OO Design Principles and Patterns

Architecture and Dependencies

Initial Design is clean, elegant and compelling

Over period of time, rot sets in

Takes a lot of effort to make simplest of changes

Symptoms of Rotting Design

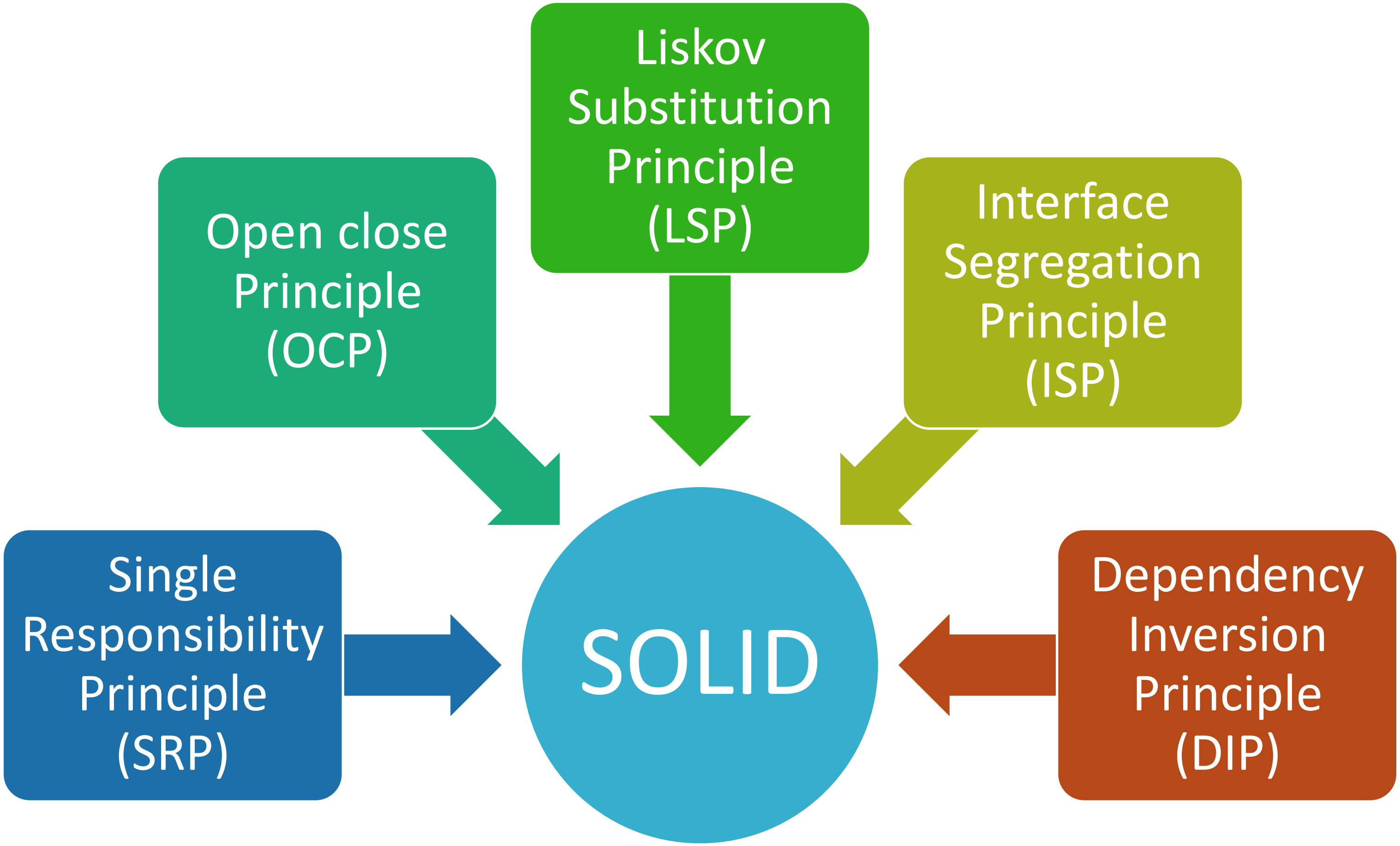
- Rigidity
- Fragility
- Immobility
- Viscosity

