View Source: Design Patterns in the Wild

Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

Design

- Algorithm
  - Sequence of steps to solve a computational problem
- Design Pattern
  - Patterns in structure of code to solve a design problem

Why Design Patterns?

- Common solutions to design problems
- Shared language of design

Use to enable...

- Reusability
- API Design
- Modularity
- High Cohesion and Low Coupling

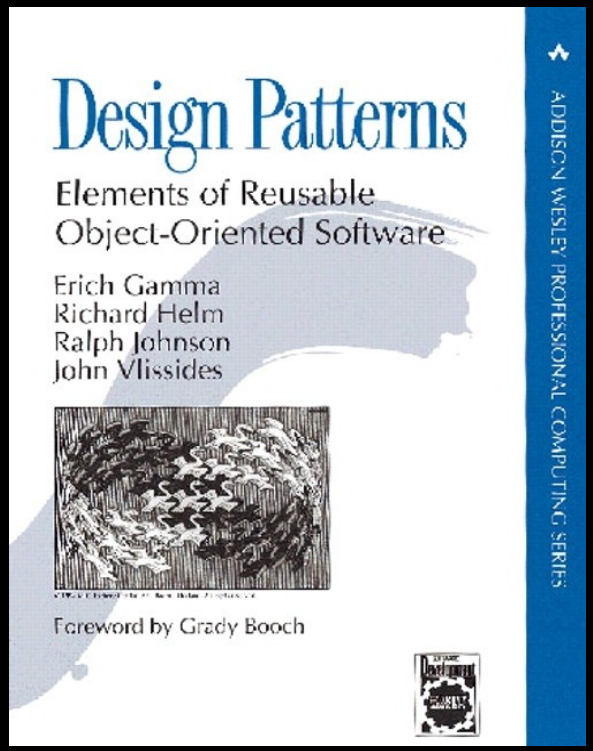
“When people begin to look at design patterns, they often focus on the solutions the patterns offer. This seems reasonable because they are advertised as providing good solutions to the problems at hand.

However, this is starting at the wrong end. When you learn patterns by focusing on the solutions they present, it makes it hard to determine the situations in which a pattern applies. This only tells us what to do but not when to use it or why to do it.”

Alan Shalloway

GoF Patterns

"Design Patterns: Elements of Reusable Object-Oriented Software" by Gamma, Helm, Johnson and Vlissides



