communicate between each other, if there are some external dependencies, what are the guidelines and implementation technologies, what are the risks to take into consideration, and many more.
Going back to what I said, the architecture of a Software includes decisions. If you would make a change to one of the software's aspects or decisions, would there be a huge impact and would it be very hard to change it?
Most probably, the answer will be yes. So why is it that we talk so much about having a good software architecture?

Why is Software Architecture so important?

Here are the three main reasons why a good software architecture is so important when it comes to development.

- **A basis for communication:**
-

software architecture is a sort of plan of the system and is primordial for the understanding, the negotiation and the communication between all the stakeholders (user side, customer, management, etc.). In fact, it makes it easier to understand the whole system and therefore makes the decisions process more efficient.
The earliest decisions:

the first decisions taken are at this stage. Those early decisions have a huge importance on the rest of the project and become very difficult to change the more we advance in the process
Transferability of the model: software architecture defines the model of the software and how it will function. Having it makes it possible to reuse this model for other softwares; code can be reused as well as the requirements. All the experience we get while doing that architecture is also transferred. This mean that we know and can reuse the consequences of the early decisions we took on the first place.

In other words, the architecture will define the problems you might encounter when it comes to implementation. It also shows the organisational structure and makes it much easier to take decisions and manage all sort of change. It also permits us to get a better estimate of the time & costs of a project.

Q NO 1 Explain any four tasks of architect.

An ARCHITECT'S TASKS

- Establish dynamic control relationships among different subsystem in terms of data flow, control flow orchestration, or message dispatching.
- Consider and evaluate alternative architecture styles that suit the problem domain at hand.

ARCHITECTURAL INFLUENCES

- Influences

_system stakeholders
_Developing oraganization
_Architects background and experience
_technical enviroment

- **PRECAUTIONARY MEASURES**
-

_know your constraints
_Early engagement of stakeholder

Q.2:- Explain Architecture Business Cycle (ABC) in detail with figure.

Ans:-

Software architecture is a result of technical, business and social influences.These are in turn affected by the software architecture itself. \ This cycle of influences from the environment to the architecture and back to the environment is called 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.

Q NO 3 EXPLAIN ABC ACTIVITIES?

a. Create the business case.

B) Understand the requirement.
C) Create the architecture.
D) Document & communicate the architecture.
E) Analyse the architecture.
F) Implement the system based on architecture
G) Confirms the implementation.

Creating the business case for the system

It is simple to create a business case than understanding the needs of marketHow much should be the product cost?What is the Targeted market?What is the targeted time to market?Will it need to interface other system?Are there system limitations

Understanding the requirements

There are variety of techniques to understand requirements from stakeholders.Object oriented analysis: use cases & scenariosSafety Critical Systems:Finite state machine modelsFormal specification languagesQuality attributesPrototypesRegardless of technique used, --the desired qualities of the system to be constructed determine the shape of architecture.| Website for Students

Creating the architecture

Conceptual integrityA small no. of minds coming together to design the system's architecture.

Communicating the architecture

For effective architectureIt must be communicated clearly and unambiguously to all stakeholders.Developers must understand work assignments.Testers must understand the task structuresManagement must understand the scheduling implications

Analyzing the architecture

Out of multiple designs, after analyzing, some design will be accepted or some are

rejected.Evaluating an architecture for the qualities it supports is essential to ensure the stakeholders satisfaction (needs).Scenario- based techniques are for evaluation of architecture.| Website for Students

Implementing based on the architecture

Concerned with keeping the developers faithful to the structures.Should have an environment that assists developers in creating the architecture.Ensuring conformance to an architecturefinally, when an architecture is created and used, it goes into maintenance phase.Constant vigilance is required to ensure that actual architecture and its implementations remain faithful to each other.

Confirming the implementations

The final step in the cycle is to confirm the implementations and reviewed by a single architect or small group of architects.Gather both the functional requirements and a well specified, prioritized list of quality attributes.Be well documented, with at least one static view and one dynamic view.Be reviewed by the system’s stakeholders.Be analyzed for applicable quantitative measures and formally evaluated for quality measures.

Question No 04:(20)

Pair programming is an agile software development technique in which two programmers work together at one work station. 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.