Causes of Rotting

- Changing Requirements
- Dependency Management

Design Principles

Design Principles



Liskov Substitution Principle (LSP)

Open close Principle (OCP)

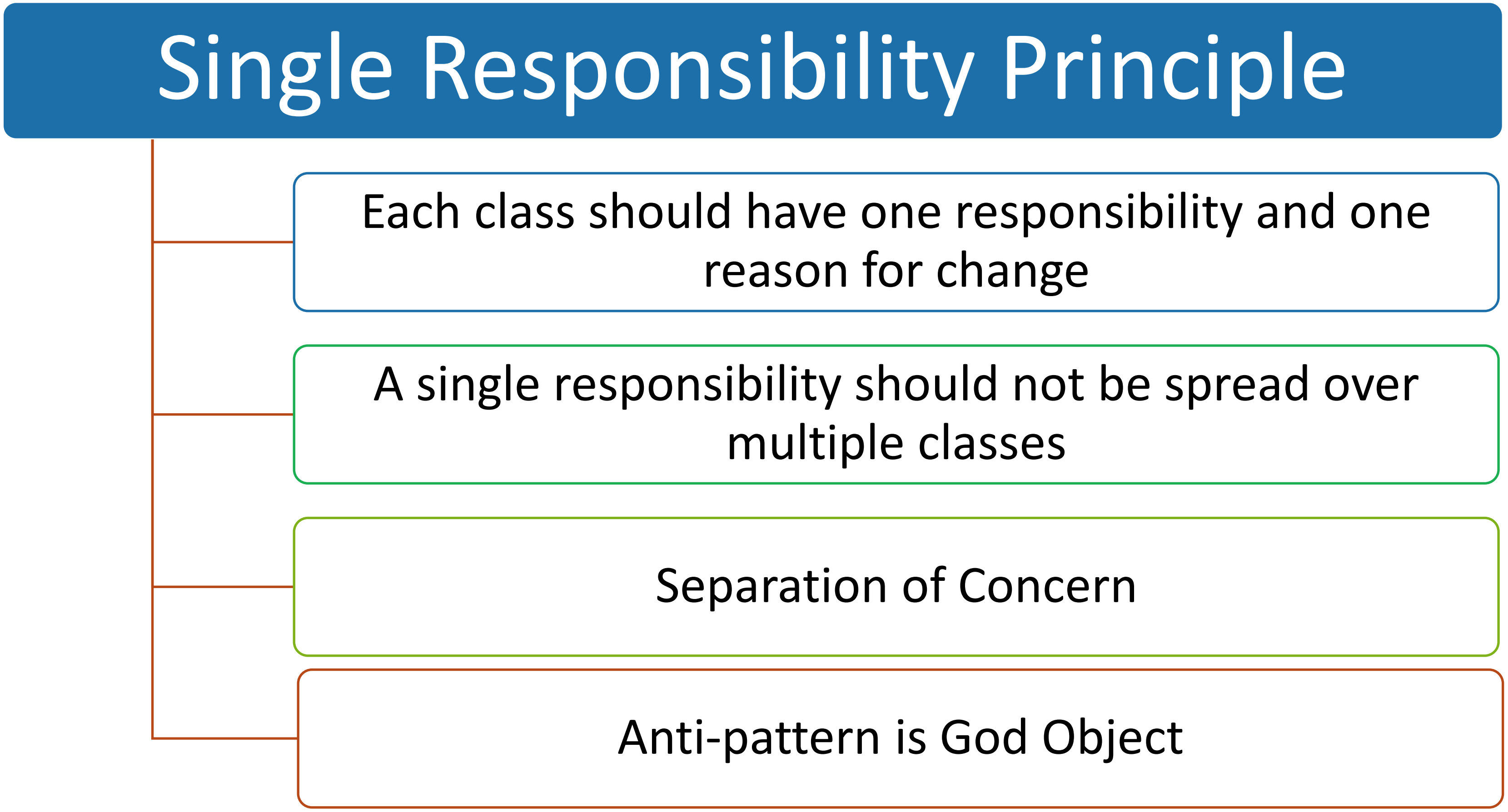
Interface Segregation Principle (ISP)

