PREGRADO

UNIDAD 1: Overview

SOFTWARE DESIGN PATTERNS

SI720 | Diseño y Patrones de Software

Al finalizar la unidad, el estudiante elabora y comunica artefactos de diseño de software aplicando principios básicos y patrones de diseño para un dominio y contexto determinados

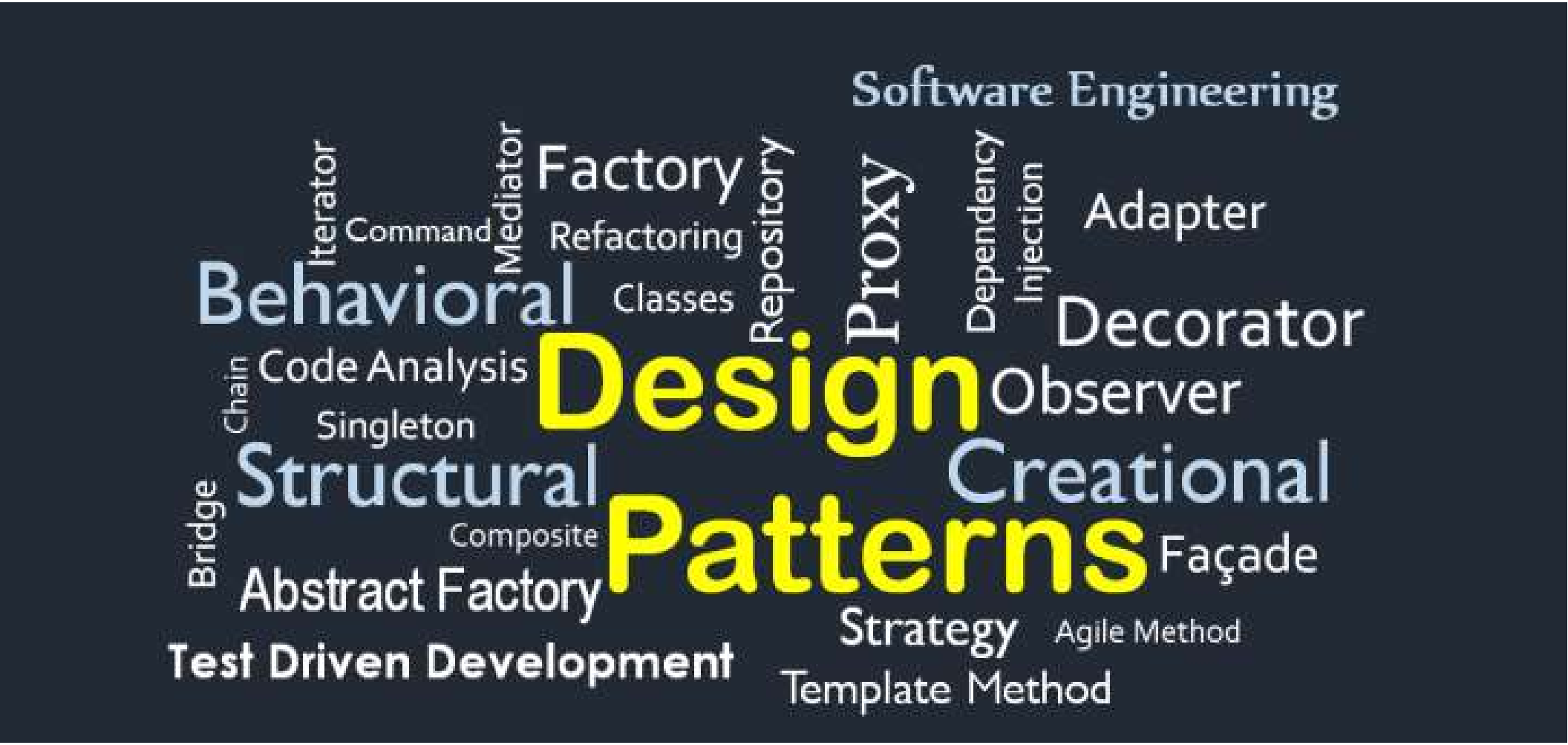
AGENDA

CONCEPT

EVOLUTION

GOF DESIGN PATTERNS

¿Qué entiendes por **pattern** (patrón)?



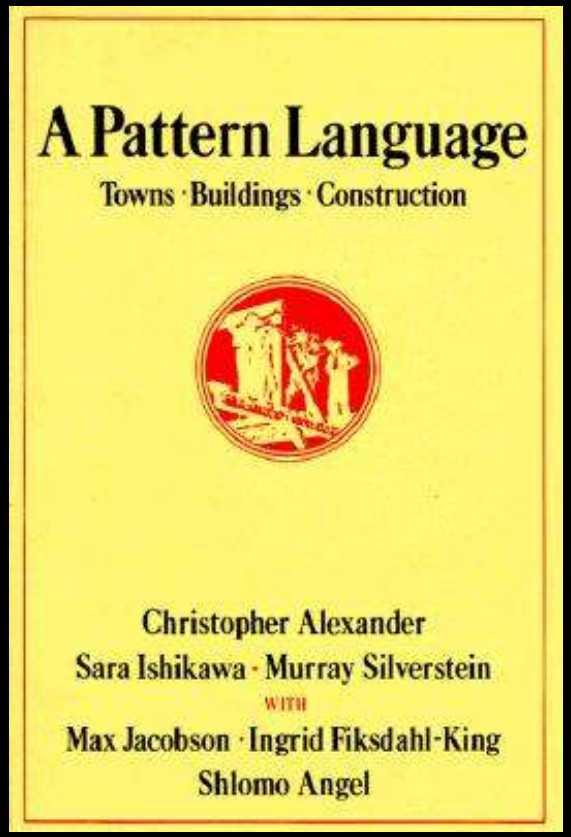
Pattern

Podría definirse pattern (patrón) como aquella serie de variables y constantes identificables dentro de un conjunto mayor de datos.

¿Qué es un pattern language?

Pattern Language

"A method of describing good design practices within a field of expertise"



A Pattern Language: Towns, Buildings, Construction
Christopher Alexander, 1977

Pattern Language

Elements

Name
Problem
Forces
Solution
Benefits / Consequences

AGENDA

CONCEPT

EVOLUTION

GOF DESIGN PATTERNS

Software Design Pattern

1987

Pattern Language in Programming | OOPSLA Conference
(Object-Oriented Programming, Systems, Languages & Applications)
1987

Software Design Pattern

1987

Pattern Language in Programming | OOPSLA Conference 1987
(Object-Oriented Programming, Systems, Languages & Applications)

“A pattern language guides a designer by providing workable solutions to all of the problems known to arise in the course of design. It is a sequence of bits of knowledge written in a style and arranged in an order which leads a designer to ask (and answer) the right questions at the right time.”

