Name wasim akram

Id 11758

Q1 What is software architecture and why is it important?

- is the description of elements from which system is built, interactions among those elements, patterns that guide their composition, and constraints on the patterns.
- Considers system as a collection of components and their interactions.

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 organization of the system, how the system parts communicate with each other, whether there are some external dependencies, what the guidelines and implementation technologies are, what risks to take into consideration, and much more.

Why is software architecture design so important?

- 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,
  - exhibits unpredictable behavior, or
  - performs badly.

B explain any four task architecture

- Influences
  - System Stakeholders
  - Developing organization
  - Architects' background and experience
  - Technical environment
  - Precautionary measures
  - Know your constraints
  - Early engagement of stakeholders
- 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.
  - Each element's interface encapsulates details and provides loose coupling with other elements or subsystems.
- Establish dynamic control relationships among different subsystems in terms of data flow, control flow orchestration, or message dispatching.
- Consider and evaluate alternative architecture styles that suit the problem domain at hand.
- 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.

Q2 explain Architecture business cycle ABC in detail with figure

The Architecture Business Cycle(ABC)

- 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

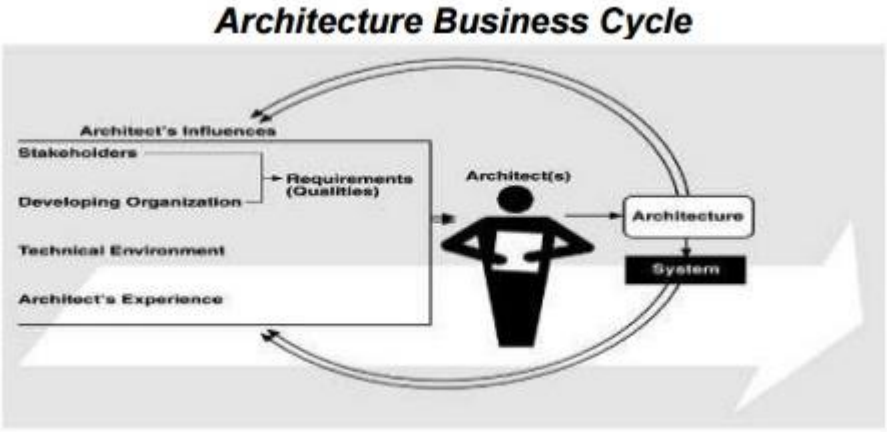
the *Architecture Business Cycle (ABC)*.

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.

The Architecture Business Cycle:

Definition: 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)."



Q3 Explain ABC activities

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

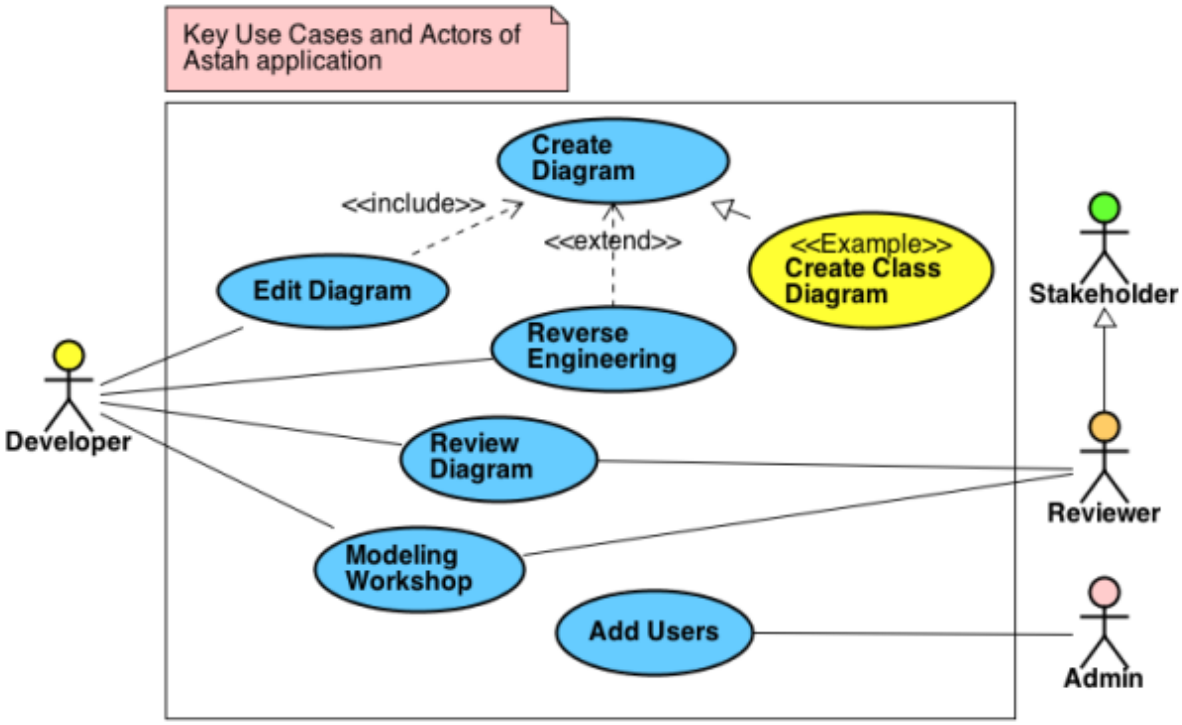
Analysing or evaluating the architecture

- Evaluate candidate designs
- Architecture maps the stakeholders' requirements/needs

Implementation based on architecture

Ensuring conformance to an architecture

Q4



Nonfunctional Requirements

security

reliability

performance

, maintainability

Scalability

- Pair programming can improve overall productivity through the process of collaboration
- Higher quality code as a result of real-time review
- Better designed solutions through shared collaboration
- Better distribution of knowledge

- Greater job satisfaction for the developers
- Faster delivery because solutions to challenging problems are found more quickly

Collective code ownership on team

confident in solutions they came up with during the work.