Single Responsibility Principle (SRP)

SOLID

Dependency Inversion Principle (DIP)

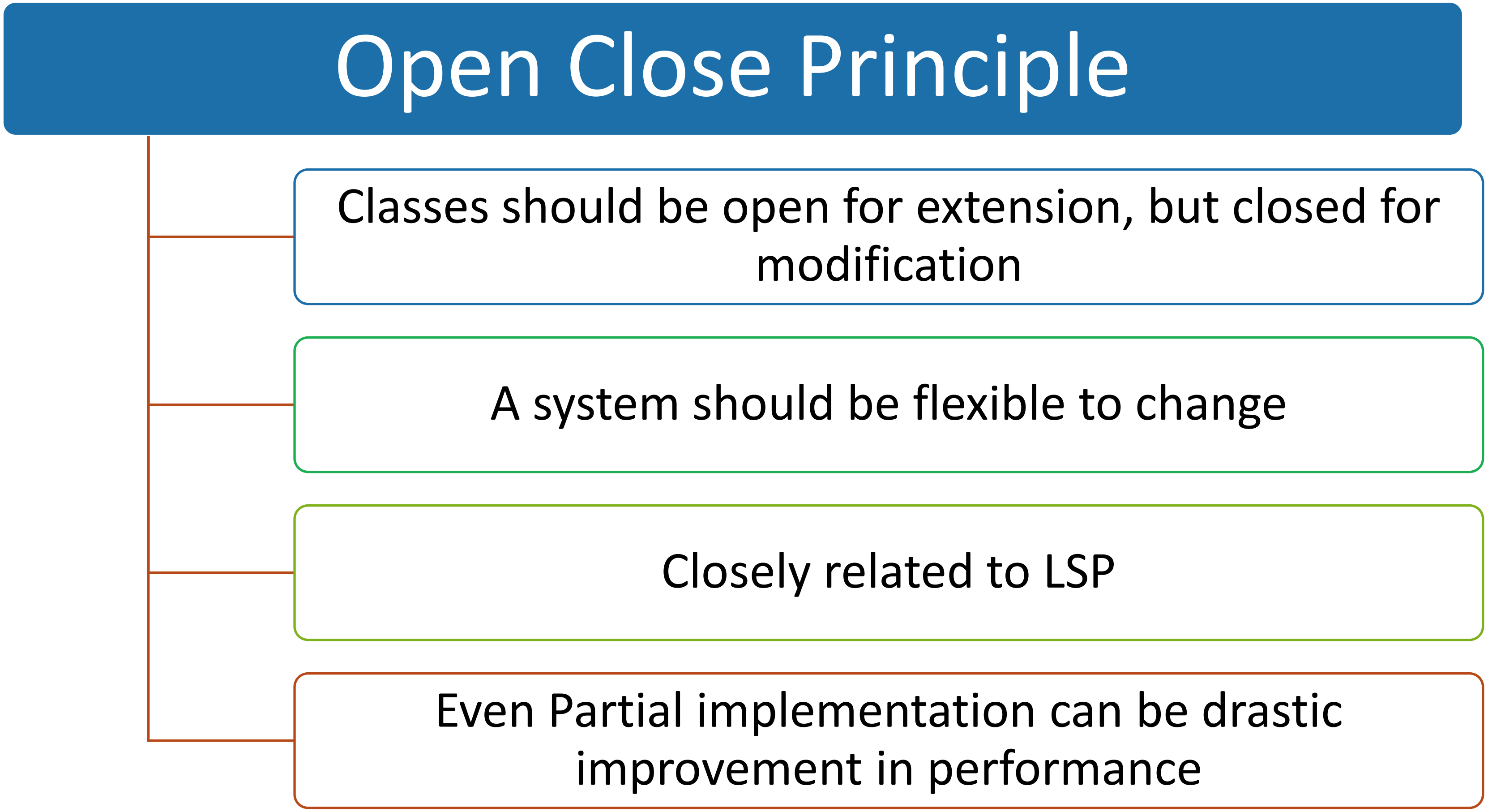
Design Principles



Single Responsibility Principle

- Each class should have one responsibility and one reason for change
- A single responsibility should not be spread over multiple classes
- Separation of Concern
- Anti-pattern is God Object

