Introducing Design Patterns

Ridi Ferdiana | ridi@acm.org
Version 1.0.0

Agenda

- What is Design Patterns
- Why we need it ?
- How we implement it on our codes

Design Patterns

- Design Patterns: Common problems and a core solution that can be applied into programming

> Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use the solution a million times over, without ever doing it the same way twice.

-Christopher Alexander

Analogy: Roof Patterns

Part of Design Patterns

- Pattern Name
  - i.e. Observer
- Problem that want to be solved
  - How to simplify complex API
- Solution
  - Core elements and interaction
- Consequences
  - High Memory issues

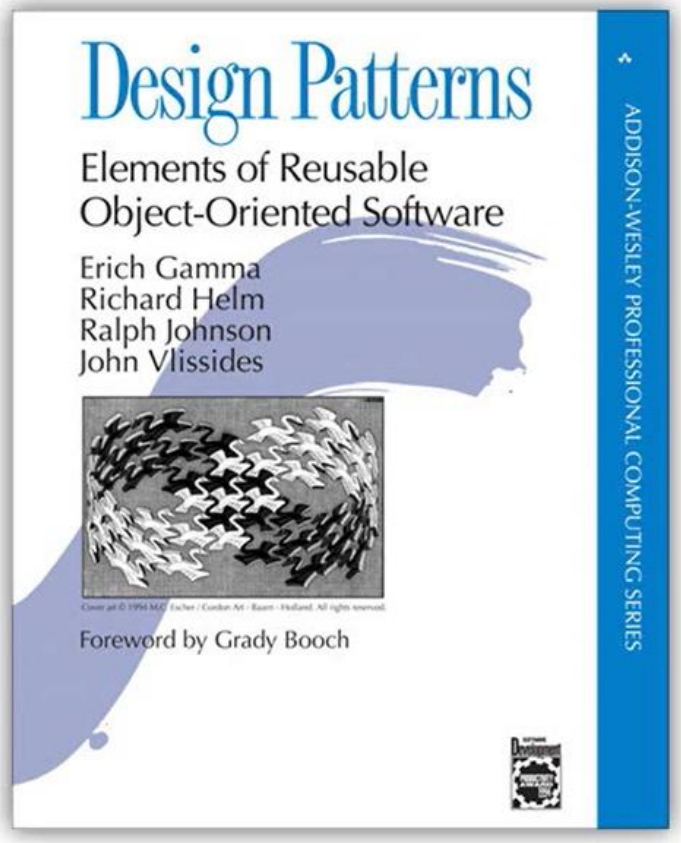
Design Patterns Catalogue

- 28th edition

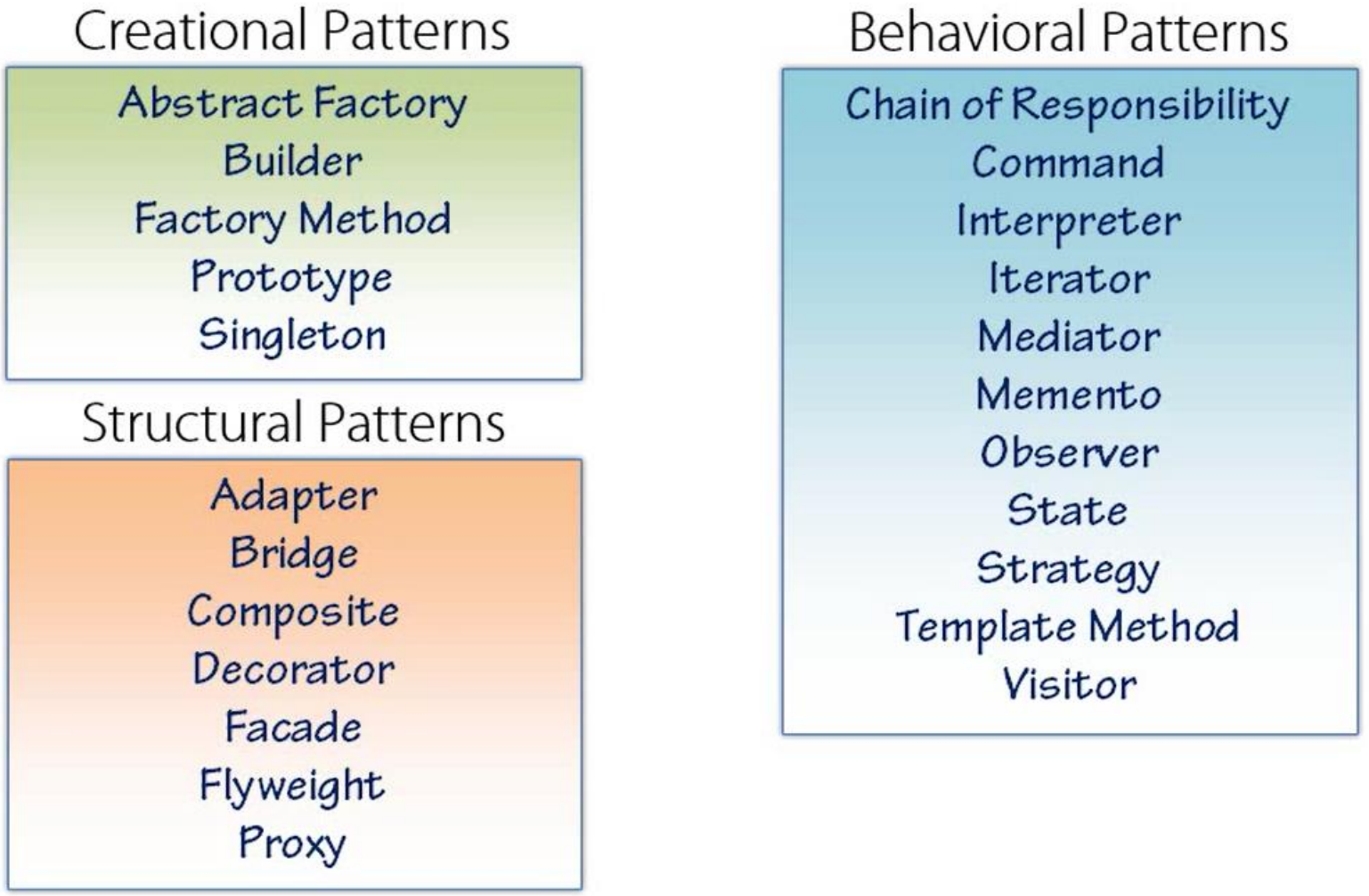
  - **Erich Gamma**
  - **Richard Helm**
  - **Ralph Johnson**
  - **John Vlissides**

