Software Design And Architecture

ID 13126

Muhammad kamal khan

INSTRUCTOR :: Madam Asma Khan

Dept :: BS (SOFTWARE ENGG)

DATE :: 23-6-2020

IQRA NATIONAL UNIVERSITY (PESHAWAR) INU

Q.1:-a) What is Software Architecture? Why is software architecture design so important?

ANS:- Software architecture refers to the fundamental structures of a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations. The architecture of a software system is a metaphor, analogous to the architecture of a building. It functions as a blueprint for the system and the developing project, laying out the tasks necessary to be executed by the design teams. For example, the systems that controlled the Space Shuttle launch vehicle had the requirement of being very fast and very reliable. Therefore, an appropriate real-time computing language would need to be chosen. Additionally, to satisfy the need for reliability the choice could be made to have multiple redundant and independently produced copies of the program, and to run these copies on independent hardware while cross-checking results.

A poor design may result in a deficient product that

does not meet system requirements,

is not adaptive to future requirement changes,

is not reusable.

b)Explain any four tasks of architect?

Ans(b):-

1. Perform static partition and decomposition of a system into subsystems and communications among subsystems.

- A software element can be configured, delivered, developed, and deployed, and is replaceable in the future.

2. Establish dynamic control relationships among different subsystems in terms of data flow, control flow orchestration, or message dispatching.
3. Perform tradeoff analysis on quality attributes and other nonfunctional requirements during the selection of architecture styles.

- For example, in order to increase a distributed system's extensibility, portability, or maintainability, software components and Web services may be the best choice of element types, and a loose connection among these elements may be most appropriate.

4. Consider and evaluate alternative architecture styles that suit the problem domain at hand.

====================

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 03:- Explain ABC Activities?

Ans 04:-

Creating the business case for the system

Why we need a new system, what will be its cost? Time to market, integration with existing systems?

Understanding the Requirements

- Various approaches for requirements elicitation i.e., object-oriented approach, prototyping etc.
- The desired qualities of a system shape the architectural decisions – architecture defines the tradeoffs among requirements.

Creating/selecting the architecture

Communicating the architecture

- Inform all stakeholders (i.e., developers, testers, managers, etc.)
- Architecture's documentation should be unambiguous

Analysing or evaluating the architecture

Evaluate candidate designs

Architecture maps the stakeholders' requirements/needs

Implementation based on architecture

Ensuring conformance to an architecture

Question No 04: (20)

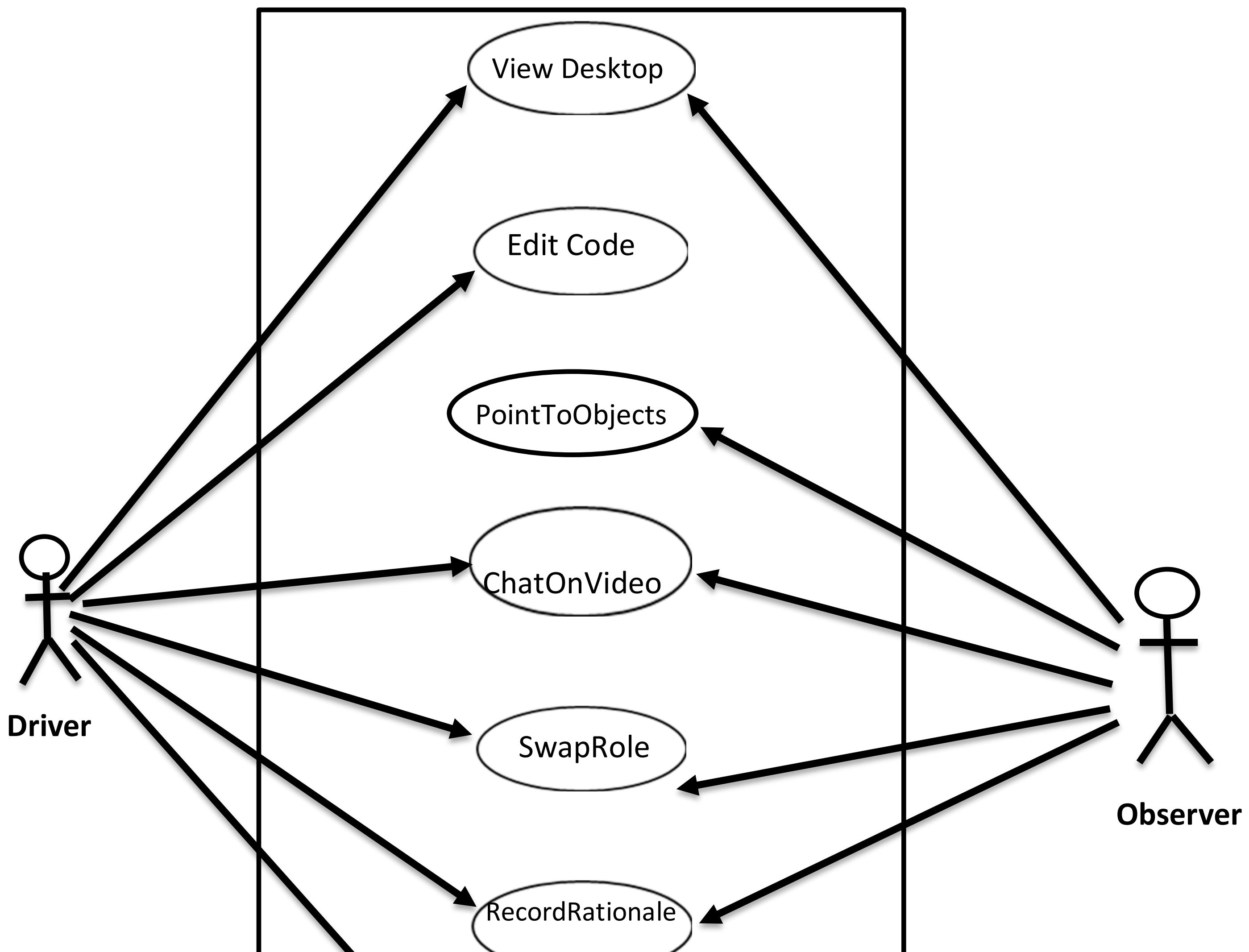
Pair programming is an agile software development technique in which two programmers work together at one work station. One types in code while the other reviews each line of code as it is typed in. The person typing is called the driver. The person reviewing the code is called the observer. The two 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.

