Assignment#01

Subject: Software Architecture and Design

The University of Lahore

Department of Software Engineering

Submitted By: Ahmed Sohail

Roll# 70067362

Section: S

Q1. Based on your own knowledge, some of the application types discussed in class, explain, with examples, why different application types require specialized software engineering techniques to support their architecture, design and development?

- Costs and frequency of change. Some embedded systems are extremely expensive to change others, must change frequently in response to changing requirements. But these systems are expensive so they are or cost effective.

- The most important 'non-functional' requirements. Different systems have different priorities for non-functional requirements. For example, a real-time control system in an aircraft has safety as its principal priority; an interactive game has responsiveness and usability as its priority.

- The software lifetime and delivery schedule. Some software systems have a relatively short lifetime (many web-based systems), others have a lifetime of tens of years (large command and control systems). Such as ERP systems which are costly and there lifespan is long.

Q2. IEEE Standard 1471 identifies sound practices to establish a framework and vocabulary for software architecture concepts. Explore and discuss various activities of this standard?

According to IEEE Standards 1471 architect is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

IEEE Standard 1471 identifies sound practices to establish a framework and vocabulary for software architecture concepts. In 2000, the Computer Society approved IEEE Standard 1471, which documents a consensus on good architectural description practices. Five core concepts and relationships provide the foundation for the approved IEEE 1471 version: every system has an architecture, but an architecture is not a system; an architecture and an architecture description are not the same thing; architecture standards, descriptions, and development processes can differ and be developed separately; architecture descriptions are inherently multiviewed; and separating the concept of an object's view from its specification is an effective way to write architecture description standards. IEEE 1471 focuses on both software intensive systems and more general systems, such as information systems, embedded systems, systems-of-systems, product lines, and product families in which software plays a substantial role in development, operation, or evolution.

To establish terms and concepts for architectural thinking. We provide a means to talk about Architectural Descriptions within the context of

- System Stakeholders
- Life Cycle
- Uses of Architectural Description

***Architectural Description* (AD):**

- Is a collection of products to document an architecture.
- An AD is addressed to the system's stakeholders to answer their architectural concerns about the system
- An AD is organized into one or more views of the system
- Each view addresses one or more concerns of the stakeholders

Architectural View:

An AD consists of one or more views

- *View*: a representation of a whole system from the perspective of a related set of concerns
- The architectural views are the actual description of the system
- Support multiple audiences each with their own concerns
- Reduce perceived complexity through separation of concerns

Q3. What are the four important attributes that all professional software should have? Suggest four other attributes that may sometimes be significant?

1. Software must be maintainable.

Unmaintainable software is difficult to change to meet new demands. Software should be able to easily evolve, as change is inevitable in a growing business. Making software maintainable would include having a well organized framework, good documentation, clean code.

2. Software must be dependable and secure.

Software should ideally not have many bugs. In the event of failure, software should not cause any damages. Software must be secure. It would be unfortunate if an online banking website had a security flaw that allowed outsiders to access a users account.

3. Software must be efficient.

Software should only use as many system resources as it needs. Wasteful use of system resources can slow processes down and decrease responsiveness. Those systems are helpful in lifesaving equipment's.

4. Software must be acceptable.

It should be compatible with the systems its users use, understandable, and usable. As an example, a web application should be compatible with all browsers. A good user interface would help with understandability and usability.

Suggestions

- Software should be scalable.
- Software should be modern.
- Software should be ethical.
- Software should be original.