Where To Go Next

The Journey Continues

Jeremy Clark
www.jeremybytes.com
jeremy@jeremybytes.com

pluralsight
hardcore developer training

A Million Implementations

Each pattern describes a problem that 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 the solution a million times over, without ever doing it the same way twice.

-Christopher Alexander

Alexander, et al, *A Pattern Language*. Oxford University Press, 1977.

Observer Implementations

- Event Handler

```
ClickMeButton.Click += Observer1;

void Observer1(object sender, RoutedEventArgs e)
{
  TextBlock1.Text = "Hello from Observer 1";
}
```

- EventAggregator (from the Prism library)

```
eventAggregator.GetEvent<StockTickerEvent>()
      .Subscribe(ProcessStockTickerData,
            ThreadOption.UIThread, false,
            p => p.StockId == this.stockFilter);
```

- IObservable<T> and IObserver<T> (from .NET 4.0+)

Common Question

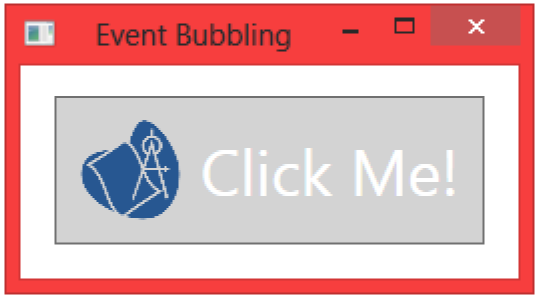
- **Is [insert programming implementation here] an example of the [insert pattern name here]?**

- **Example:**
  Is event bubbling in WPF an example of the Chain of Responsibility Pattern?
  Or is it an example of the Observer Pattern?

Common Answer

- **Is [insert programming implementation here] an example of the [insert pattern name here] Pattern?**
- **Answer: Maybe**
- **Let’s think about it a little more:**
  - Does it try to solve the problem described?
  - Does the implementation follow the “core of the solution”?

WPF Event Bubbling



- **Unhandled Events “bubble” to the Parent**
- **Example – MouseOver the Image**
- **Event Handling Order**
  - Image
  - Button
  - Window

Event Bubbling

- **Is event bubbling in WPF an example of the Chain of Responsibility Pattern?**
  - YES

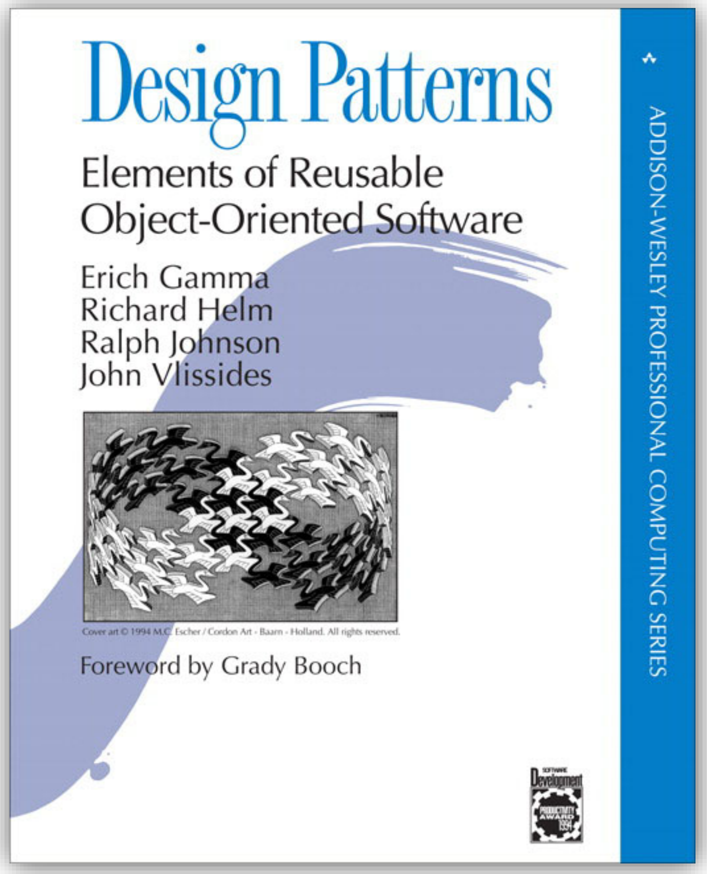
- **Or is it an example of the Observer Pattern?**
  - Probably Not

1994?