Software Design Pattern

1987

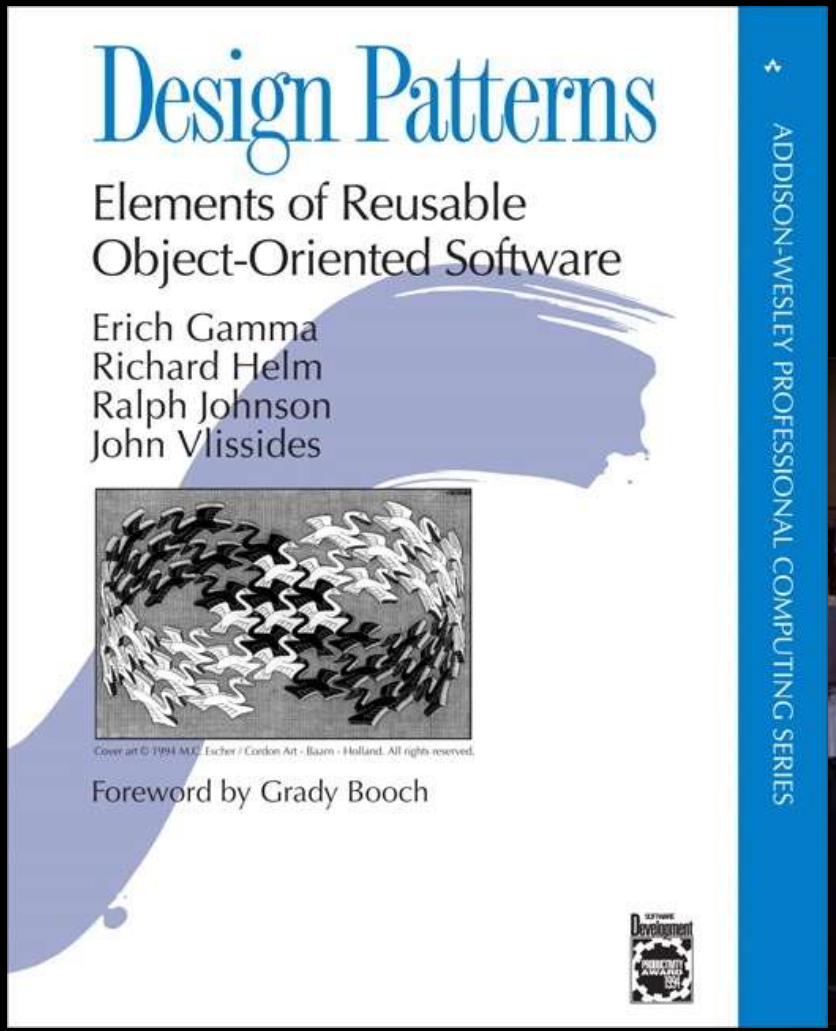
Pattern Language in Programming | OOPSLA Conference 1987

SmallTak UI Design
Patterns

Window Per Task
Few Panes Per Window
Standard Panes
Short Menus
Nouns and Verbs

Software Design Pattern

1994

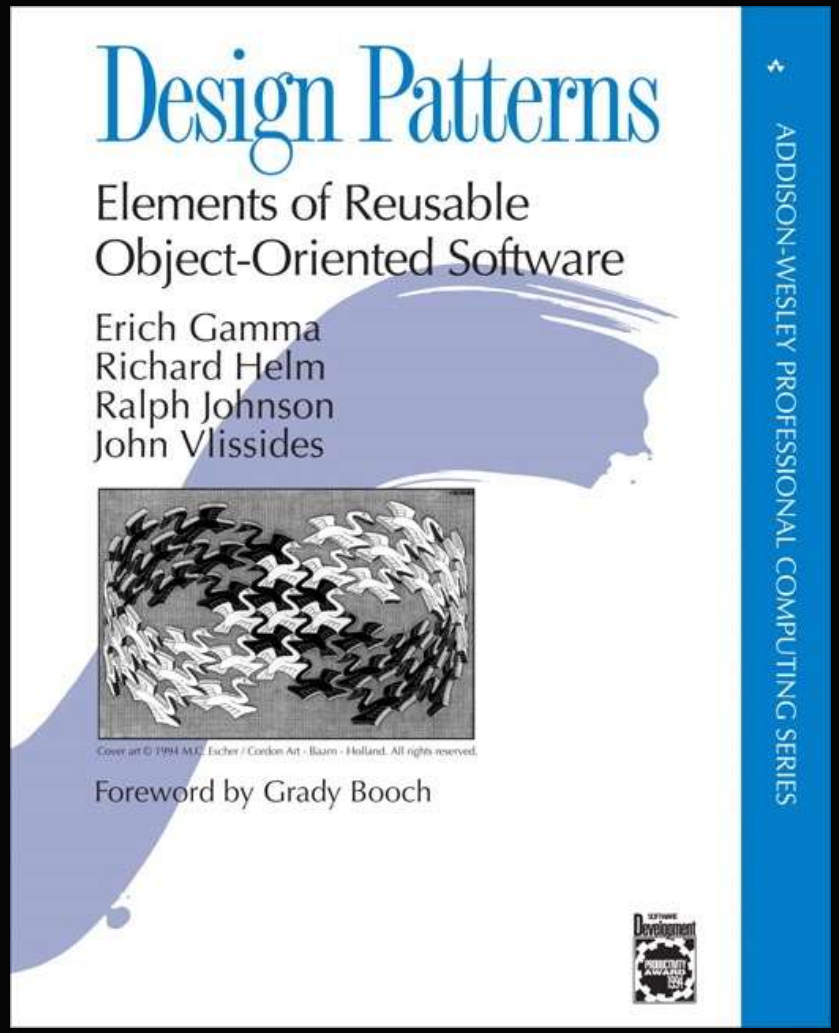


Gang of Four (GoF)
1994

Erick Richard Ralph John

Software Design Pattern

1994



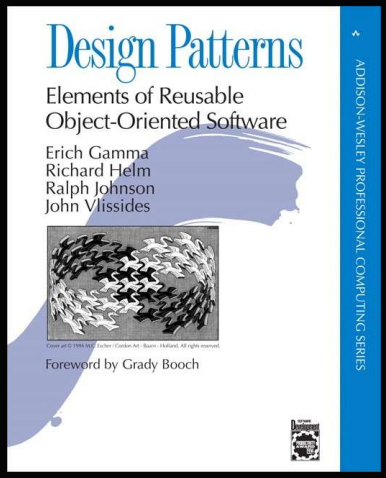
"A pattern is a solution to a problem in a context "

