Patterns
in Software Engineering

Giovanni Sakti

Starqle

What is Patterns?

Patterns

Patterns describes a problem, which occurs over and over again in our environment and then desribes the core of the solution to that problem...

[cited from `Christopher Alexander']

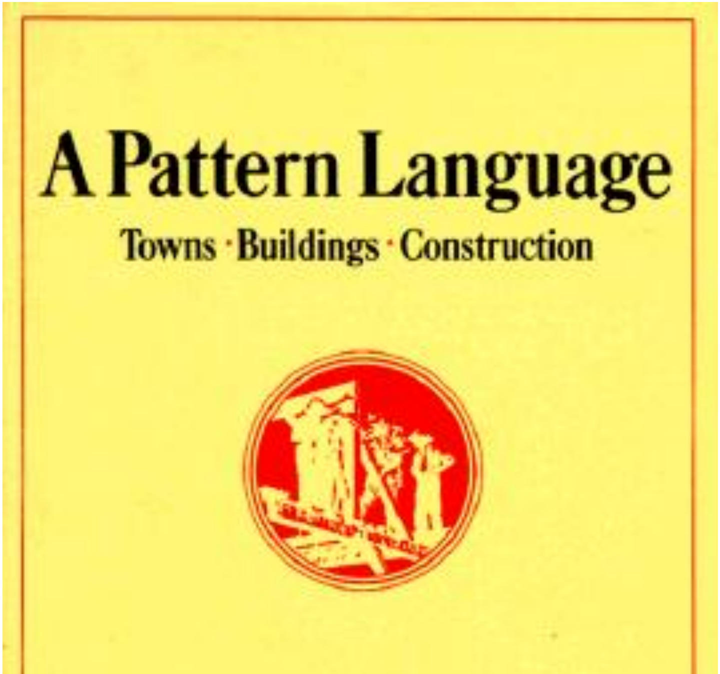
Patterns

…in such a way that you can use this solution a million times over without ever doing it the same way twice.

[cited from `Christopher Alexander']

Okay, but who is Christopher Alexander?

He authored widely-influential book in 1977.



A Pattern Language

Towns · Buildings · Construction

The book influences multiple disciplines including software engineering.

So, what is patterns from the viewpoint of software engineering?

Patterns

Patterns are **distilled commonalities** that you find in software.

Patterns

It allows us to deconstruct a large complex structure and build using the pattern itself.

Patterns

Patterns contain solution that have **developed** and **evolved** over time.

Patterns

It is rarely designs that people tend to get initially.

We know that designing software is **hard**.

Designing software with reusable components are even **harder**.

Your design should be **specific** to the problem at hand, but **general enough** to address future problems and requirements.

Reusable & flexible design is **difficult**, if not **impossible**, to get "right" the first time.

Even for experienced designer.

Reusability

Instead, experienced designer won't try to solve every problem from scratch.

Reusability

They will try to **reuse** existing solution instead.

Reusability

So, patterns help designer gets a design ″right″ **faster**.

What can learning patterns help you?

Expectations

Common design vocabulary

Expectations

Documentation and learning aid

Expectations

An adjunct to existing methods

Expectations

A target for refactoring

Expectations

- Common design vocabulary
- Documentation and learning aid
- An adjunct to existing methods
- A target for refactoring

Patterns Essential Elements

Patterns Essential Elements

An excellently documented patterns will have several elements attached to it.

Patterns Essential Elements

Which you can use to learn more about them.

Patterns Essential Elements

- Name
- Intent
- Sketch

Patterns Essential Elements (cont'd)

- Problem
- Solution
- Consequence(s)

Patterns Essential Elements (cont'd)

- When to Use It
- Example(s)

Patterns Categories

Patterns Categories

There are several categories of patterns, based on the **level** in which they reside.

Patterns Categories

From ″lowest″ level to ″highest″ level

- Programming Paradigms
- Design Patterns
- Architectural Patterns

Patterns Categories

We'll try to discuss it one-by-one.

Programming paradigms

Programming Paradigms

Programming paradigms[*1], in a way, is a pattern.

[*1]Such as: OO, Functional or Procedural

Programming Paradigms

To be precise, programming paradigms is the **smallest** and **lowest** level of patterns possible.

Programming Paradigms

Programming paradigms are most likely to **influence** patterns that reside above it.

Programming Paradigms

And because programming paradigms are **tightly coupled** to programming language..

Programming Paradigms

..our pick of programming language may **influence** the way we design our software.

Design Patterns

Design Patterns

Design patterns are code-level commonalities.

Design Patterns

Providing schemes for refining &
building smaller subsystems.

Design Patterns

Design patterns are **medium-scale tactics** that flesh out some of the structure & behaviour of entities and their relationships.