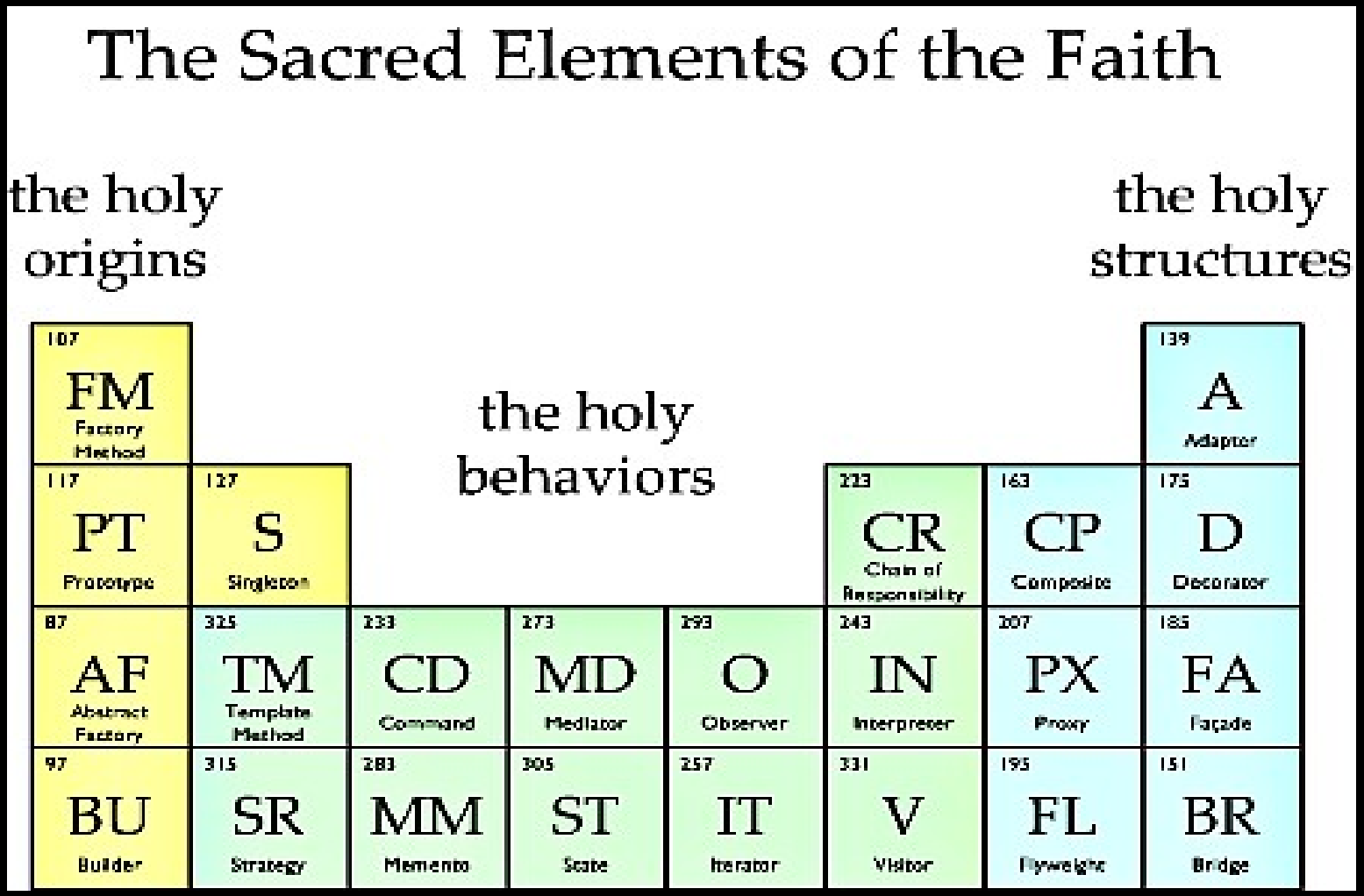
Creational: Singleton, Factory, ...

Structural: Adapter, Proxy, ...

Behavioural: Command, Observer, ...

Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

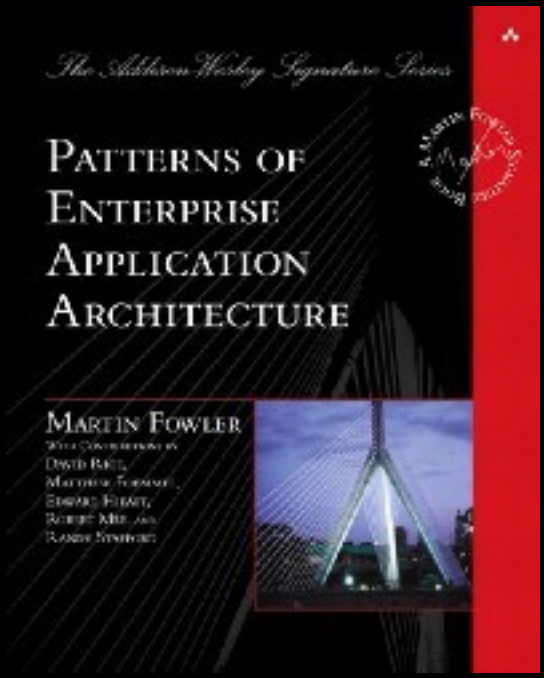
The 23 GoF Patterns



From http://www.vincehuston.org/dp/

Enterprise Application Patterns

"Patterns of Enterprise Application Architecture" by Martin Fowler



- Domain Logic
- Data Source
- Object-Relational
- Web Presentation
- Offline Concurrency
- Session State