- Gang of Four (GoF)
1994

GoF Design Patterns

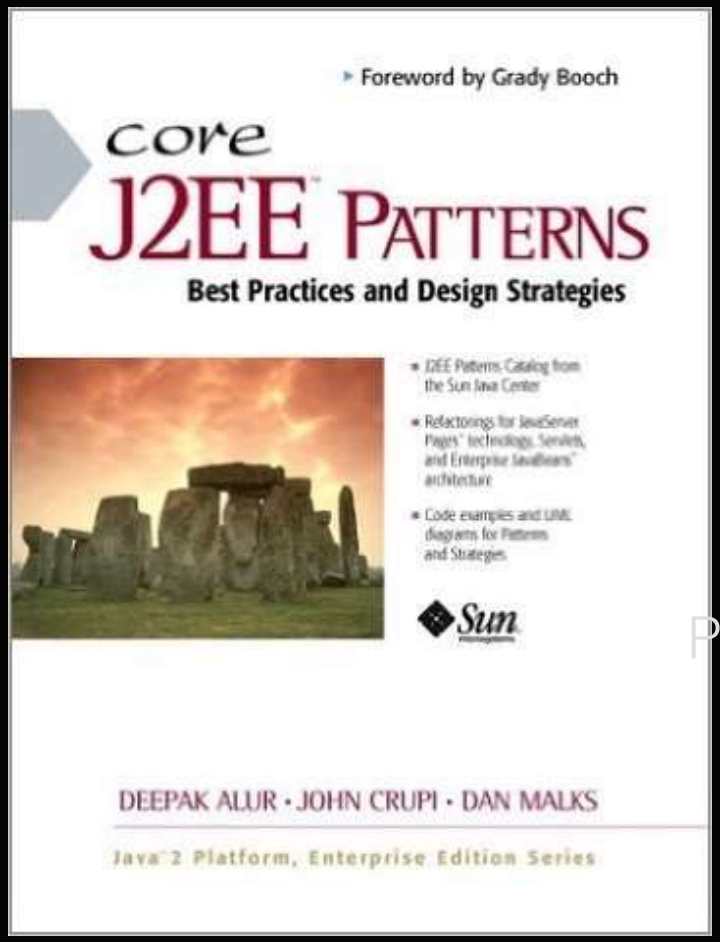
1995

Creational	Structural	Behavioral
Builder	Adapter	Chain of responsibility
Factory	Bridge	Command
Method	Composite	Interpreter
Prototype	Decorator	Iterator
Singleton	Façade	Mediator
	Flyweight	Memento
	Proxy	Observer
		States
		Strategy
		Template Method
		Visitor



Software Design Patterns

2001



Deepak Alur

John Crupi

Dan Malks

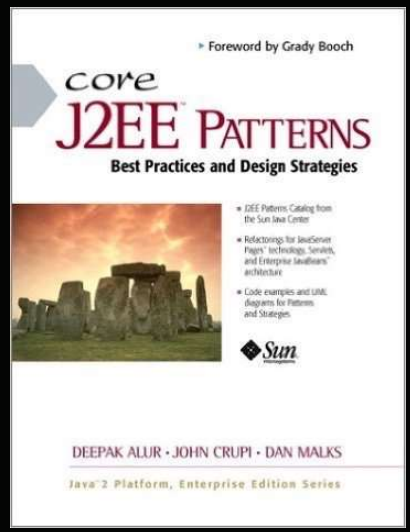
Prototyping Patterns for the J2EE Platform
JavaOne conference, 2000

Book, 2001

Core J2EE Patterns

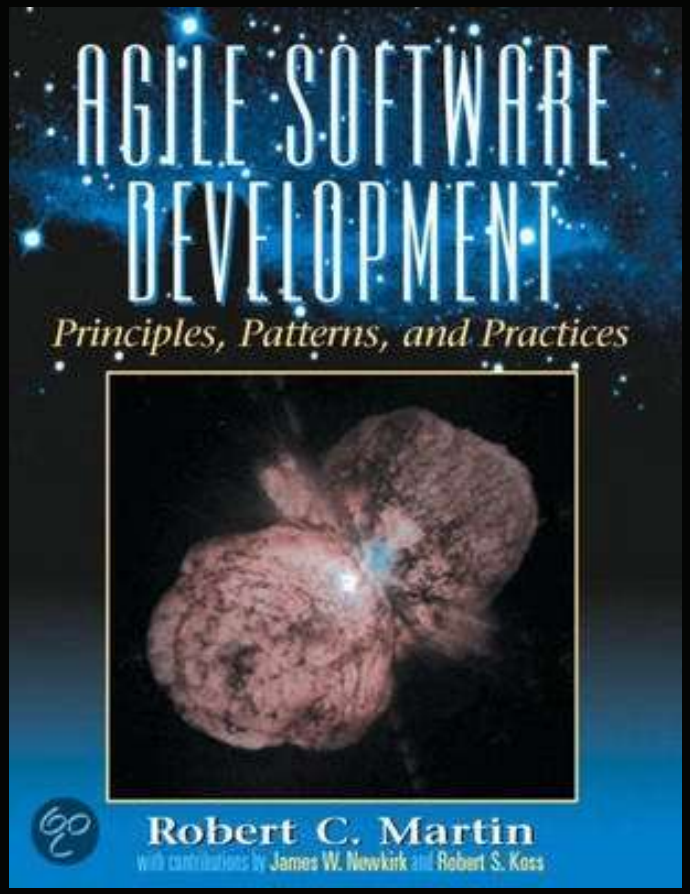
2001

Presentation Tier	Business Tier	Integration Tier
Intercepting Filter	Business Delegate	Data Access Object
Front Controller	Value Object	Service Activator
View Helper	Session Facade	
Composite View	Composite Entity	
Service to Worker	Value Object Assembler	
Dispatcher View	Value List Handler	
	Service Locator	



