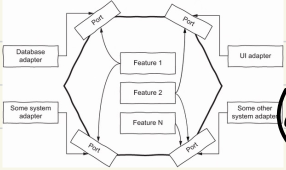
Chapter 7. Designing for Testability

1. Infrastructure code vs domain code

domain → business rules, logic, entities, services

infrastructure → handles external dependency

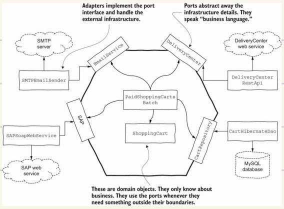


hexagonal architecture

port ≡ domain

adapter ≡ infrastructure

↳ interface implementation


Adapters implement the port interface and handle the external infrastructure.

Ports abstract away the infrastructure details. They speak "business language."

These are domain objects. They only know about business. They use the ports whenever they need something outside their boundaries.

2. Dependency injection & Controllability

controllability → dependency injection

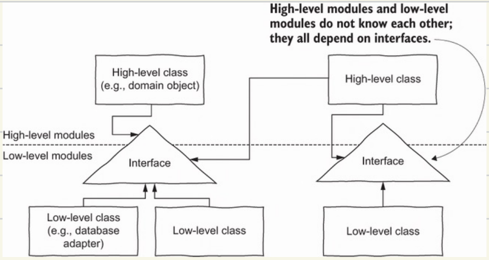
high-level modules (business) → independent of low-level modules

details (implementations) → should depend on abstractions (interfaces)

3. Observability

assert the behavior of the functionality

- Assertions facilitation methods
  ↳ field getters
- Void method observability
  Mockito → Argument Captor → object assertions


High-level modules and low-level modules do not know each other; they all depend on interfaces.

4. Dependency — class constructor vs. method parameter

class constructor ↳ easy stub

complexity on this level

complexity in higher level ← still needs parameter

method parameter ↳ implementation

testing

5. Testability

-> cohesion

-> coupling

-> complex conditions

-> private methods

-> static methods, singletons

-> hexagonal architecture