Design Patterns:
Elements of Reusable
Object-Oriented Software

ISBN: 978-0-201-63361-0

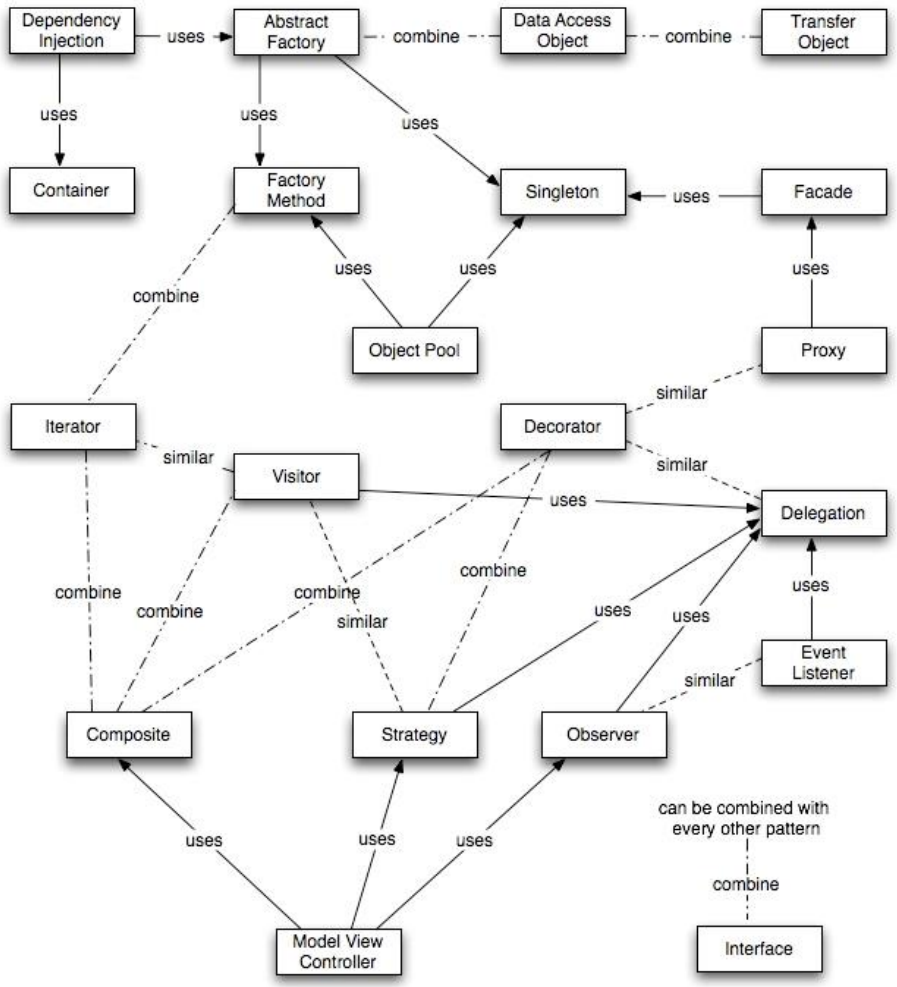


Gang Of Fours Catalogue



Why We Adopt Design patterns

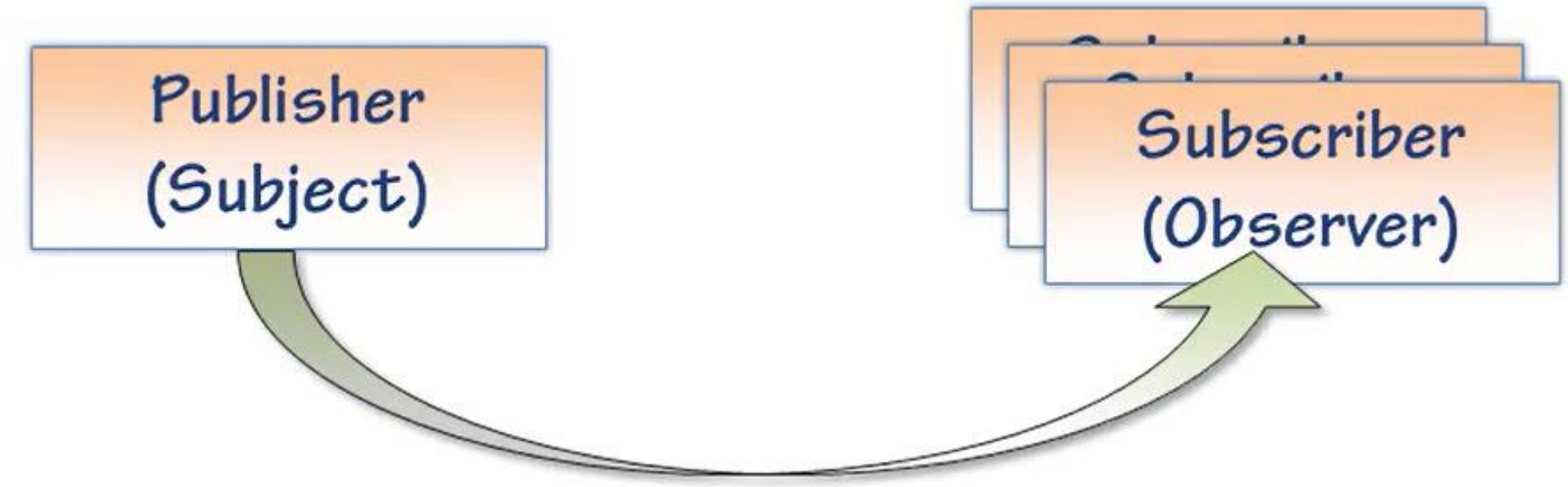
- Well-Described solution
- Shared Vocabulary
- Productivity between developers