Software Design Pattern

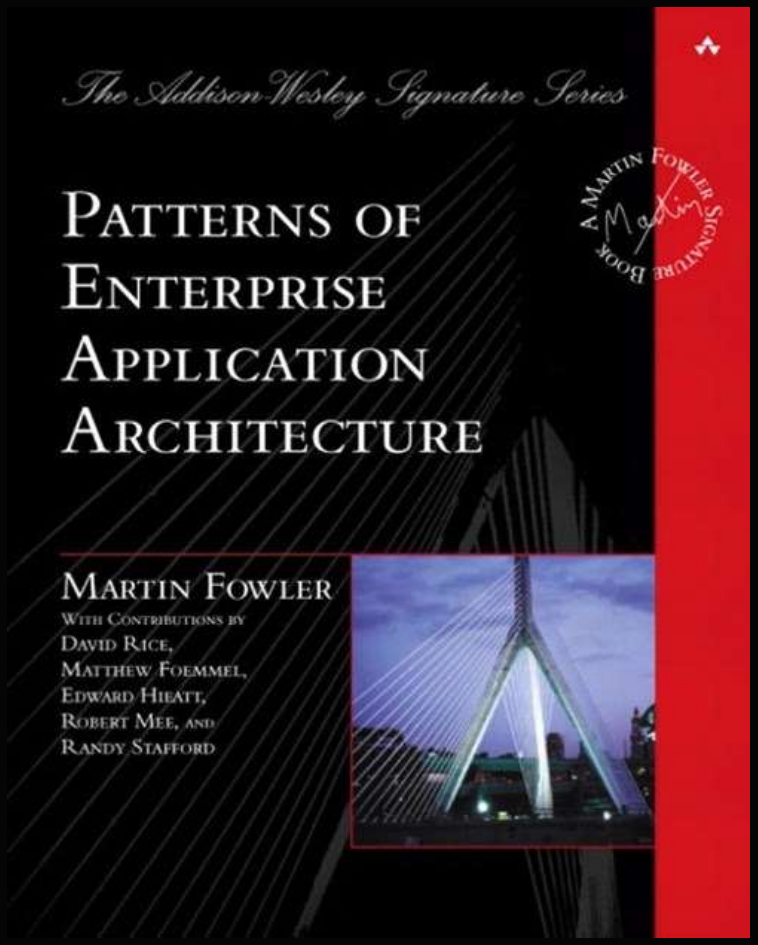
2002



Robert C. Martin
2002

Software Design Patterns

2002

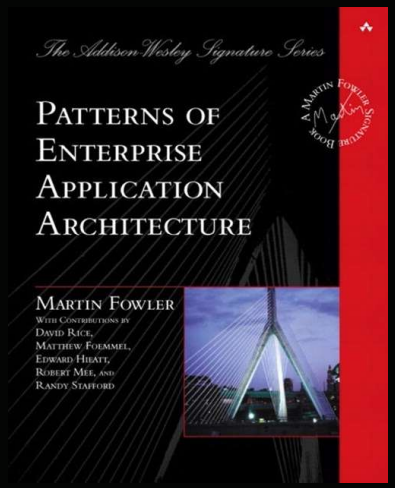


Martin Fowler
2002

Enterprise App Design Patterns

2002

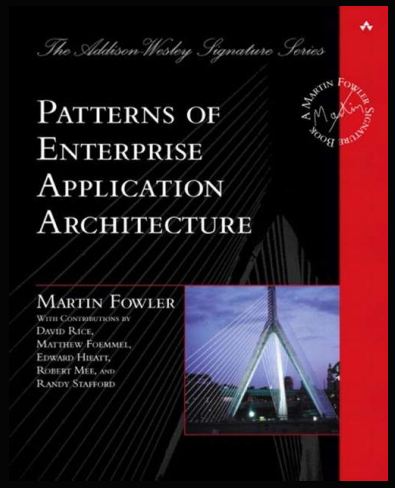
Domain Logic	Data Source Architectural Patterns	Object-Relational Behavioral	Object-Relational Structural	Object-Relational Metadata Mapping
Transaction Script Domain Model Table Module Service Layer	Table Data Gateway Row Data Gateway Active Record Data Mapper	Unit of Work Identity Map Lazy Load	Identity Field Foreign Key Mapping Association Table Mapping Dependent Mapping Embedded Value Serialized LOB Single Table Inheritance Class Table Inheritance Concrete Table Inheritance Inheritance Mappers	Metadata Mapping Query Object Repository



Enterprise App Design Patterns

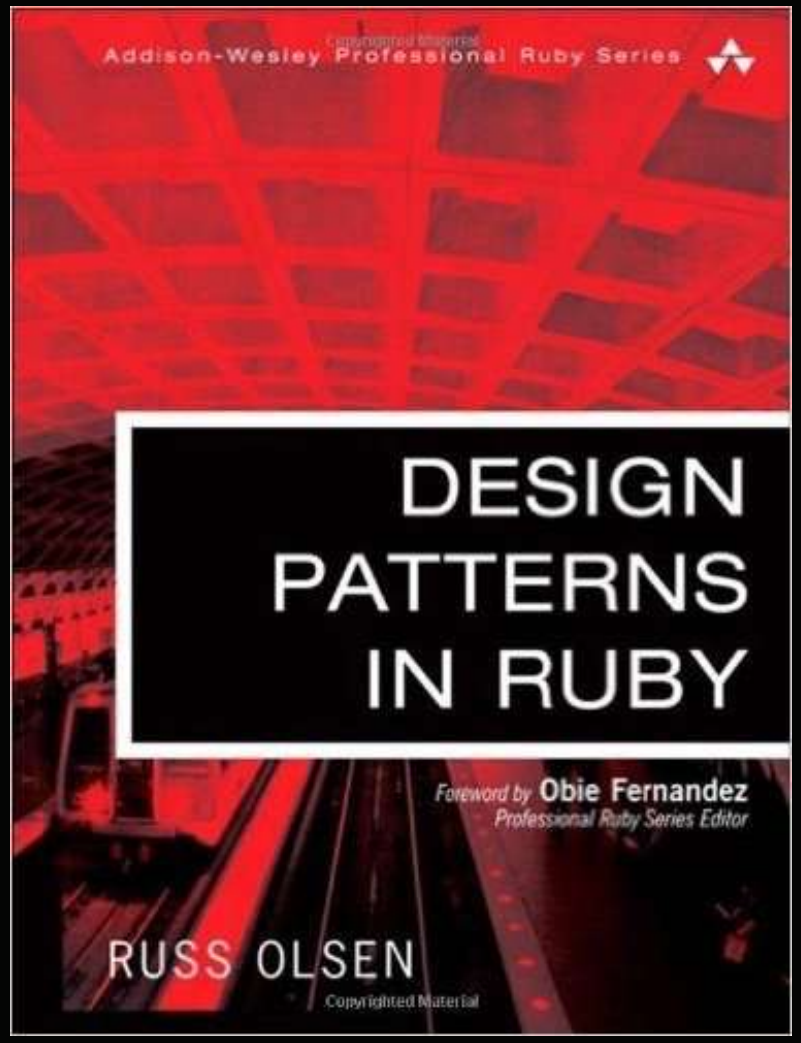
2002

Web Presentation	Distribution	Offline Concurrency	Session State	Base
Model View Controller	Remote Façade	Optimistic Offline Lock	Client Session State	Gateway
Page Controller	Data Transfer Object	Pessimistic Offline Lock	Server Session State	Mapper
Front Controller		Coarse Grained Lock	Database Session State	Layer Supertype
Template View		Implicit Lock		Separated Interface
Transform View				Registry
Two-Step View				Value Object
Application Controller				Money
				Special Case
				Plugin
				Service Stub
				Record Set



