# Section 2: *What is Software Architecture?*

## Question 1

Describe how a system’s decomposition into services is driven by the system’s features / requirements.

How can this decomposition be used to determine that we have full coverage of the requirements in the system’s design?

### Answer

The decomposition of the system’s features / requirements into services. Each service is ‘responsible’ for the implementation of at least one, and likely several of the system’s requirements. These responsibilities are met by the services (methods in implementation) each service makes available to other system services and to clients of the system (i.e. users).

When we examine the system’s design we should see that each of the requirements has been assigned to at least one service for implementation. Any requirements not covered by a service will be missing from the ultimate design and implementation of the system.

## Question 2

Explain these statements in a few paragraphs:

The architecture presents a broad and shallow view of the system.

The architecture should present the what, but not the how of the system’s design.

### Answer

The architecture provides a broad and shallow view of the system’s design. The architecture covers all aspects (is broad) of the systems design i.e. a design that facilitates all of the services required by the requirements. However, the architecture can only provide a description this lacks detail (is shallow). A detailed design of the features identified in the architecture is prepared by the individual developers assigned responsibility for the implementation of these features i.e. narrow and deep.

The architecture should present the what, and not the how of the system’s design. The architecture described what the system does though the components and interfaces presented in the architectural design. But how the interfaces are implemented is described by the individual designs of the components implementing the interfaces by the developers.

## Question 3

What is Miller’s Law?

In the context of Miller’s Law, what is the purpose of decomposing a system into sub-systems / services / components?

### Answer

Miller’s Law describes how the average person can only maintain 7±2 ideas, tasks, objectives, etc. in short-term memory.

Component decomposition describes how the designer decomposes (divides) complex problems, systems, etc. into smaller, less complex units. If properly decomposed (i.e. cohesively and loosely coupled), the each of these simpler services or components can be independently designed and constructed reducing the conceptual load on the individuals.

## Question 4

Provide a description of Component (Class) Cohesion.

Describe why the decomposition of a system into components should be driven by cohesion.

Note: Cohesion was not covered in the lecture. If you do not remember from 3354 (Software Engineering), you should research.

### Answer

Cohesion is a property we can assign to a component that describes how closely related the features implemented by the component. A cohesive component is ‘focused in its intent’. A component that implements unrelated features (i.e. customer and product management) has less cohesion than the two components that would result if the same set of features were decomposed into separate components. A component with high cohesion is ‘better’ i.e. the component is easier to reuse, understand, modify, etc.

When we decompose a system’s design into components, each component should maintain an individual set of focused responsibilities which is the definition of high cohesion. If we find that a component in the evolving design is not focused, too large, etc., we should further decompose this component into two or more components based on the responsibilities assigned to the original component.

## Question 5

Describe how client - server component design can increase reusability in the system’s design. Hint: Consider multiple services requiring the same client service.

### Answer

If we include the implementation of the service in a single component we will have embedded the service’s implementation in the same component requiring the service. What if the service is needed by a second or third component in the design? Do we duplicate the service in each of the component requiring the service? Duplicating code is always a poor decision as it increases the size of the code base and makes maintaining (bug fixes, etc.) the service error prone.

When we decompose a component and spit the client from the service (server), we can reuse the service in other situations i.e. we make the service available to other clients in the system’s design. This produces a system that is easier to maintain and opens the possibility of reusing the service in other systems.

## Question 6

Briefly describe the characteristics of applications the three-tier architecture should be applied to.

Name each tier and describe the types of responsibilities each tier implements.

### Answer

The three-tier architecture describes how application / system designs should be partitioned along three types of services that are usually required by a web-application, or any application that provides both a user interface and uses a database to maintain persistent information.

These tiers are:

Presentation: Components that implement GUI / user interface services i.e. services that present information to the user.

Business (Application): Components that implement application / business logic including data validation, workflows, report generation, controllers, and others. Sometimes called the Application tier.

Data (Service): Components that provides the services needed to persist (save and retrieve) application data to/from a database server. This tier also contains services that act as proxies to external services used by the system. Sometimes called the Data or Persistence tier.

## Question 7

Which of the hierarchical views (System, Service, or Component) best describes the follow aspects of the system’s design?

1. The decomposition of the system’s overall functionality into categories of related functionality. Service
2. Identifying processors and other hardware that make up the system’s design. System
3. The classes and relationships between classes that make up the system’s design. Component
4. Most useful when communicating the system’s design to a non-technical member of the team. System
5. Most useful when communicating the system’s design to the developers for implementation. Component

### Answer

Given in questions.

## Question 8

Describe what is meant by the statement: Components of a software architecture should provide a black-box view of the services they implement.

How does a component make its publicly accessible services know to its clients?

When is a White-Box view of the system designed?

### Answer

A black-view of a component means that external components cannot see inside the target component i.e. external components cannot see the implementation of the services provided by the target component.

Each component should present its “interface’. Interfaces is the definition of the methods (name, arguments, return type) that the component is responsible for implementing.

The white box view of the system is created when individual components are designed and implemented during the design and construction phases of the project’s execution.

## Question 9

What is the name of the mechanism we use to describe the individual services provided by a component to its clients?

How does this mechanism describe the component’s service?

Hint: The mechanism is typically used to decouple the definition of a service from the service’s implementation.

### Answer

We use an ‘interface’ to describe the services a component provides to its clients. Understand Why: The interface defines the methods that a component / class makes available to its clients. By describing only the interface, clients can be designed without coupling to the specific class that implements the interface. This reduces coupling in the system’s design.

## Question 10

Which of the following best describes a ‘uses’ relationship between components.

1. Client-Server
2. Shared Information Repository
3. Asynchronous Messages
4. None of the above

### Answer

Client-server describes the situation where the client uses the server to request a service provided by the server.