An-Najah National University

Course: Advanced Software Engineering

Semester: First 2025/2026

Refactored the Event Processing System Refactoring

Design Decision Document

Instructor: Dr. Mustafa Assaf

Prepared By

Fathi Al Heelo

12323885

Date: DEC 12,2025

Why was each design pattern chosen?

1.1 Strategy Pattern

The system has maluti class some behavior and events sash as (User ,systeam , and security events) but each event has unique execution logic, instead of using if statements, we can apply the stratigy pattern his pattern assist for encapsulate job specific behavior in separate strategy classes and allow adding new job without modifying existing code(applying OCP) also improve clearlarity and reduce duplicate.

1.2 Proxy Pattern

EventProcessorProxy intercepts calls to event processor to perform: Permission Validation and execution time measurement and logging, and monitoring also observer notification.

We use the proxy pattern because it allows adding cross cutting behavior without editing processing, also maintaining SRP.

1.3 Prototype Pattern

Event creation may requirw setting nultiple default properties such as encryption,compression ,or metadata . so a prototype allows cloning them instead of rebuilding every time.

This improves performance and reduces repeated work.

1.4Registry Pattern

Stores all templete in one place and returens clones when needed .

This avoids duplication and make tempelte clean and scalable.

1.5Observer Pattern

Logging and dashbord updates should not be tightly coupled with the core event processing logic .

The observer pattern was used to notify different observers such as logger and dashboard ,whenever an event is processed. This allows adding or removing observers without modifying the main processing code and imporves separation of concerns .

1.6Decorator pattern

Events may require optional transformations such as encryption , compression , or metadata . the decorator pattern allows applying these transformations dynamically without using complex conditional logic or creating many subclasses .this follows the OCP and keeps the code flexible and clean

1.7Connection Pool Pattern

The system stores events in a database , and creating a new connection for each event is inefficient . the Connection Pool pattern was used to reuse database connections, improving performance and resource management , especially when processing many events.

Alternatives Considered

Using if/else logic was rejected because it violates the OCP and makes the processor difficult to maintain.

Hardcoding encryption ,compression ,and metadata inside the processor was avoided due to code duplication and SRP violations ; the decorator paatern offers a cleaner solution crating event using constructors each time was considered inefficient while the creating events using constructors ech time was considered inefficient ,while the prototype pattern provides faster and consistent object creation .

What alternatives you considered?

For example, when using if/else instead of a strategy, this is rejected because it makes processor class large and hard; the strategy pattern solves that and applies OCP.

And Hardcoding encryption , comperssion , and metadata logic inside the processor was rejected

because it causes code duplication and violates the SRP and the decorator Pattern provides a cleaner and more flexible soultion.

Creating events using constructors every time was also considered inefficient , and prototype pattern offers faster object creation through cloning and ensures consistent configuration .

How the patterns interact?

The prototype and registary patterns handle event creation the proxy pattern validates events and triggers observer notifications before delegating processing .

The decorator pattern transforms event data dynamically .

The strategy pattern executes event type specific behavior and the connection pool manages database resources transparently ,together , these patterns form a clean processing pipeline where each component has a sigle responsibility.

How your architecture improves scalability, flexibility, and maintainability?

Scalability ConnectionPool and prototype cloning reduce resource overhead , allowing the systeam to process more events efficiently.

Flexibility can add any event types and templetes or proxy

Behaviors and transformations also observers ,can be added easily without modifying base classes.

Maintainability clear separation of responsibilities and solid principles make the code easy to read and test and update.