Object-Oriented Methodology HW 05

2024 Fall Semester

21 CST H3Art

Short answer questions

Based on the system sequence diagram and domain model from your Homework04, select one of the system operations in the system sequence diagram and write an operation contract for it.

I select the `authorizePurchase(cardInfo)` operation from the System Sequence Diagram, specifically the 6th scenario—where the system authorizes a purchase.

Operation Contract: authorizePurchase

- Operation: `authorizePurchase(cardInfo)`
- Cross Reference: Use Case: Buy a Product
- Preconditions:
  - Valid credit card information (`cardInfo`) must be provided by the customer.
  - The system must be online and able to access the credit authorization service.
- Postconditions:
  - If the credit card is valid and has sufficient funds:
    - The purchase is authorized.
  - Otherwise:
    - The purchase is denied, and an authorization failure is recorded.

How does a strict layered architecture differ from a relaxed layered architecture?

Strict Layered Architecture: In this architecture, each layer can only communicate with the layer directly below it (or above it, depending on the direction). This enforces a clear separation of concerns and dependencies.

Relaxed Layered Architecture: This allows for some degree of communication between non-adjacent layers. While it still maintains the concept of layers, it can lead to more flexibility and easier modifications, but at the cost of potentially increased coupling between components.

What's the connection between SSDs, system operations, and layers?

SSDs illustrate interactions between external actors and the system over time, helping us understand the dynamic behavior of system operations. In an SSD, each operation represents the system's response to external requests, often corresponding to different layers in the system architecture. For example, high-level operations may trigger low-level implementation details within a layered architecture. SSDs clarify the dependencies and information flow between layers, ensuring that operations are implemented appropriately within the architecture. Additionally, they help identify potential issues in the system, such as performance bottlenecks or unnecessary coupling between layers.

What is a key advantage of sequence diagrams over communication diagrams?

A key advantage of sequence diagrams is their explicit representation of the temporal order of events. They use vertical timelines to show the sending and receiving of messages, allowing developers to understand the flow and dependencies of operations intuitively. This clear temporal sequence helps identify potential concurrency issues and interaction delays, enhancing the system's predictability. Sequence diagrams are particularly suited for complex interaction scenarios, as they can clearly depict collaboration among multiple objects. In contrast, communication diagrams focus more on static connections between objects and lack the emphasis on timing. Thus, sequence diagrams are often more effective for designing and analyzing complex systems.

How do communication diagrams benefit Agile Modeling practices?

Communication diagrams offer several benefits in Agile modeling practices. Firstly, their simplicity allows teams to quickly iterate and adjust designs. In Agile development, requirements and designs often need to change frequently, and communication diagrams enable teams to discuss and modify designs flexibly. Additionally, communication diagrams emphasize relationships between objects rather than the sequence of events, helping development teams capture the overall structure and interactions of the system early on, which promotes collaboration and communication among team members. Their ease of understanding allows for swift communication of design ideas, improving overall team efficiency and responsiveness. This quick feedback and adjustment capability is crucial in an Agile environment.