
2. *Reference architectures* capture important features of (all) system (i.e., application) architectures *in a given domain*. Essentially, they include everything that might be in an individual *generic application architecture* within that domain, but it is very unlikely that any individual application architecture would include all the feature in the (more inclusive) reference architecture. (The main purpose of reference architectures is to evaluate and compare design proposals, and to educate people about architectural characteristics, in a given domain.)

3. a. They argued that using UML for architectural descriptions in a “loose and informal way” (i.e., abstractly) was a bad thing.
   b. Sommerville disagreed. He feels that the UML was designed for describing object oriented systems and, at the architectural design stage, you often want to describe systems at a higher level of abstraction. Object classes are too close to the implementation to be useful for architectural description.

4. The isolation (“hiding”) of potentially changeable design decisions (e.g., in objects, functions, procedures, etc.) to minimize the impact of change.

5. A deployment diagram shows how software components are physically deployed on processors. (I.e., it shows the hardware and software in the system, and the middleware used to connect different components in the system.) Essentially, it defines and documents the target environment.

6. c

7. d

8. a. **Problem tracking**: It allows the reporting of bugs and other problems and allows all developers to see who is working on these problems and when they are fixed.
   b. system integration


10. **Servicing**: The system remains useful, but the only changes made are those **required to keep it operational** (i.e., bug fixes and changes to reflect changes in the software’s environment). **No new functionality is added.**

   **Evolution**: The system is **in operational use** and is evolving **as new requirements are proposed and implemented.**

11. b

12. a. One should think of software engineering as a spiral process with requirements, design, implementation, and testing going on throughout the lifetime of the system. Post-delivery “evolution” really just represents additional iterations of the initial development process.
12. (cont’d)
   b. The first stage of change implementation may involve program understanding. (This involves understanding how the program is structured, how it delivers functionality, and how the proposed change might affect the program.) It is most likely to be critical if those responsible for change implementation are not the original system developers.

13. a.
   ![Diagram]

   b. This would only be possible if the test case(s) written and implemented failed to reveal that the increment of functionality had not yet been implemented in the program.

14. a. true; b. false; c. true; d. true

15. c