3 Architecture - exam questions

You should know the answers to these questions

What's the role of a software architecture?

Software architecture provides a description of the system's structure, including its components and connectors. It maps requirements (both functional and non-functional) onto the system structure, helping developers manage complexity and scale while enabling quality attributes like maintainability and reliability.

What is a component? And what's a connector?

- **Component:** An encapsulated part of a software system with a designated interface, such as a class, module, or subsystem.
- **Connector:** A connection or relationship between components. They can be static (e.g., imports, dependencies) or dynamic (e.g., method calls or data streams).

What is coupling? What is cohesion? What should a good design do with them?

- **Coupling:** The measure of the strength of the connection between components.
- **Cohesion:** The degree to which elements within a component work together to fulfill a specific function.
- **Good Design:** Minimize coupling while maximizing cohesion to reduce dependencies and enhance modularity.

What is a pattern? Why is it useful for describing architecture?

- **Pattern:** The essence of a solution to a recurring problem in a specific context.
- **Usefulness:** Patterns document established solutions, provide a shared vocabulary for design discussions, and help manage tradeoffs in architectural decisions.

Can you name the components in a 3-tiered architecture? And what about the connectors?

- **Components:**
  - **Application Layer:** User interface and application logic.
  - **Domain Layer:** Problem domain, usually modeled as a set of classes.
  - **Database Layer:** Data management, often using a relational database.
- **Connectors:** APIs, method calls, or data queries connecting layers.

Why is a repository better suited for a compiler than pipes and filters?

A repository (or blackboard) architecture allows different components (e.g., lexical analyzer, syntax analyzer) to operate independently while sharing a common data structure. This flexibility makes it better suited for a compiler compared to the linear data flow of pipes and filters.

What's the motivation to introduce an abstract factory?

Abstract factories help manage families of related objects without specifying their concrete classes, reducing tight coupling and making it easier to switch configurations or create platform-independent designs.

Can you give two reasons not to introduce an Adapter (Wrapper)?

1. **Overhead:** Adapters may introduce performance and maintenance overhead.
2. **Limited Use Case:** If adaptation is needed for a single, simple scenario, creating an adapter can add unnecessary complexity.

What problem does an abstract factory solve?

It addresses the issue of tight coupling with specific concrete implementations by providing a way to create families of related objects through abstract interfaces.

List three tradeoffs for the Adapter pattern.

1. **Adapting Scope:** Adapting an entire class hierarchy may be more challenging than adapting a single class.
2. **Evolution:** Adapters may require significant updates if the underlying classes change.
3. **Performance:** Indirect calls through adapters may impact performance.

How do you decide on two architectural alternatives in scrum?

By running a **spike**, which includes prototyping both alternatives and conducting tests (e.g., speed, load, security). The results are documented and shared with the team to decide collectively.

What's the distinction between a package diagram and a deployment diagram?

- **Package Diagram:** Focuses on logical decomposition of the system into packages (modules, classes).
- **Deployment Diagram:** Shows the physical runtime layout of components on hardware nodes.

Define a sensitivity point and a tradeoff point from the ATAM terminology.

- **Sensitivity Point:** A property critical to achieving a quality attribute, e.g., performance depending on encryption levels.
- **Tradeoff Point:** A situation where conflicting sensitivity points exist, e.g., balancing security (encryption) with real-time performance.

You should be able to complete the following tasks

Take each of the patterns and identify the components and connectors. Then assess the pattern in terms of coupling and cohesion. Compare this assessment with the tradeoffs.

- **Example (Observer Pattern):**
  - **Components:** Subject and Observer.
  - **Connectors:** Attach, detach, and notify methods.
  - **Coupling and Cohesion:** Low coupling between subject and observers; cohesion within the subject or observer.
  - **Tradeoffs:** Increased complexity due to notifications; excessive updates if not carefully implemented.

Can you answer the following questions?

What do architects mean when they say "architecture maps function onto form"? And what would the inverse "map form into function" mean?

- **Mapping Function onto Form:** Translating requirements into system structure.
- **Mapping Form into Function:** Designing structure first and then determining the functionality it supports.

How does building architecture relate to software architecture? What's the impact on the corresponding production processes?

Both manage complexity and ensure alignment with requirements. Software architecture is more flexible and allows iterative improvement, impacting agile processes with emergent design and intentional architecture.

Why are pipes and filters often applied in CGI-scripts?

CGI-scripts frequently process data streams with discrete steps, making pipes and filters ideal for their flexibility and reusability.

Why do views and controllers always act in pairs?

In the Model-View-Controller pattern, the controller translates user actions into changes to the model and updates the view. Their interaction ensures consistency.

Explain the sentence "Restricts communication between subject and observer" in the Observer pattern

Observers register with a subject and are notified of changes anonymously. This limits direct dependencies and ensures decoupling.

Can you explain the difference between an architecture and a pattern?

- **Architecture:** The system's high-level structure, including components and connectors.
- **Pattern:** A reusable solution for a recurring design problem within an architecture.

Explain the key steps of the ATAM method?

1. Present the architecture.
2. Identify quality attribute goals.
3. Analyze architectural decisions.
4. Highlight risks, sensitivity points, and tradeoff points.

How can you balance emergent design with intentional architecture?

Combine agile practices (emergent design) with upfront planning (intentional architecture) to avoid waste while maintaining flexibility.

What happens when your team goes outside the boundaries of the guardrail?

Going beyond guardrails requires additional discussions, approvals, or adopting/changing new standards to mitigate risks.

How would you organize an architecture assessment in your team?

Conduct reviews early, use the ATAM method to identify risks, and evaluate tradeoffs. Ensure documentation and team consensus.