Discussion Forum Questions

* Discuss the role of actors in a use case. You may wish to structure your answer around some of the questions below. Be sure to explain the reasons for each response.
  + Is an actor internal or external to a system?
  + Does an actor have a single well-defined purpose or many different purposes?
  + Does an actor have to be a person?
  + How is a use case related to the concept of an actor.
  + Is it necessary for every actor in a system environment to participate in a use case?
  + How do you determine who (or what) should be an actor? How many actors should there be?
  + What distinguishes a primary actor from a supporting actor?
  + What is the relationship between a stakeholder and an actor?
  + Is the goal associated with a use case always determined by the actor?
  + What techniques are helpful when trying to identify actors?
* One of the main problems with constructing class diagrams is deciding which classes are needed. There are two general approaches. In the top-down approach, classes are found first on the basis of general understand of the subject matter. Top-down analysis is about finding a basic structure of classes that the bottom-up analysis, which is more detailed, can build upon. The (simplified) question is: Which information or domain concepts can be of use for my IT system?

Domain knowledge, verbal descriptions of the area of application, and user representatives are important sources of information. In this way, a basic structure of classes can be found for most IT systems. In bottom-up analysis, classes are found mainly on the basis of the inputs and outputs of the IT system. The question is: What information is needed for the individual inputs and outputs of the IT system?

Here, the classes that were found during top-down analysis serve as the basis to find these classes. Already existing inputs and outputs, for instance, screen forms and paper forms are important sources of information.

Which approach would you suggest that an organization follow to generate the most accurate class diagrams? If the goal were to create the simplest class diagrams, would your answer be the same? Explain both of your answers.

* One of the classic tech support stories is about the befuddled user who calls tech support and says: "The screen says to press any key to continue. Where's the ANY key?" What's wrong with "Press any key to continue."? Find one or two other examples of dialog boxes or menu items that display the same problem. Then, describe what could be done to alleviate the problem.
* In database management courses, we teach students to normalize all relational databases to at least third normal form. However, in practice, some tables are purposely not normalized. Why do you think this is so? What are the pros and cons of such a strategy? What would you advise if you were a database consultant?
* The switch from structured analysis and design to object-oriented analysis and design was fairly easy for database designers because there is a connection/relationship between classes and entities. How are classes and entities the same? How are they different?
* We have been using client/server architectures for more than 20 years now. It seems that client/server organization was accepted with little controversy. However, it does have some disadvantages. Discuss the advantages and disadvantages of client/server system organization. Do you think the advantages outweigh the disadvantages? Has our faith in this architecture been misplaced all these years or are we well served by favoring client/server organization over centralized computing architecture?
* Many of today’s Web-based systems must interact with a corporate database. (For example, consider the database that a retail site must use to store catalog data.) The challenge of such an arrangement is that the database must be accessible to the Web server and accessible to internal database users, yet still remain secure. How does the need for security affect the architecture of a system design that uses such a back-end database?
* How do you know when "enough is enough"? What should you look for when determining when software is ready to go into production? What indicates that enough testing has been done?
* Most of us would probably agree that it is very easy to ship a product or put software into production use that hasn't been tested enough. In contrast, is it possible to test too much? Why?
* One of the safest implementation techniques is parallel implementation. Is this always appropriate? In what circumstances would parallel implementation be inappropriate/impractical/unnecessary?