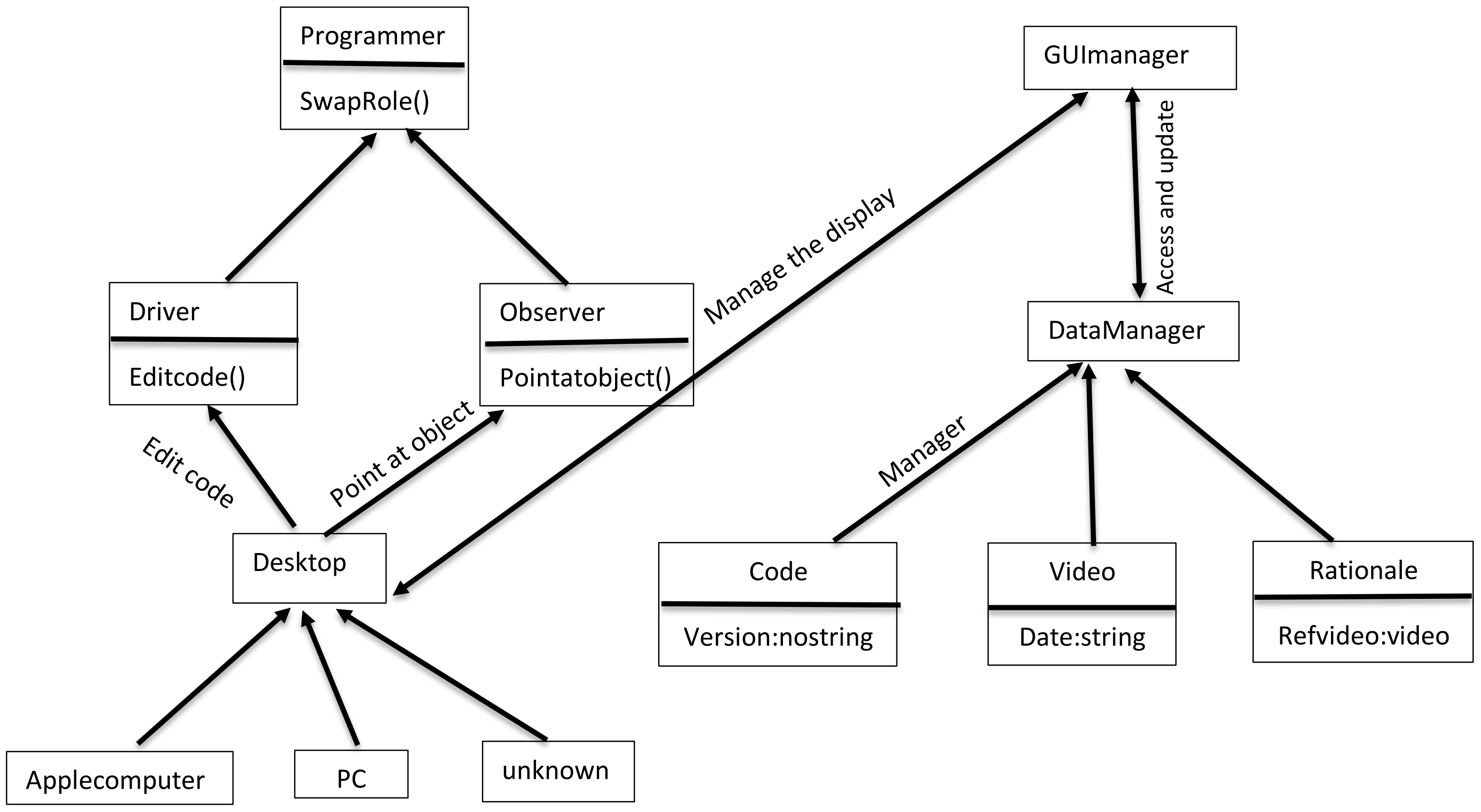
Answer:

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


Programmer
SwapRole()
GUImanager
Access and update
Manage the display
Driver
Editcode()
Observer
Pointatobject()
DataManager
Edit code
Point at object
Manager
Desktop
Code
Version:nostring
Video
Date:string
Rationale
Refvideo:video
Applecomputer
PC
unknown