Design Patterns

As we discuss previously, design patterns may be **influenced** by programming paradigms.

Design Patterns

Some design patterns can be very **important** or pale to **insignificance** due to language that we use.

Design Patterns

Design patterns can be categorized further.

But first let us discuss about the last category of pattern.

Architectural Patterns

Architectural Patterns

Architectural patterns on the other hand, are commonalities at **higher level** than design patterns.

Architectural Patterns

Architectural patterns are **high level strategies**.

Architectural Patterns

Architectural patterns concerns:

- Large-scale components
- Global properties
- Mechanism of a system

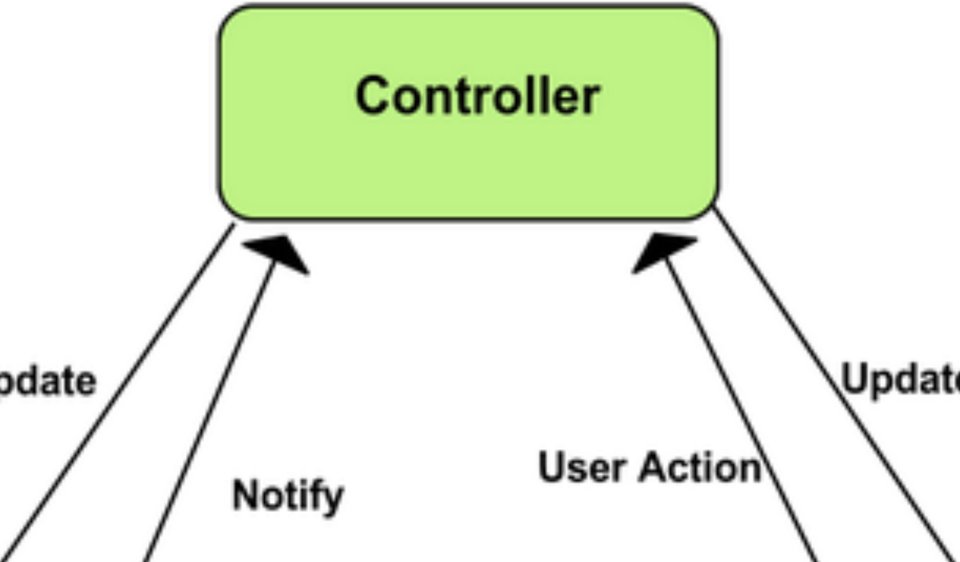
Architectural Patterns

One of the most well-known architectural pattern is the **MVC architecture**.

MVC Architecture

MVC intents are to promote efficient code reuse and parallel development.

MVC Architecture



MVC Architecture

It tries to solve the problem of tightly-coupled relation between UI codes and logic that hinders reusability.

MVC Architecture

It does so by separating codes into **three concerns**: models, views and controllers.

MVC Architecture

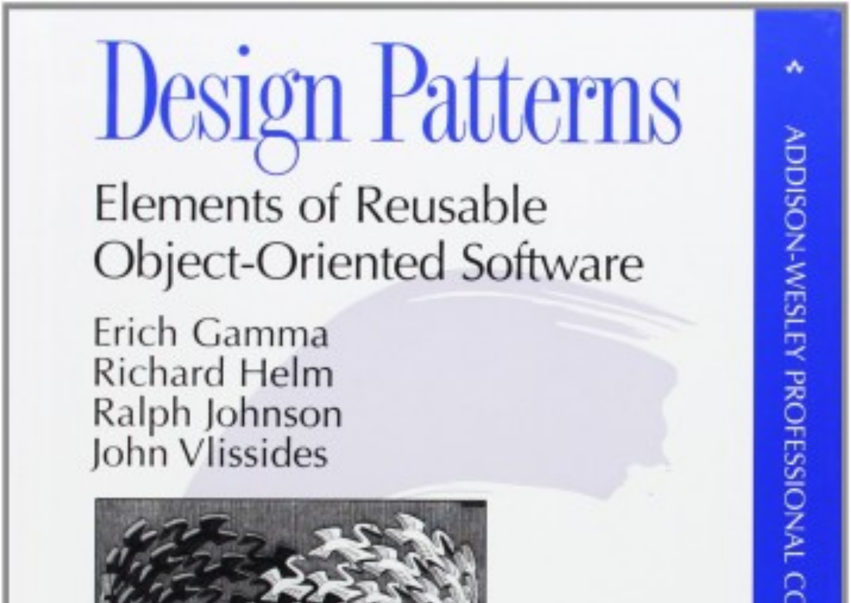
Notice that we already discuss about the name, intent, sketch, problem and solution provided by a pattern.