Case 1

Observer Patterns

- Publisher and Subscriber Relationship



Observer Implementation

- Twitter, Facebook, Instagram Follow Mechanism
- Event Handlers

Demo

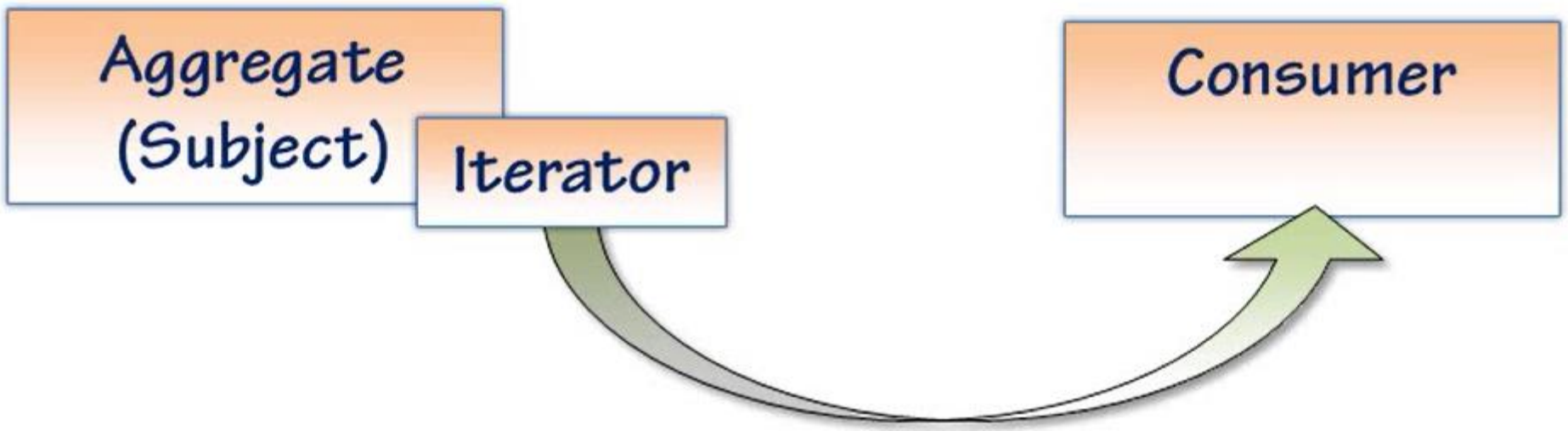
Proxy

- An placeholder object for another object to work with the object
- Implementation
  - Thumbnail on a web
  - A Proxy Object on a SOAP web services

Demo

Iterator Pattern

- Provide a way to access the elements of aggregate object sequentially
- Implementation
  - Foreach on C#
  - Inumerable<T>



Demo

Chain of Responsibility Pattern

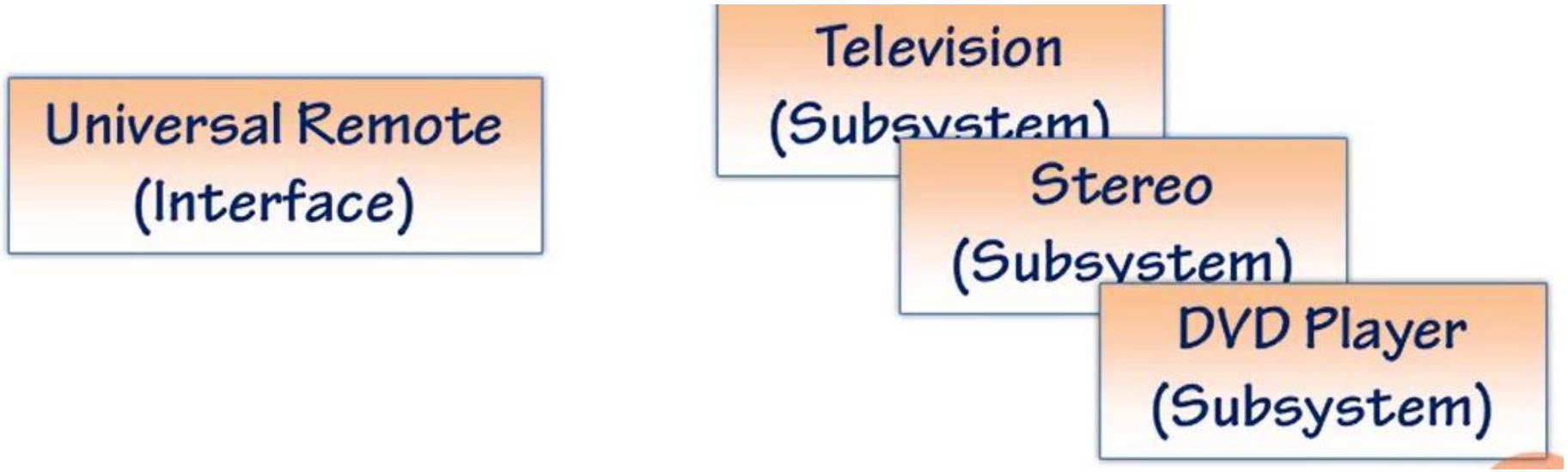
- Avoid coupling to a handle by giving separate responsibility
- Implementation
  - Exception handling
  - Approval Engine

```
private void Method2()
{
    switch (ExceptionBox.Text)
    {
        case "AccessViolationException":
            throw new AccessViolationException();
        case "NullReferenceException":
            throw new NullReferenceException();
        case "ArgumentException":
            throw new ArgumentException();
        case "Exception":
            throw new Exception();
    }
}
```

