Design Patterns
Not Just for Architects

Presented by Jeremy Clark
www.jeremybytes.com

What Are Design Patterns?

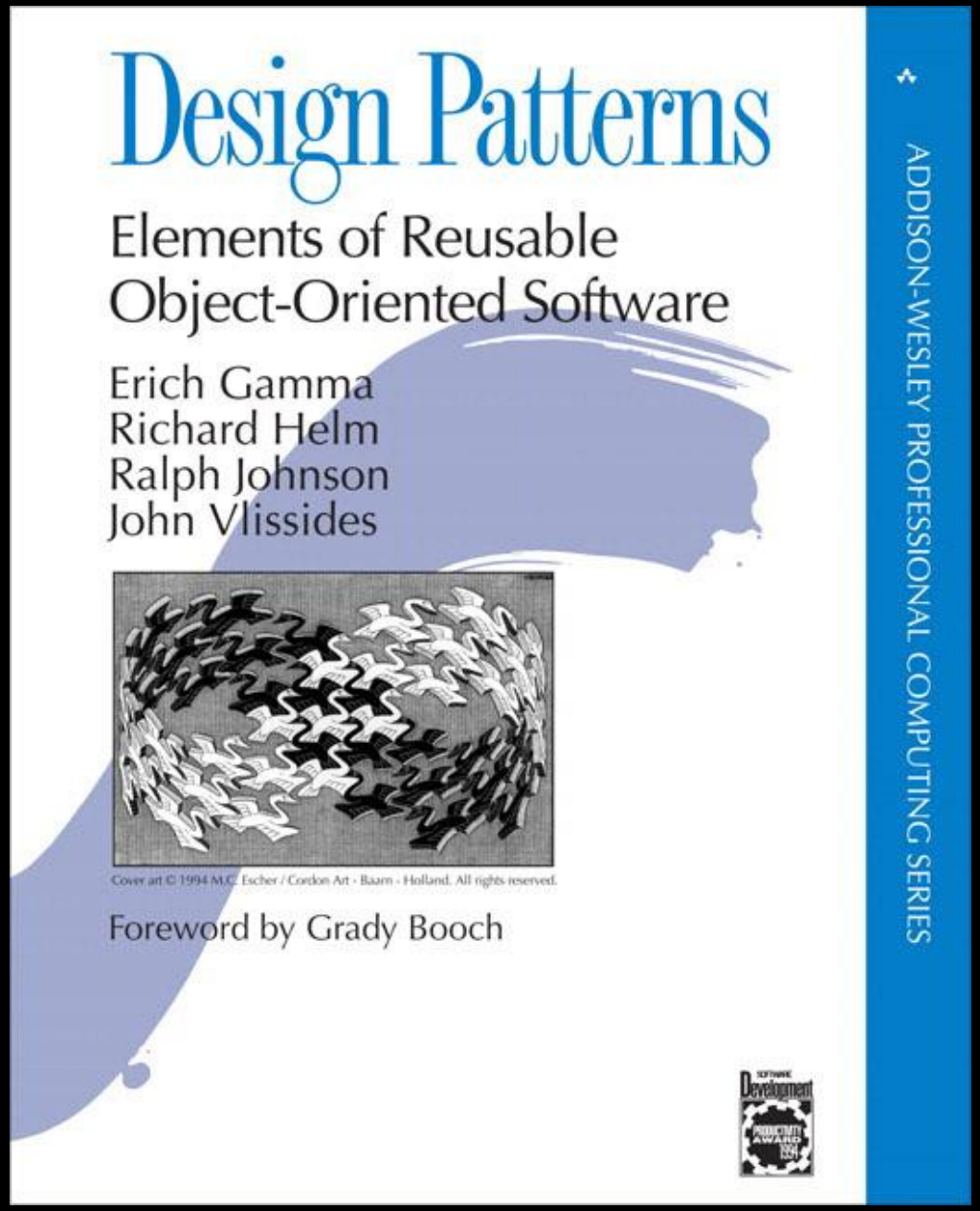
"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to the problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."

-- Christopher Alexander

The Gang Of Four (GoF)

Design Patterns:
Elements of Reusable
Object-Oriented Software

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



Anatomy Of A Design Pattern

- Pattern Name
  - A unique way of referring to the pattern
- Problem
  - The problem that occurs "over and over again"
- Solution
  - The "core" of the solution
- Consequences
  - The drawbacks or considerations when using the pattern

The GoF Patterns

Creational Patterns

- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton

Structural Patterns

- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy

Behavioral Patterns

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

A More Approachable Reference

- Head First Design Patterns (Second Edition)
  - Eric Freeman
  - Elisabeth Robson

Covers 12 GoF Patterns

Why Should We Care?

- Well described solutions
- Shared vocabulary
- Concise language

- Think in design rather than implementation
- Encourage other developers to learn patterns

Observer

GoF Description:

"Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically."

Real World Observer

- Twitter

Observer Consequences

- We don't know how many times we will be notified. It could be 1 or it could be 100,000.
- We don't know how frequently we will be notified. It could be once per day or 100 times per ms.
- We don't know if we will be notified at all.

Iterator

GoF Description:

"Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation."