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

***Design Patterns:
Elements of Reusable
Object-Oriented Software***

ISBN: 978-0-201-63361-0


Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Built-in Implementations

- **Observer**
  - Event Handler
  - EventAggregator
  - IObservable<T>

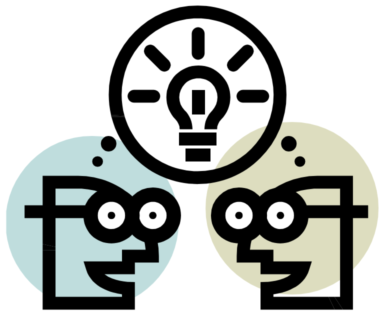
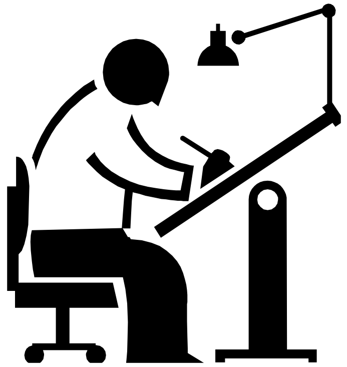
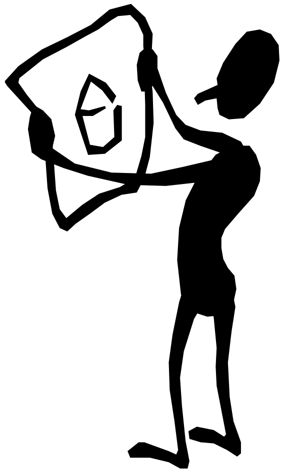
- **Iterator**
  - IEnumerable<T>
  - Built in to almost every .NET collection
  - foreach

Common Problems

Getting a notification when a process completes or a state changes.	Observer
Making a complex API easier to work with.	Facade
Adding functionality to an existing object.	Decorator
Behaving in distinct ways based on a current mode or state.	State
Looping through elements of a collection or sequence.	Iterator

Why Should We Care?

- Well-Described Solutions
- Shared Vocabulary
- Concise Language
- Stay in Design Mode Longer
- Encourage Other Developers

Gang of Four 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

Other Design Patterns

- **Repository**
  - Create Read Update Delete (CRUD)
  - Command Query Responsibility Separation (CQRS)

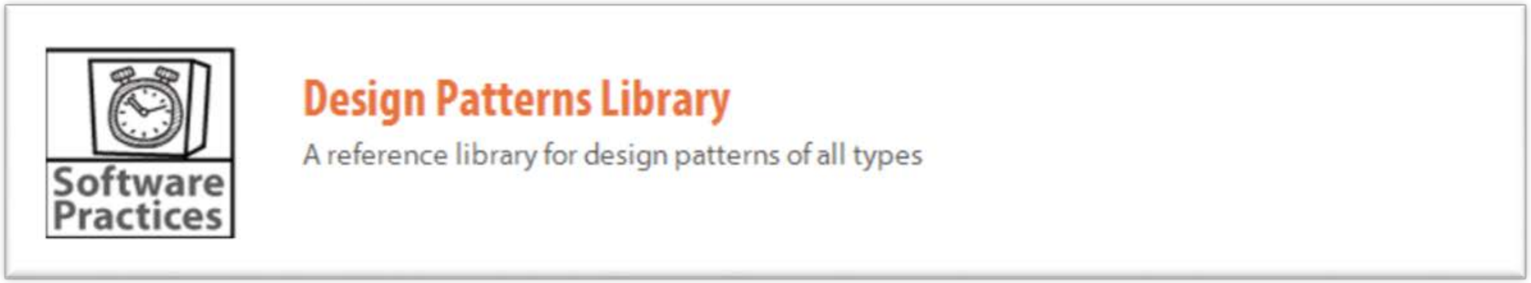
- **Inversion of Control (IoC)**
- **Dependency Injection (DI)**