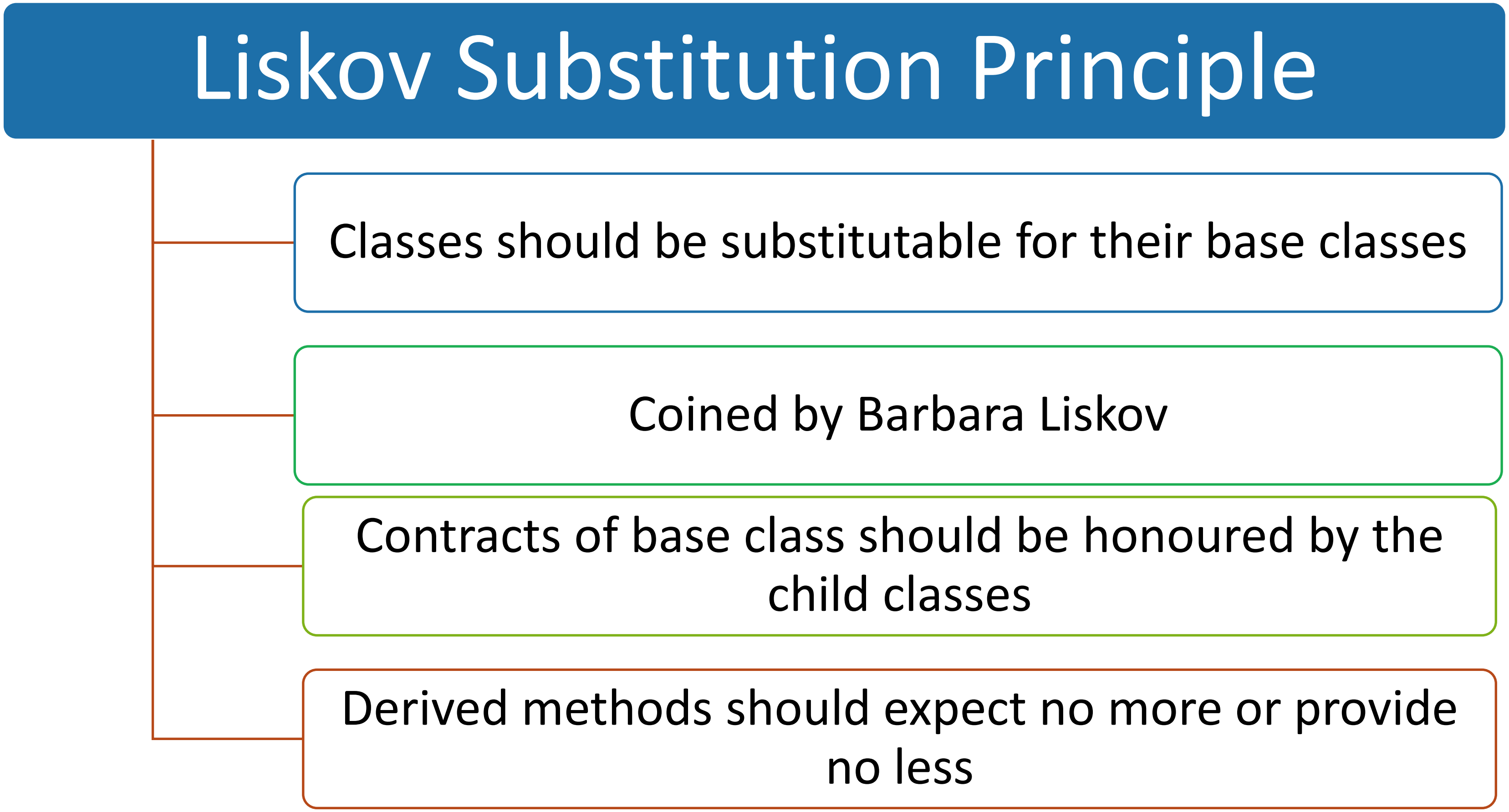
Design Principles



Open Close Principle

- Classes should be open for extension, but closed for modification
- A system should be flexible to change
- Closely related to LSP
- Even Partial implementation can be drastic improvement in performance