Demo

Facade Pattern

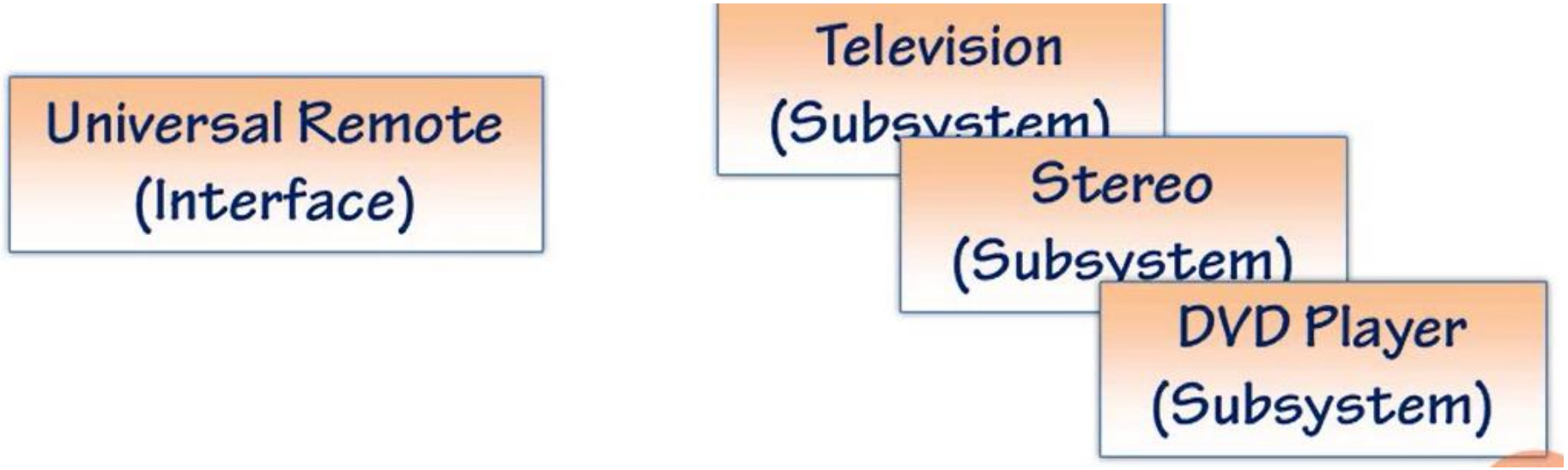
- Provide unified interface to simplify the process of functionality on sub system
- Implementation
  - Universal database access
  - Background worker



Demo

Factory Method Pattern

- Defining interface for creating object
- Implementation
  - Universal database access
  - Background worker



Keypoints

- Design Patterns are pieces of codes that related with how people built codes that solves specific problems
- Several common patterns
  - Observer
  - Proxy
  - Iterator
  - Façade
  - Factory
  - Class responsibility