- **Model-View-ViewModel (MVVM)**
- **Model-View-Controller (MVC)**
- **Model-View-Presenter (MVP)**

Good Resources

- Robert C. Martin (Uncle Bob)
- Martin Fowler
- Dino Esposito
- Joshua Kerievsky

A Good Next Stop



Software Practices

Design Patterns Library

A reference library for design patterns of all types

Design Pattern Dangers

- **New and Shiny**
- **Design Pattern Attachment**
- **Design Pattern Fads**

Recommended Reading:

“On the use and misuse of patterns” by Rockford Lhotka

http://goo.gl/5ZXpZO

Proper Use of Design Patterns

Only use a pattern if you have the problem it solves, and the positive consequences outweigh the negative consequences.

Rockford Lhotka, "On the use and misuse of patterns."
http://goo.gl/5ZXpZO

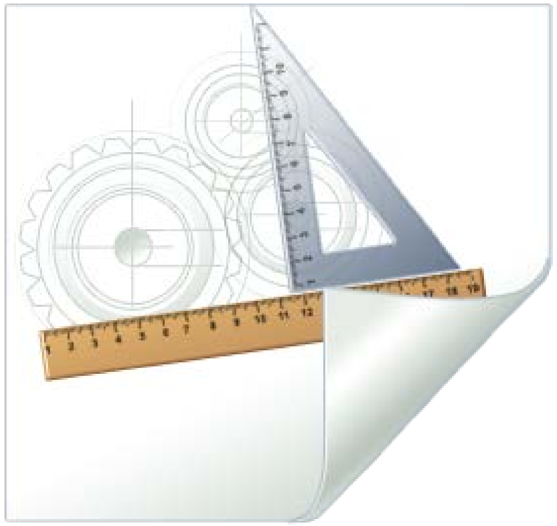
Proper Use of Design Patterns

Strive for "pattern mastery", where you are solving problems with natural solutions, not looking for ways to apply any specific pattern.

Rockford Lhotka, "On the use and misuse of patterns."
http://goo.gl/5ZXpZO

Goals Review

- **Introduction to Design Patterns**
  - What are Design Patterns?
  - Who are the Gang of Four?
  - Why do Design Patterns Matter to Me?
- **Patterns You Already Use**
- **Other Useful Patterns**
- **Where to Go Next**



Design Pattern Overview

- **What are Design Patterns?**
  - Christopher Alexander

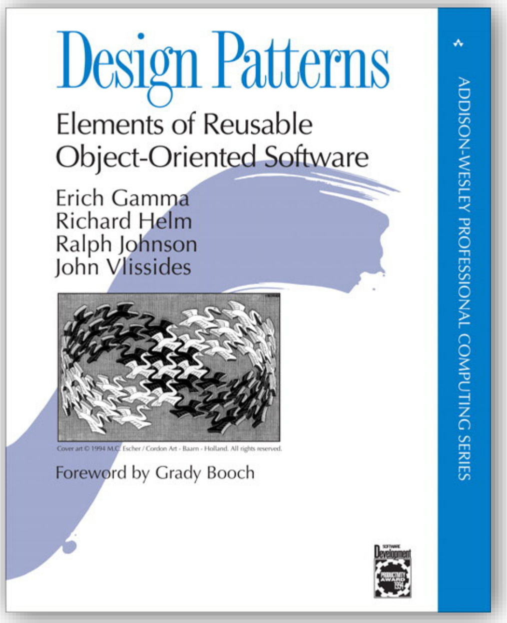
    Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use the solution a million times over, without ever doing it the same way twice.