Design Principles



Liskov Substitution Principle

- Classes should be substitutable for their base classes
- Coined by Barbara Liskov
- Contracts of base class should be honoured by the child classes
- Derived methods should expect no more or provide no less

Design Principles

Interface Segregation Principle

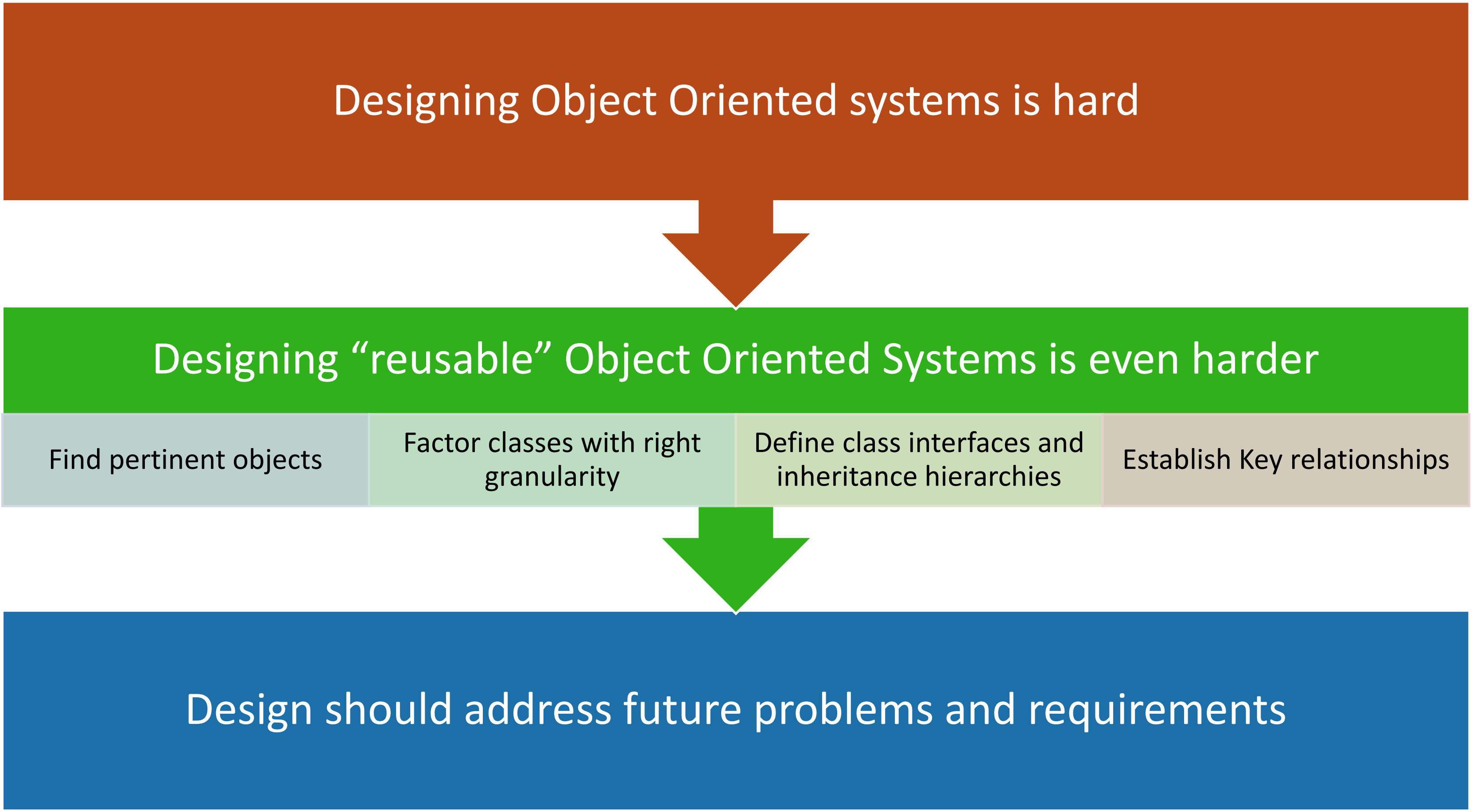
- Many client-specific interfaces are better than one general purpose interface
- Monitor the number of interfaces created.

