Abstract Driven Design (ADD) Principle

Principle Author: vu.minh.quang@outlook.com

Abstract Driven Design (ADD) is a layered architecture approach that separates concerns between the abstract domain, business application, concrete implementation, and contract descriptors. This methodology fosters a systematic way to design and implement software systems by emphasizing abstraction and separation of concerns.

Key Layers of ADD:

1. **Contract Descriptors Layer**: (Abstraction for interacting with the outside)
    - Defines external contracts and interfaces for the system.
    - Includes Data Transfer Objects (DTOs), feature interfaces, and outer events.
2. **Abstract Domain Layer**: (Abstraction for inner interaction)
    - Contains core business logic and domain objects.
    - Independent of any specific technology or framework.
    - Defines entities, interfaces, abstract classes, and inner events.
        - Entities: only have the fields (which could NOT set value from outside), & the validation rules on the fields, like the atomic action, example:

    ```
    public void ApplyTransaction(decimal amount)
    {
        if (Balance + amount < 0)
        {
            throw new InvalidOperationException("Insufficient
    funds");
        }

        Balance += amount;
    }
    ```

        - Inner events: can be called domain events, the events which use internally between Business Application & Concrete Infrastructure
3. **Business Application Layer**:
    - Implements application features (only implement what is described from the contract layer). Do not implement any functions from the service's interfaces from Domain.
    - Handle outer events.
    - Focuses on specifying behavior and functionality.

- Handles conversion between DTOs and Entities.
- Implements outputs such as Minimal APIs or controllers.

4. **Concrete Infrastructure Layer**:
    - Provides concrete implementations of services, repositories, inner event publishers, and integrations.
    - Integrates with databases, message queues, and other infrastructure components.
    - This layer might have many versions (for example: version for Postgres, version for Mongo, etc...). In theory, at the beginning, the system might run without problem with a mock infrastructure layer, which do nothing, only return default values.
5. **Program Layer**:
    - This is the layer for exe file.
    - Reference to Business Application Layer & Concrete Infrastructure Layer.
    - Define dependency injection.
    - Do any tasks for initialize the application.

Suggested Writing Order:

1. **Contract Descriptors Layer**
2. **Abstract Domain Layer**
3. **Business Application Layer**
4. **Concrete Infrastructure Layer**

Summary

The **Abstract Driven Design (ADD)** principle is a structured approach to software architecture that emphasizes clean separation of concerns across different layers. By clearly delineating abstract and concrete components, ADD aims to enhance maintainability, scalability, and clarity in software development.