Real World Iterator

- TV Remote

Iterator Consequences

- Adding or removing items during iteration can invalidate the iterator.

- Multiple iterators can work on a single object.

Facade

GoF Description:

"Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use."

- Play Blu-ray
  - Turn on TV
  - Set TV to Blu-ray player
  - Turn on sound bar
  - Set sound bar to digital in
  - Turn on Blu-ray player
  - “Play” on Blu-ray player

@jeremybytes

Real World Facade

Facade Consequences

- If functionality is added to the sub-system, we may need to change the facade as well.

- Not all functionality of the sub-system may be available. The available functions in the facade are probably limited.

Chain Of Responsibility

GoF Description:

"Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it."

Real World Chain Of Responsibility

- Tech Support

Chain of Responsibility Consequences

- If none of the receivers handle the message, then the message will fall off the end of the chain.

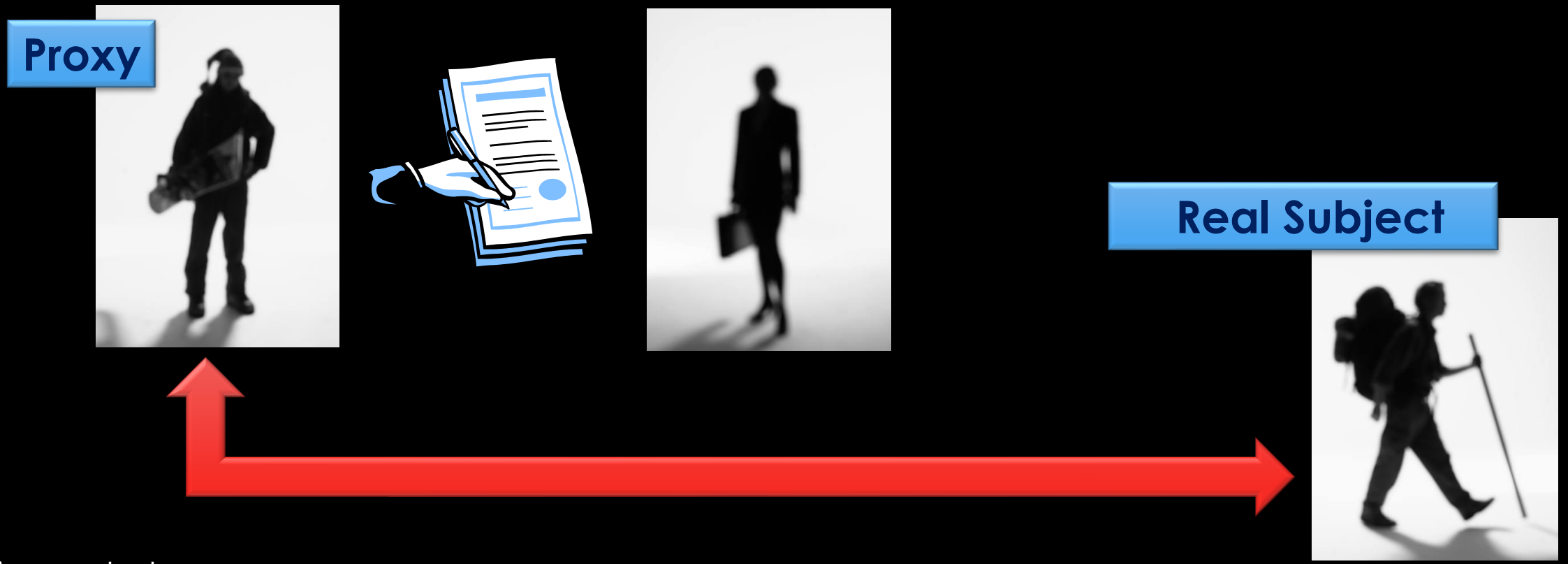
Proxy

GoF Description:

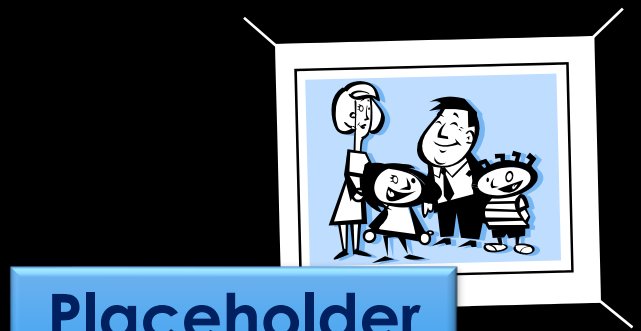
"Provide a surrogate or placeholder for another object to control access to it."

Real World Proxy

- Power of Attorney



Interesting History



Placeholder

Click

Original

Proxy Consequences

- A remote proxy can hide the fact that the object resides in a different process or across a network.
- The proxy must be kept synchronized with the real object.

A Million Implementations

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to the 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

Why Should We Care?

- Well described solutions
- Shared vocabulary
- Concise language

- Think in design rather than implementation
- Encourage other developers to learn patterns

Thank You!

Jeremy Clark

- http://www.jeremybytes.com
- jeremy@jeremybytes.com
- @jeremybytes