Software Design Patterns

2008



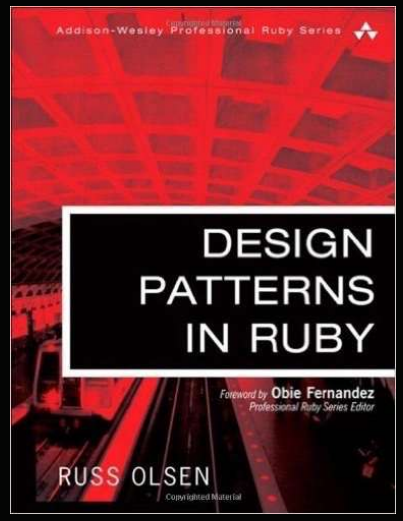
Russ Olsen

2008

Ruby Design Patterns

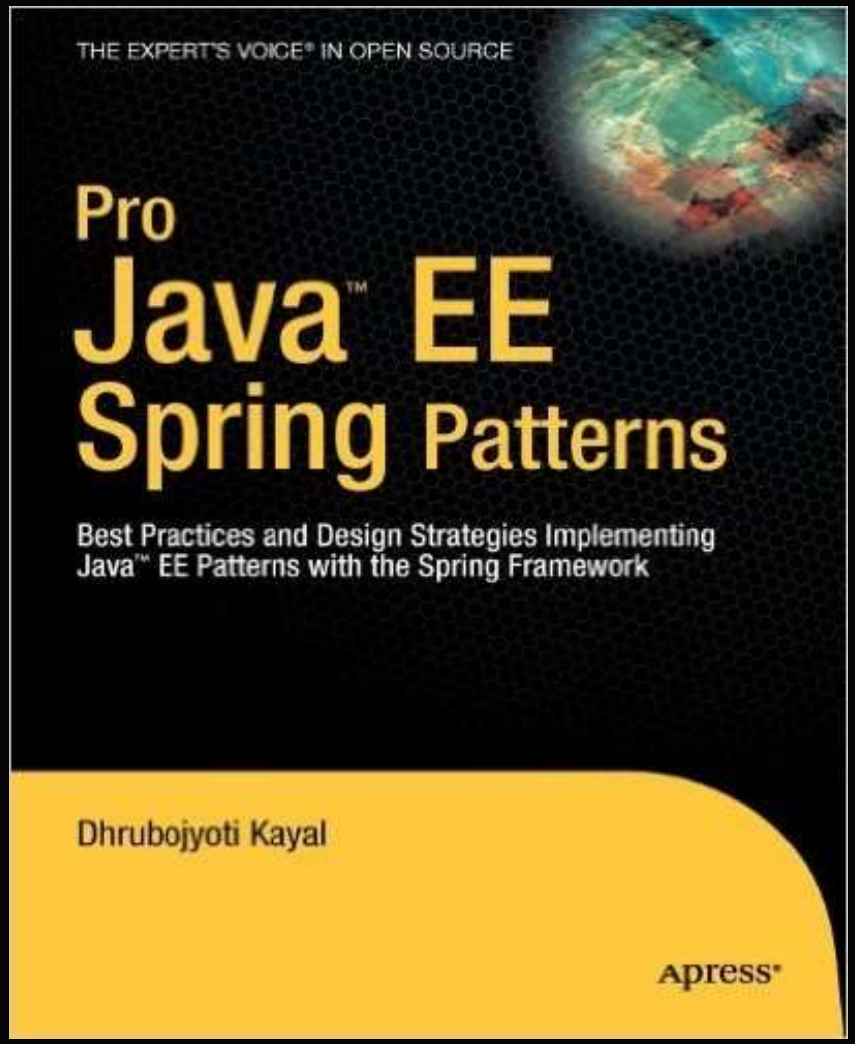
2008

GoF	Ruby
Template Method	Domain-Specific Languages
Strategy	Meta-programming
Observer	Convention Over Configuration
Composite	
Iterator	
Command	
Adapter	
Proxy	
Decorator	
Singleton	
Factory	
Builder	
Interpreter	



Software Design Patterns

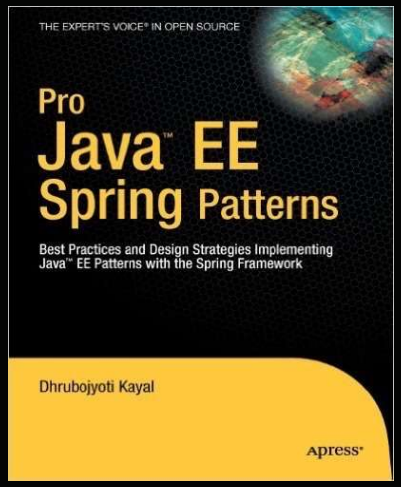
2008



Dhrubojyoti Kayal
2008

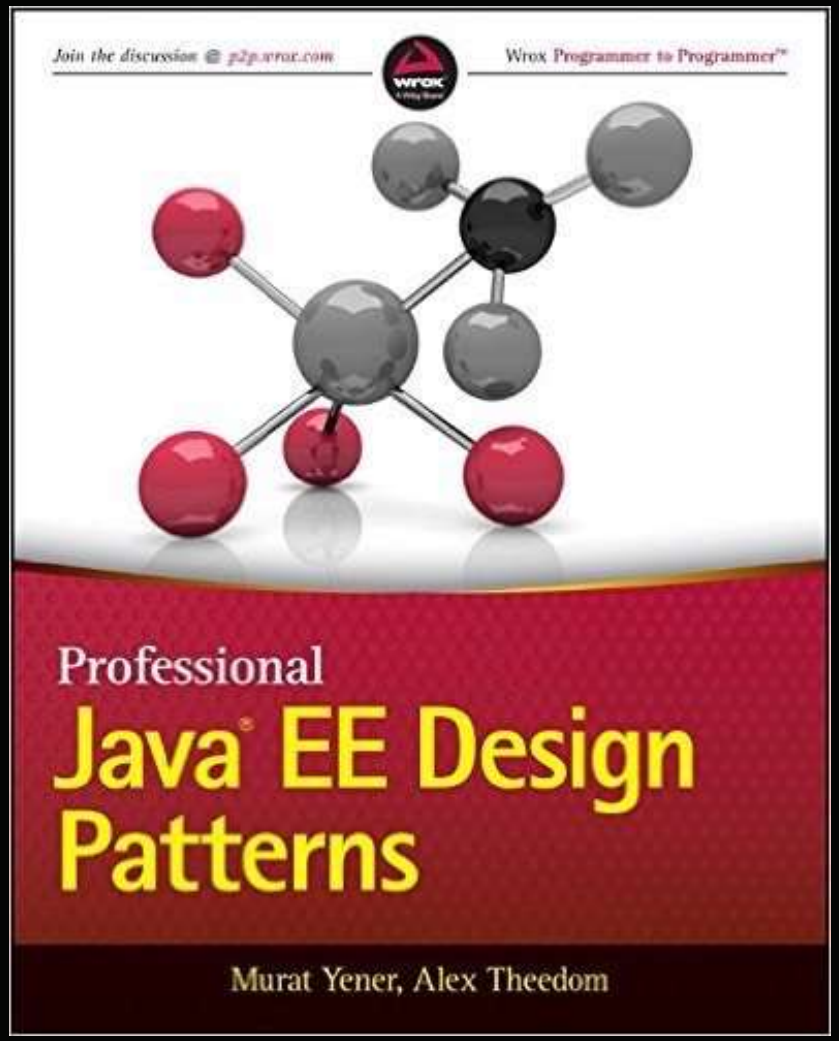
Spring-based App Design Patterns 2008

Presentation Tier	Business Tier	Integration Tier	Crosscutting
Front Controller	Service Locator	Data Access Object	Authentication and Authorization
Application Controller	Business Delegate	Procedure Access Object	Enforcer
Page Controller	Session Facade	Service Activator	Audit Interceptor
Context Object	Application Service	Web Service Broker	Domain Service Owner
Intercepting Filter	Business Interface		Transaction
View Helper			
Composite View			
Dispatcher View			
Service to Worker			



Java EE Design Patterns

2015



Murat Yener

Alex Theedom

2015

Java EE Design Patterns

GoF

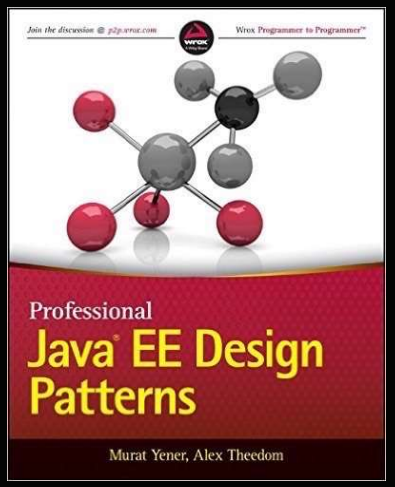
Facade
Singleton
Factory
Decorator
Observer

