**1. Describe the goal of the Design Phase of the software lifecycle**

Implement all of the explicit requirements specified in the requirements model, and it must accommodate all of the implicit requirements desired by the stakeholders

Design creates a representation or model of the software, the design model provides details about software architecture, data structures, interfaces and components that are necessary to implement the system.

Create a software design that is readable, understandable, which will serve as a guide for those who generate code, who test, and support the software

Provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective

Produce a model or representation that exhibits firmness, commodity, and delight, through diversification and convergence

2. **What documents are produced during the design phase? What do the documents contain? which UML diagrams do they use? Who reads the documents, and for what purposes do they read the document?**

Work product – a design model that include Architectural document interface, Critical Priority analysis, component-level, and Implementation plan，deployment representations, Performance Analysis, Test Plan.

UML diagrams

Interaction diagrams：UML artifact that considers the responsibilities；Illustrates use case realizations；Illustrate how objects interact via messages；2 types: Sequence; Communication

System sequence diagram **–** shows the system events (operations) identified by each use case

Activity diagram – represents workflows of stepwise逐步 activities and actions

The architectural team can now expand upon the information established in the requirement document. Using the typical scenarios provided from the requirement document, performance trade-offs can be accomplished as well as complexity of implementation trade-offs. The software teamassesses the design model in an effort to determine whether it contains errors, inconsistencies, or omissions, whether better alternatives exist, and whether model can be implemented within the time and schedule constraints

**3. What do we mean by** **fitness of purpose of a design**

Design quality is all about fitness of purpose.

We need to know what that purpose, what function must it perform, what other prosperities must it have (e.g modifiability, reliability, usability), what users needed, needs change.

4. **Explain the terms:** **modularity, (class) interface, cohesion, coupling**

Modularity模块性: It property of a system that has been decomposed into cohesive and loosely coupled modules it is the most common manifestation of separation of concerns. Effective modularization is accomplished by maximizing cohesion and minimizing coupling. Software is divided into separately named and addressable components. Development can be more easily planned; software increments can be defined an delivered; changes can be more easily accommodated; testing and debugging can be conducted more efficiently, and long-term maintenance can be conducted without serious side effects.

User interface classes: define all abstraction that are necessary for human computer interaction. In many cases, HCI occurs within the context of a metaphor, and the design classes for the interface may be visual representations of the elements of the metaphor. Description of the actions that an object can do

Cohesion: is an indication of the relative functional strength of a module. How strongly related and focused the responsibilities of an element are. A cohesive module performs a single task, requiring little interaction with other components in other parts of a program. the degree to which communication takes place among the module’s elements. Cohesive module should do just one thing: the degree to which communication takes place among the module’s elements; It collaborates and delegates.

Coupling: is an indication of the interconnection among modules in a software structure. Coupling depends on the interface complexity between modules, the point at which entry or reference is made to a module and what data pass across the interface. In software design, we should strive for the lowest possible coupling.

5. **Explain information hiding**

Modules should be specified and defined so that information (algorithms & data) contained in one module are not accessible to other modules that do not need that information. Its use is very beneficial when modification/upgrades are required during system testing or maintenance. Hiding implies effective modularity achieved by defining a set of modules that communicate with one another only that necessary info to achieve software function

**6. Explain the concept of design for change**

Software design changes continually new methods, better analysis, and broader understanding evolve.

Design for change indicates a change in perspective that is necessary when considering the development of a highly tailor able application. Software development is often perceived认为 as producing a solution for a problem. Design for change perspective puts the support of a work practice in the focus, that is expected to change in anticipated as well as in unexpected意外 ways. The changes that can be foreseen can be taken into account during development, so that the application can be easily extended or that part of the application can be implemented in a way that allows for tailoring.During the lifetime of a useful program, its users’ requirements change and the code changes to track the requirements. Code needs to be designed in a way that allows for these changes to be done efficiently while reducing the potential to introducing new faults.

**7. What is issue-driven design? How does it relate to design rationale? How do they both relate to the design cycle and the** **major design decisions?**

Design issue: Describe the architectural design issues that are to be addressed. Design issue has alternative solutions. Solutions have pros and cons. Weigh pros and cons to make decision. Incorporate selected solution to resolve issue.

State the approach you have chosen to address the design issue.

Draft design has alternative solutions. Review/Assess/ Evaluate design-discover issues. Resolve design issue = make decision。

**8. What is a** **construction plan? Can you have a construction plan before you have a design?**

Construction Plan: must have plan to develop / implement/ construct the software. Based on dependencies between design components.

Construction plan determines schedule of implementation and test, skills required by developers, number of developers needed, when developer are needed, work tasks for developers.

No, u cannot.

**9. Is OO development** **seamless无缝的? Why or why not?**

OOP is not seamless, because it may have a “business domain” gap. It either does not provide significant benefits for business applications, Or that how to use it for business applications is poorly documented and/or poorly understood. Not every software object is an OOA entity

**10. What is the** **representation gap?**

The representational gap is how different is the domain logic of an application from its design. Having a low representational gap means that the design has very few difference with the domain logic it represents.Gap between the mental representation based on business analysis and requirements (domain model) and actual software components (class diagram).

**11. Following Jacobson's terminology, which kind of objects represent things in the** **domain model?**

Larman GRASP Patterns: GRASP is general responsibility assignment software patterns guidelines for how to jump the representation gap.