Design Patterns Categories

Design Patterns Categories

In arguably the most influential book on design patterns (The **GoF book**),

Design Patterns Categories



Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Design Patterns Categories

the authors categorize design patterns into three categories

- Creational
- Structural
- Behavioural

Creational Patterns

Creational Patterns

Creational patterns concern about object creation.

Creational Patterns

It **abstract** the **instantiation** process.

Creational Patterns

They help make a system **independent** on how its objects are created, composed and represented.

Creational Patterns

Useful when creating objects with particular behaviour **requires more** than simply instantiation a class.

Creational Patterns

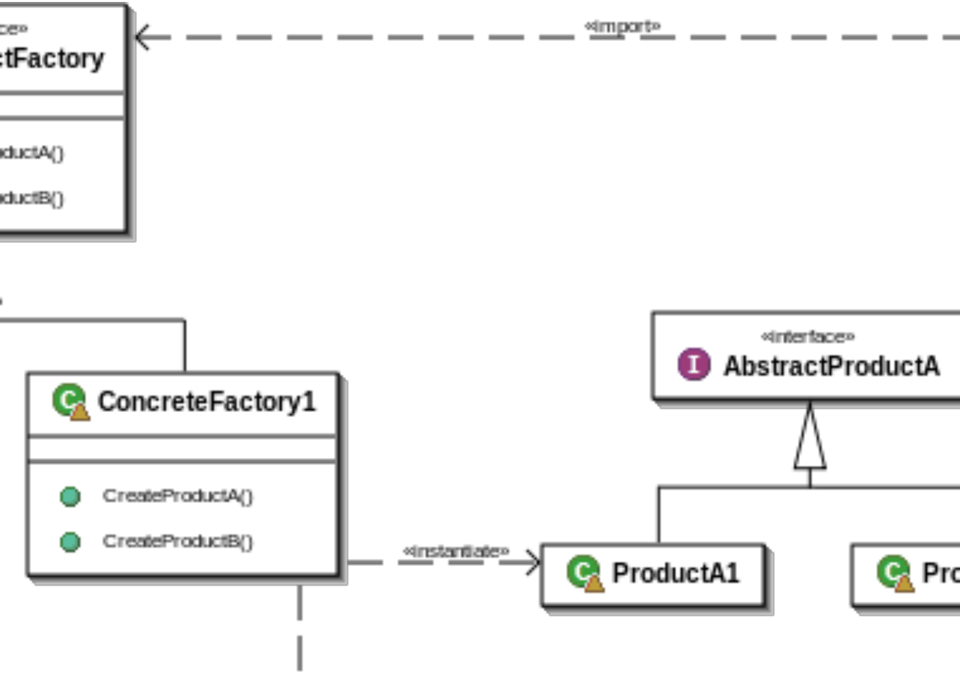
Favour system that prefer to use **object composition** instead of class inheritance.

Creational Patterns

Example: Abstract Factory

Abstract Factory

Provide an interface for creating families of related or dependent objects without specifying their concrete class.



ce»
tFactory
ductA()
ductB()
«import»
«interface»
AbstractProductA
ConcreteFactory1
CreateProductA()
CreateProductB()
«instantiate»
ProductA1
Pro

Structural Patterns

Structural Patterns

Structural patterns deal with the **compositions** of classes or objects to **form larger structures**.