Enterprise

Data Access
Model View Controller

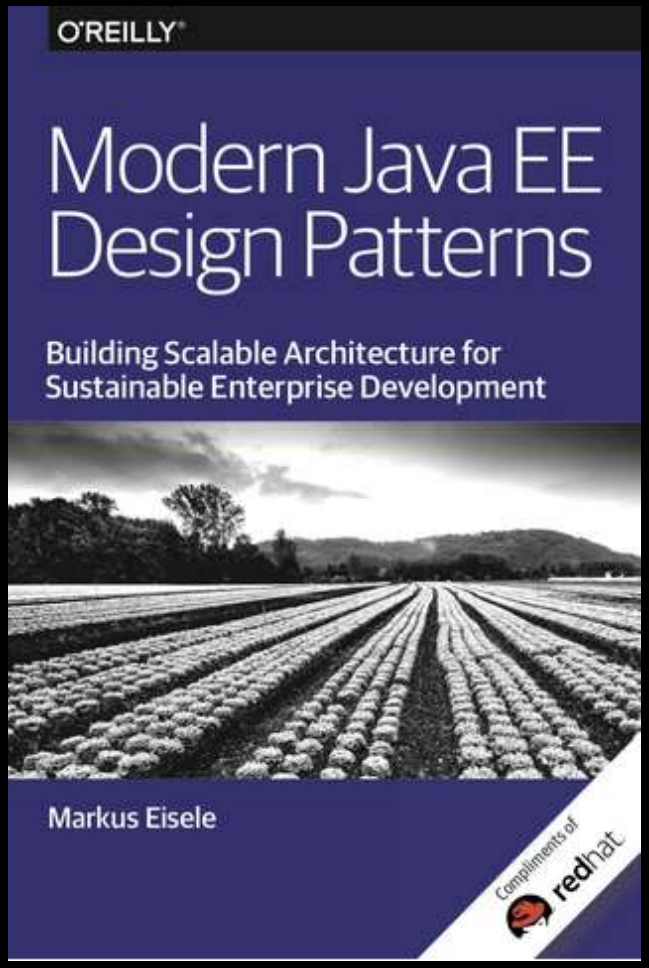
Additional

Aspect-Oriented Programming
Asynchronous
Timer Service
RESTful Web Services
Microservice Architecture
Anti-Patterns



Modern Java EE Design Patterns

2016



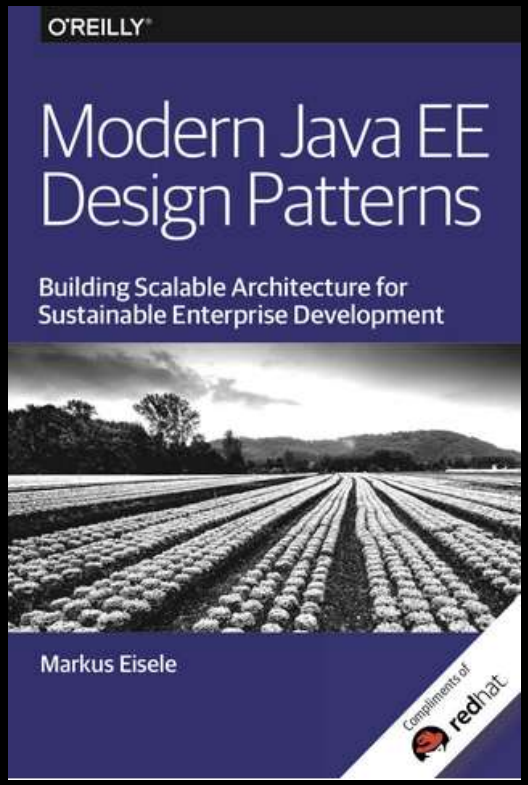
Markus Eisele

Modern Java EE Design Patterns

2016

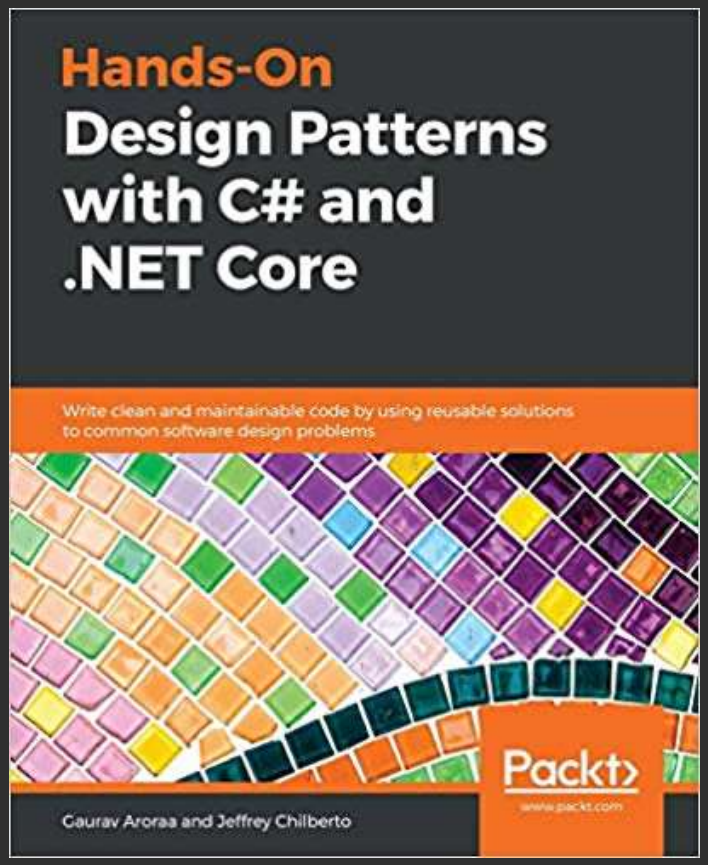
Microservices

Agregator
Proxy
Pipeline
Shared Resources
Asynchronous Messaging



Hands-On Design Patterns with C# and .NET Core

2018



Gaurav Aroraa

Software Design Patterns

Additional Patterns

Web UI Design Patterns
Oracle Application Development Framework Functional Patterns
Reactive Programming Patterns

AGENDA

CONCEPT

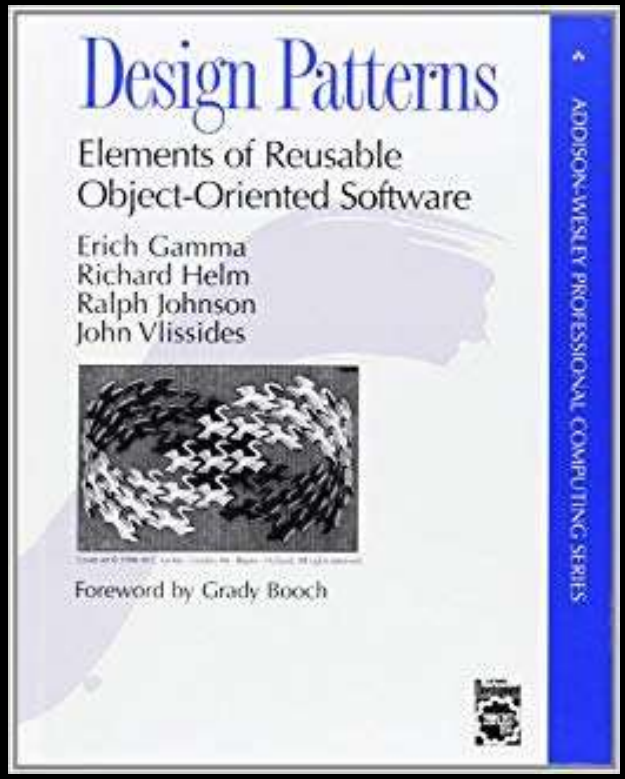
EVOLUTION

GOF DESIGN PATTERNS

History

Como vimos, en 1994 se publicó el libro "Design Patterns: Elements of Reusable Object Oriented Software" escrito por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides.

Ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos.