Dependency Inversion Principle

- Depend on abstractions. Do not depend on concretions
- Subsystems should expose interfaces or abstract classes

Design Patterns

Introduction



Designing Object Oriented systems is hard

Designing “reusable” Object Oriented Systems is even harder

- Find pertinent objects
- Factor classes with right granularity
- Define class interfaces and inheritance hierarchies
- Establish Key relationships

Design should address future problems and requirements

Introduction

Recurring design structures promote

- Abstraction
- Flexibility
- Modularity
- Elegance

Problem

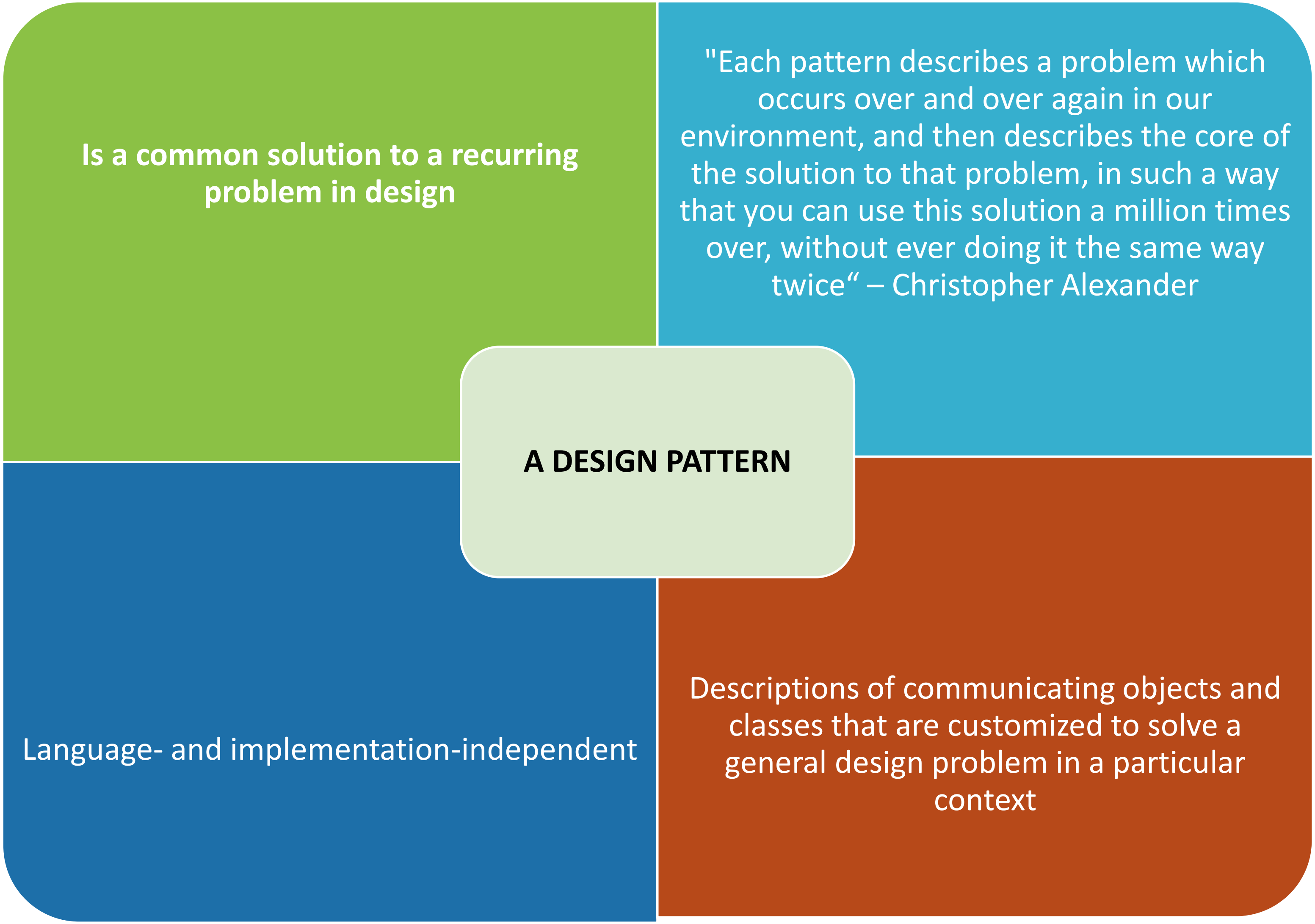
- Capturing the pattern,
- Communicating the intent, and
- Apply the knowledge