  - Parts
    - Pattern Name
    - Problem
    - Solution
    - Consequences

Design Pattern Overview

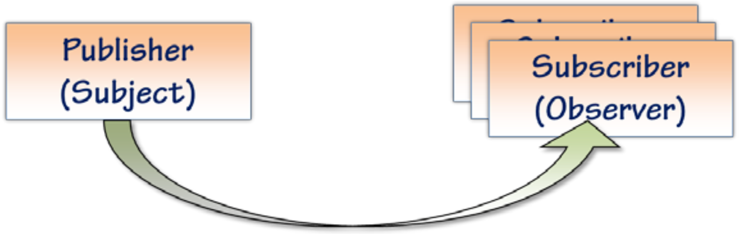
- **Who are the Gang of Four?**

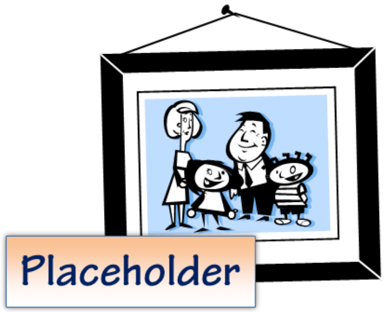
- **Why Should We Care?**
  - Well-Described Solutions
  - Concise, Shared Vocabulary
  - Stay in Design Mode Longer


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Patterns You Already Use

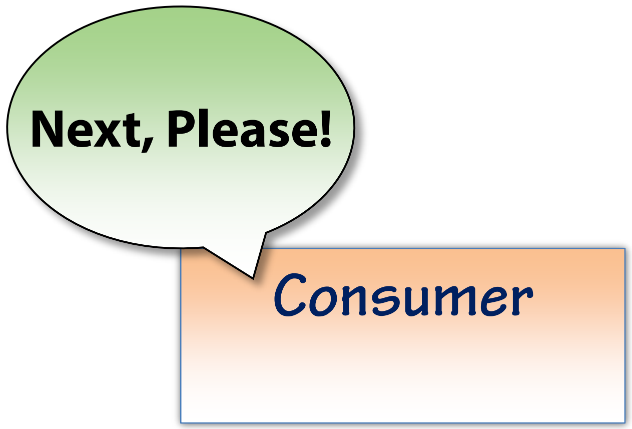
- **Observer**
  - Publish/Subscribe relationship
  - Everyday usage: Event Handlers
  - EventAggregator and IObservable<T>
- **Proxy**
  - Placeholder / Stand-in for an actual object
  - Everyday usage: SOAP Service Proxies





Patterns You Already Use

- **Iterator**
  - Next, Please!
  - Everyday usage: foreach
  - IEnumerable<T>
  - yield return
  - MP3 Library
  - LINQ



Patterns You Already Use

- **Chain of Responsibility**
  - Message Handling
  - Everyday usage: try/catch blocks
  - Approval Engine
- **Facade**
  - Hiding Complexity
  - Everyday usage: foreach
  - BackgroundWorker and other components

Other Useful Patterns

- **Factory Method**
  - Creating Objects
  - RepositoryFactory
- **Decorator**
  - Adding Functionality
  - CachingRepository
- **Adapter**
  - Resolving Incompatible Interfaces
  - ApplicationPerson

What's Next

- Is X an example of the Y pattern?
- Are the Gang of Four Patterns (from 1994) still relevant?
- Design Pattern Benefits
- Patterns beyond the Gang of Four
- The Use and Misuse of Design Patterns

Design Patterns On-Ramp

An Introduction to Patterns

Jeremy Clark
www.jeremybytes.com
jeremy@jeremybytes.com

pluralsight
hardcore developer training