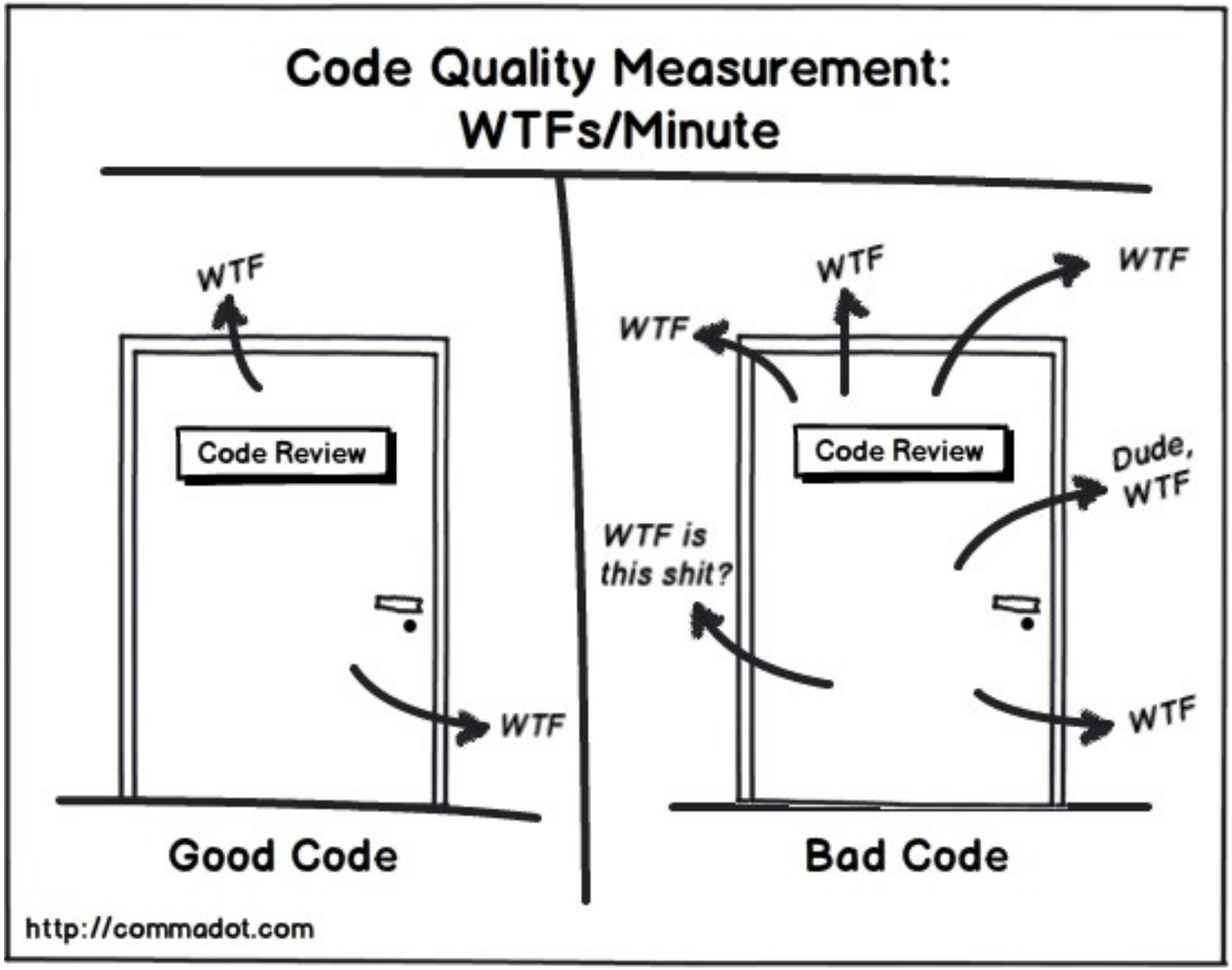
GoF Design Patterns

Los patrones de diseño describen cómo resolver problemas recurrentes de diseño de software orientado a objetos flexible y reutilizable.

> "Un patrón describe un problema el cual ocurre una y otra vez en nuestro ambiente, y además describen el núcleo de la solución a tal problema, en tal una manera que puedes usar esta solución millones de veces, sin hacer lo mismo dos veces."

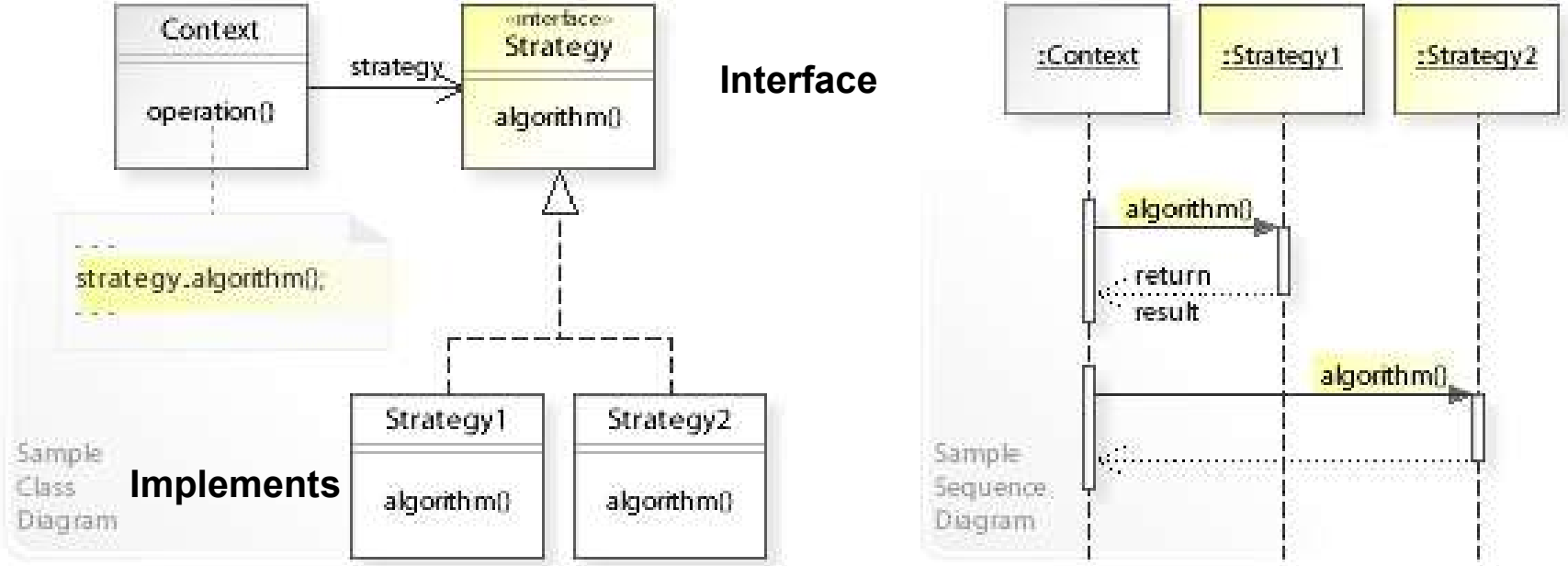
Son soluciones exitosas a problemas comunes

Es decir:


Code Quality Measurement:
WTFs/Minute
WTF
Code Review
WTF
Good Code
WTF
WTF
WTF
WTF
Code Review
Dude,
WTF
WTF is
this shit?
WTF
Bad Code
http://commadot.com