Introduction

Design Patterns

- Systematically names, explains and evaluates recurring designs
- Captures design Experience in a form that people can use effectively
- Easies reuse of successful designs and architectures
- Helps choose design alternatives to make system reusable
- Improves documentation and maintenance of existing systems

What Is a Design Pattern?



Is a common solution to a recurring problem in design

"Each pattern describes a problem which 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 this solution a million times over, without ever doing it the same way twice“ – Christopher Alexander

A DESIGN PATTERN

Language- and implementation-independent

Descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context

What Is a Design Pattern?

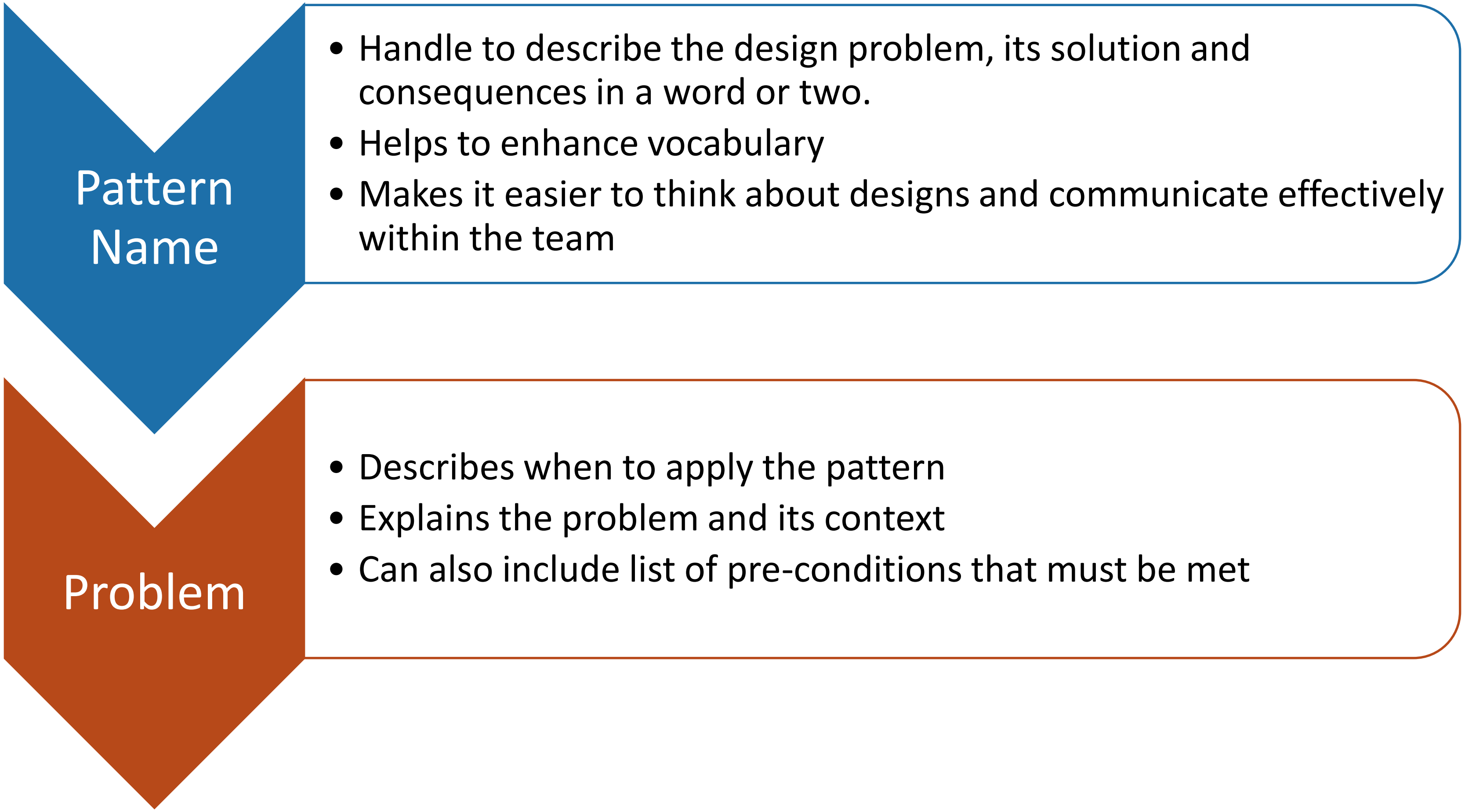
A design pattern has 4 basic parts:

- 1. Pattern Name
- 2. Problem
- 3. Solution
- 4. Consequences and trade-offs of application

Identifies the key aspects like

- Participating classes and instances
- Their roles and collaborations
- Distribution of responsibilities

What Is a Design Pattern?



Pattern Name

- Handle to describe the design problem, its solution and consequences in a word or two.
- Helps to enhance vocabulary
- Makes it easier to think about designs and communicate effectively within the team

Problem

- Describes when to apply the pattern
- Explains the problem and its context
- Can also include list of pre-conditions that must be met

What Is a Design Pattern?

Solution

- Describes elements that make up the design, their relationships, responsibilities and collaborations
- Doesn’t describe a particular concrete design or implementation
- Provides and abstract description and general arrangement of elements

Consequences

- Results or trade-offs of applying the pattern
- Are critical for evaluating design alternatives
- Can often concern space and time trade-offs

Describe Design Patterns

Pattern Name and classification	• Conveys the essence of the pattern and its classification
Intent	• Short statement to describe what the pattern does
Also Known as	• Other well-known names for the pattern
Motivation	• Illustrates the design problem
Applicability	• Situations in which the pattern can be applied
Structure	• Graphical representation of the classes
Participants	• Classes and/or objects participating and their relationships

Describe Design Patterns

Collaborations	• How participants collaborate to carry out responsibilities
Consequences	• Trade-offs and results of using this pattern
Implementation	• Pitfalls, hints and techniques to be aware of
Sample Code	• Code fragments
Known Uses	• Examples of patterns found in systems
Related Patterns	• Closely relation to other patterns

Goals

Codify good design

- Distil and disseminate experience
- Aid to novices and experts alike
- Abstract how to think about design

Give design structures explicit names

- Common vocabulary
- Reduced complexity
- Greater expressiveness

Capture and preserve design information

- Articulate design decisions succinctly
- Improve documentation

Facilitate restructuring/refactoring

- Patterns are interrelated
- Additional flexibility

Classification of GoF Design Pattern

Creational	Structural	Behavioral
Factory Method	Adapter	Interpreter
Abstract Factory	Bridge	Template Method
Builder	Composite	Chain of Responsibility
Prototype	Decorator	Command
Singleton	Flyweight	Iterator
	Facade	Mediator
	Proxy	Memento
		Observer
		State
		Strategy
		Visitor