Structural Patterns

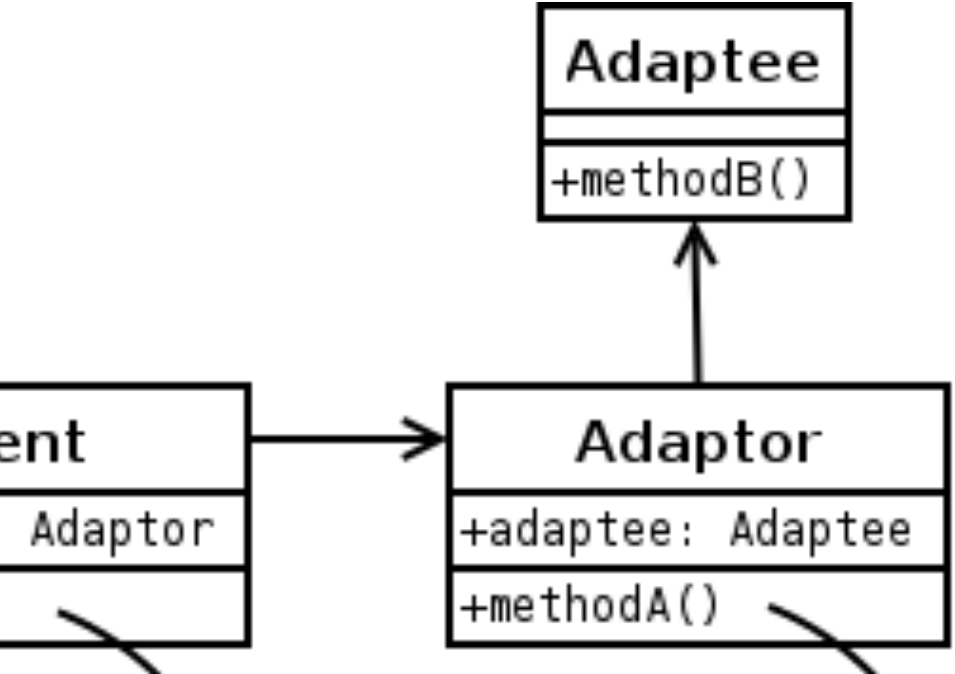
Example: Adapter

Adapter

Convert the interface of a class into another interface clients expect.

Adapter

Adapter lets classes **work together** that **couldn't otherwise** because of incompatible interfaces.


Adaptee
+methodB()
ent
Adaptor
Adaptor
+adaptee: Adaptee
+methodA()

Behavioural Patterns

Behavioural Patterns

Behavioural patterns characterize the way in which classes or objects **interact** and **distribute responsibility**.

Behavioural Patterns

Not just patterns of classes and objects but also the patterns of **communication** between them.

Behavioural Patterns

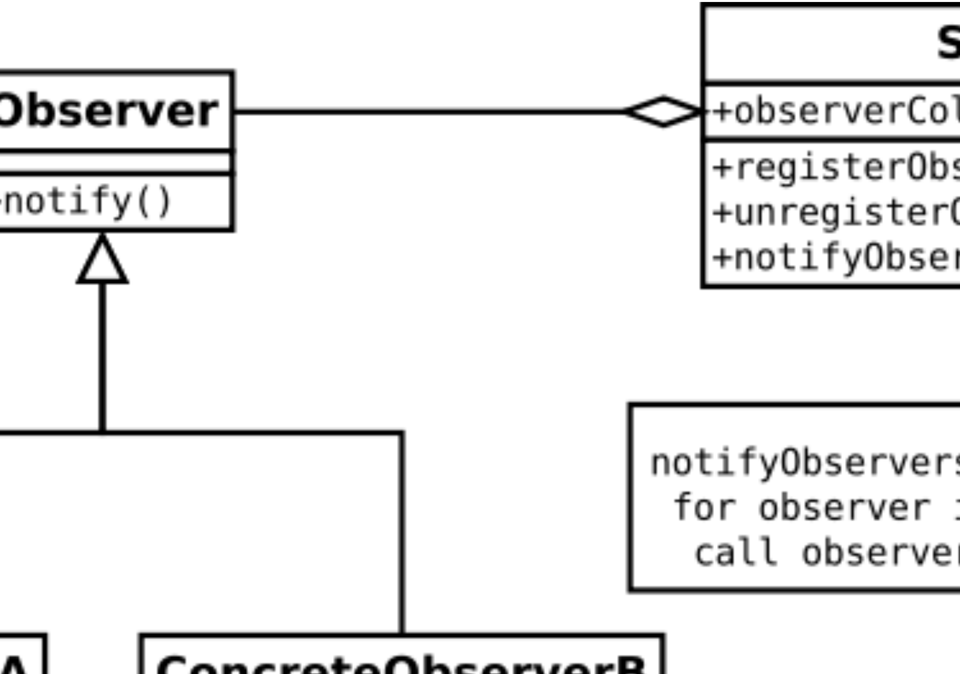
Example: Observer or Pub-Sub

Observer

Define **one-to-many dependency** between objects.

Observer

When one object change state, all its **dependents are notified** and updated automatically.


Observer
notify()
S
+observerCol
+registerObs
+unregister(
+notifyObser
notifyObservers
for observer i
call observer
ConcreteObserverB
A

Design Patterns Categories

There are no limits in defining design pattern categories, what we just discussed is just a (famous) example.

How to Utilize Patterns Properly?

Utilizing Patterns

Consider how patterns solve the problems

Utilizing Patterns

Scan intent and sketch sections

Utilizing Patterns

Study how patterns relate with each other

Utilizing Patterns

Study patterns of like purpose

Utilizing Patterns

Examine a cause of redesign

Utilizing Patterns

Consider what should be variable in your design

Anti-patterns

Anti-patterns

There are also patterns that have negative consequences when it is present in our software

Anti-patterns

It is called the **anti-patterns**

Anti-patterns

Anti-patterns are common response to a recurring problem that is usually **ineffective** and risks being highly **counterproductive**.

Anti-patterns

Example: Big ball of mud

Big ball of mud

Software system that **lacks** a perceivable **architecture**.

Big ball of mud

Although undesirable from a software engineering PoV, such systems are common in practice.

Big ball of mud

Due to business pressure, developers turnover and code entropy.

Thanks!