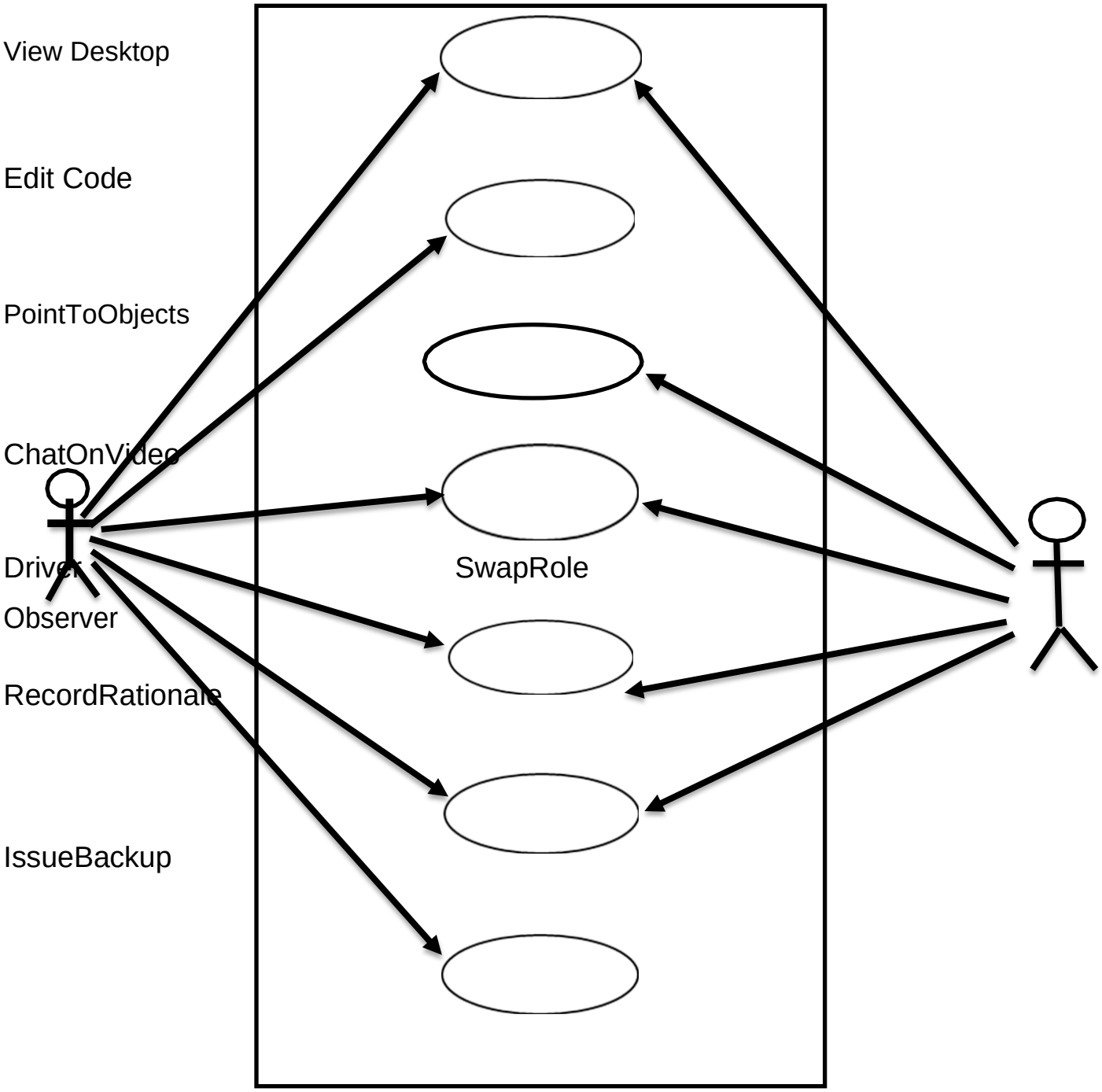
Draw a use case diagram to show all the functionality of the system.

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:

Use-Case Diagram



Assumptions: when the Driver edits code, we assume that the Observer can see the changes in realtime through the ViewDesktop use case, thus there is no arrow pointing back to the Observer for the EditCode use case. A similar assumption is made for the PointToObjects use case, so no arrow points back to the Driver.
we assume that both the Driver and Observer can initiate the ViewDesktop, ChatVideo, SwapRole, and RecordRationale use cases.

Nonfunctional:

Ease of use - the front-end interface must be simple and easy to use.
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 also.
Availability - the system should be available to both programmers all the time.
Portability - the programmers should be able to use the system regardless of what computer and operating system used by the programmers.

Give a prioritized list of design constraints for the system and justify your list and the ordering.
Answer:
Example 1: "**Portability-** 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).

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

Answer is on next page….

Class Diagram