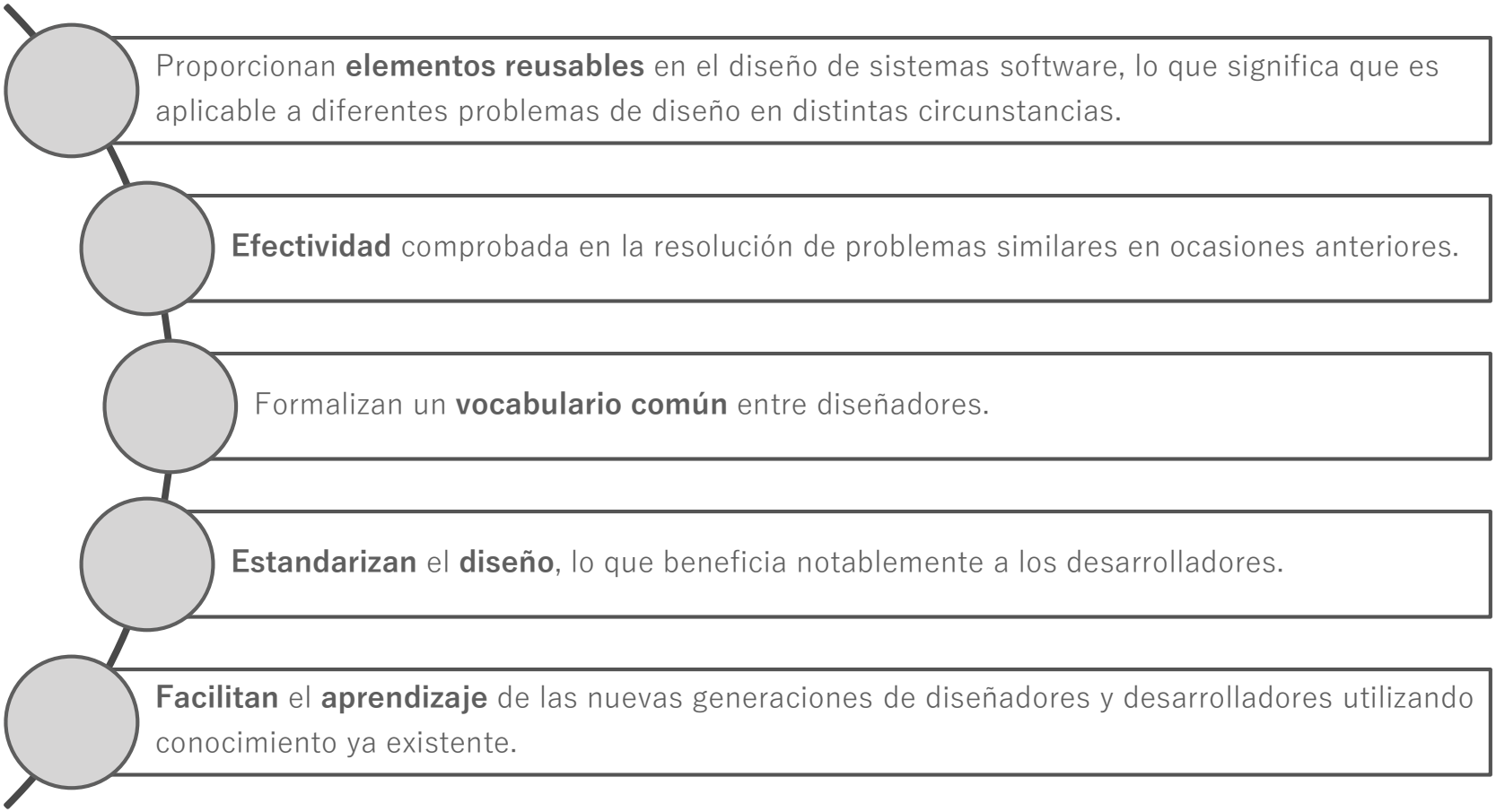
Por ejemplo

Program to an interface, not an implementation. First Design Principle [GoF, p18]



By Vanderjoe - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=60733582

Beneficios


Proporcionan **elementos reusables** en el diseño de sistemas software, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.
Efectividad comprobada en la resolución de problemas similares en ocasiones anteriores.
Formalizan un **vocabulario común** entre diseñadores.
Estandarizan el **diseño**, lo que beneficia notablemente a los desarrolladores.
Facilitan el **aprendizaje** de las nuevas generaciones de diseñadores y desarrolladores utilizando conocimiento ya existente.

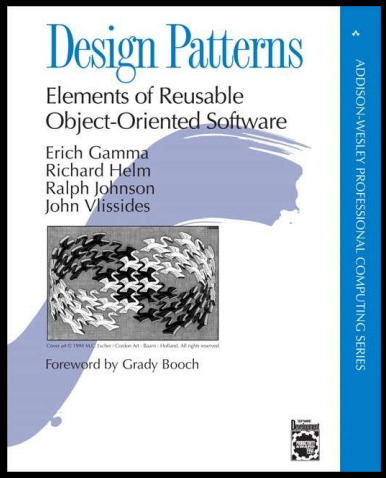
Entonces, ¿estás listo para aprender sobre patrones de diseño?

Types

S.N.	Pattern & Description
1	**Creational Patterns** These design patterns provide a way to create objects while hiding the creation logic, rather than instantiating objects directly using new operator. This gives program more flexibility in deciding which objects need to be created for a given use case.
2	**Structural Patterns** These design patterns concern class and object composition. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities.
3	**Behavioral Patterns** These design patterns are specifically concerned with communication between objects.
4	**J2EE Patterns** These design patterns are specifically concerned with the presentation tier. These patterns are identified by Sun Java Center.

GoF Design Patterns

Creational	Structural	Behavioral
Builder	Adapter	Chain of responsibility
Factory	Bridge	Command
Method	Composite	Interpreter
Prototype	Decorator	Iterator
Singleton	Façade	Mediator
	Flyweight	Memento
	Proxy	Observer
		States
		Strategy
		Template Method
		Visitor



RESUMEN

Recordemos

- Los patrones de diseño describen cómo resolver problemas recurrentes de diseño de software orientado a objetos flexible y reutilizable.
- The Gang of Four (GoF)son los cuatro autores del libro, "Patrones de diseño: elementos de software orientado a objetos reutilizables.
- Los patrones de diseño ofrecen soluciones generalizadas en forma de plantillas que pueden aplicarse a problemas del mundo real.
- Es más importante comprender los conceptos que describen los patrones de diseño, en lugar de memorizar sus clases , métodos y propiedades exactas.
- Los tipos de GoF Design Patterns: estructurales, de comportamiento y creacionales.

REFERENCIAS

Para profundizar

- Design Patterns- Libro de Erich Gamma, John Vlissides, Ralph Johnson y Richard Helm.
- http://www.blackwasp.co.uk/gofpatterns.aspx
- http://www.w3sdesign.com/

PREGRADO

Ingeniería de Software

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería

UPC
Universidad Peruana
de Ciencias Aplicadas

Prolongación Primavera 2390,
Monterrico, Santiago de Surco
Lima 33 - Perú
T 511 313 3333
https://www.upc.edu.pe

exígete, innova