INRIA

Selected Design Patterns

Stéphane Ducasse
stephane.ducasse@inria.fr
http://stephane.ducasse.free.fr/

Goal

What are patterns?
Why?
Patterns are not god on earth
Example

Design Patterns

Design patterns are ***recurrent*** solutions to design ***problems***
They are ***names***
Composite, Visitor, Observer...
There are pros and cons

From Architecture

- Christoffer Alexander
  - "The Timeless Way of Building", Christoffer Alexander, Oxford University Press, 1979, ISBN 0195024028
- More advanced than what is used in computer science
  - only the simple parts got used.
  - pattern languages were skipped.

Why Patterns?

- Smart
  - Elegant solutions that a novice would not think of
- Generic
  - Independent on specific system type, language
- ***Well-proven***
  - Successfully tested in ***several*** systems
- Simple
  - Combine them for more complex solutions

- There are really stupid patterns (supersuper) in some books so watch out!!!

Patterns provide...

- ***Reusable*** solutions to ***common*** problems
  based on experiences from real systems
- ***Names*** of abstractions above class and object level
  a common vocabulary for developers
- Handling of functional and non-functional aspects
  - separating interfaces/implementation, loose coupling between parts, …
- A basis for ***frameworks*** and toolkits
  basic constructs to improve reuse
- Education and training support

Elements in a Pattern

- Pattern **name**
  Increase of design vocabulary
- **Problem** description
  When to apply it, in what context to use it
- **Solution** description (generic !)
  The elements that make up the design, their relationships, responsibilities, and collaborations
- **Consequences**
  Results and trade-offs of applying the pattern

Example

The composite pattern...

Open the other file :)

Patterns...

Categories of Design Patterns

- Creational Patterns
  - Instantiation and configuration of classes and objects
- Structural Patterns
  - Usage of classes and objects in larger structures, separation of interfaces and implementation
- Behavioral Patterns
  - Algorithms and division of responsibility
- Concurrency
- Distribution
- Security

Some Creational Patterns

- ***Abstract factory***
- Builder
- Factory Method
- Prototype
- ***Singleton***

Some Structural Patterns

- Adapter
- Bridge
- ***Composite***
- Decorator
- Façade
- Flyweight
- Proxy

Some Behavioral Patterns

- ***Chain of responsibility***
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- ***Strategy***
- Template Method
- ***Visitor***

Alert!!! Design Patterns are invading

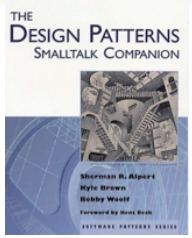
- Design Patterns may be a real ***plague***!
- Do not apply them when you do not need them

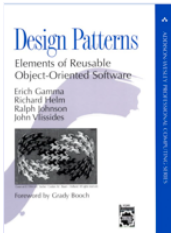
- Design Patterns make the software more complex
  - More classes
  - More indirections, more messages
- Try to understand when NOT applying them!

About Pattern Implementation

- This is ***POSSIBLE*** implementation not a definitive one
- Do not confuse structure and intent!!!
- Patterns are about ***INTENT***
  and ***TRADEOFFS***

Source





12 of 12 people found the following review helpful:

★★★★★ **Easier to understand than the original GoF**, February 4, 2000
Reviewer: **Nicolas Weidmann** (Zurich, Switzerland) - See all my reviews
This book gives you a better understanding of the patterns than in its original version (the GoF one). I am not a SmallTalk programmer but a 9 years C++ one. At work I had to use the GoF book and never liked reading it. In contrast to this, the SmallTalk companion is easy to read and you can understand the patterns within the first few lines of their description. Take the Bridge pattern and compare their discussions in the two books. If you really like the Gof one then buy it. But according to me, it would be a big mistake buying the GoF in favour of the SmallTalk companion. Trust a C++ programmer :-)

Was this review helpful to you? Yes No (Report this)

Wrap-up

Patterns are names

Patterns are about tradeoffs
Know when not to apply them