...and more

Ashwin J Mathew
School of Information, UC Berkeley

Spring 2008

Distributed Computing Patterns

- Publish-Subscribe (Pub-Sub)
- Pipeline
- Blackboard
- Scatter-Gather
- Map-Reduce
- Comet

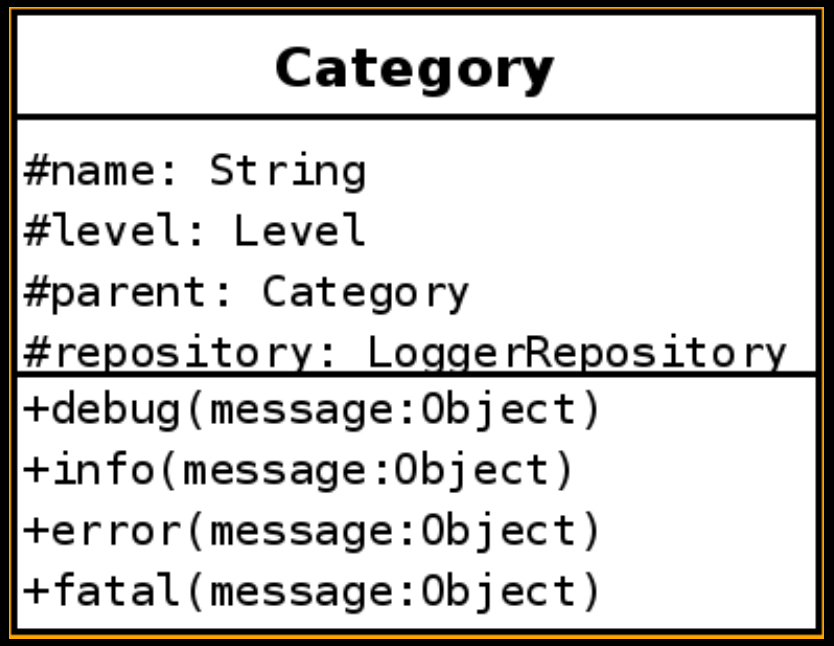
Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

Object Oriented Analysis and Design (OOAD) Review with Log4J

Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

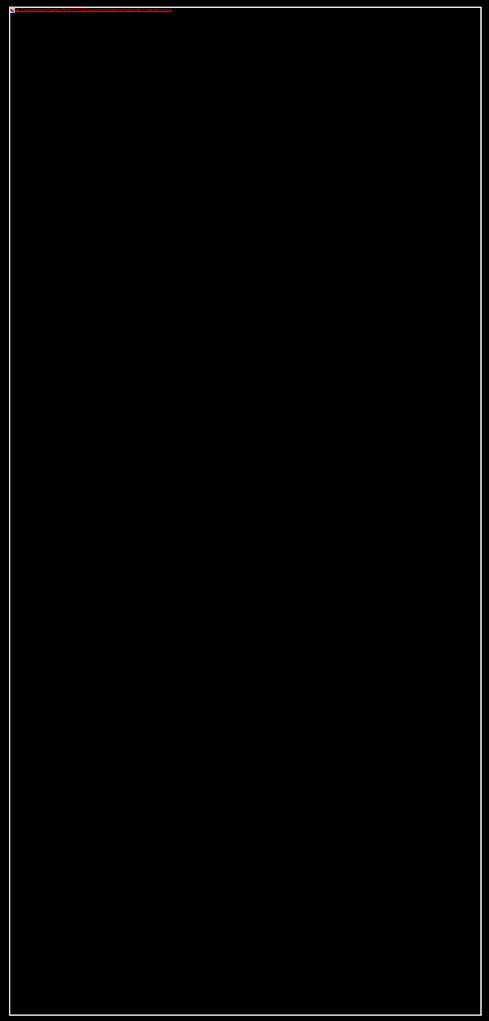
Encapsulation

- “Information hiding”
- Hide state
- Expose behaviour

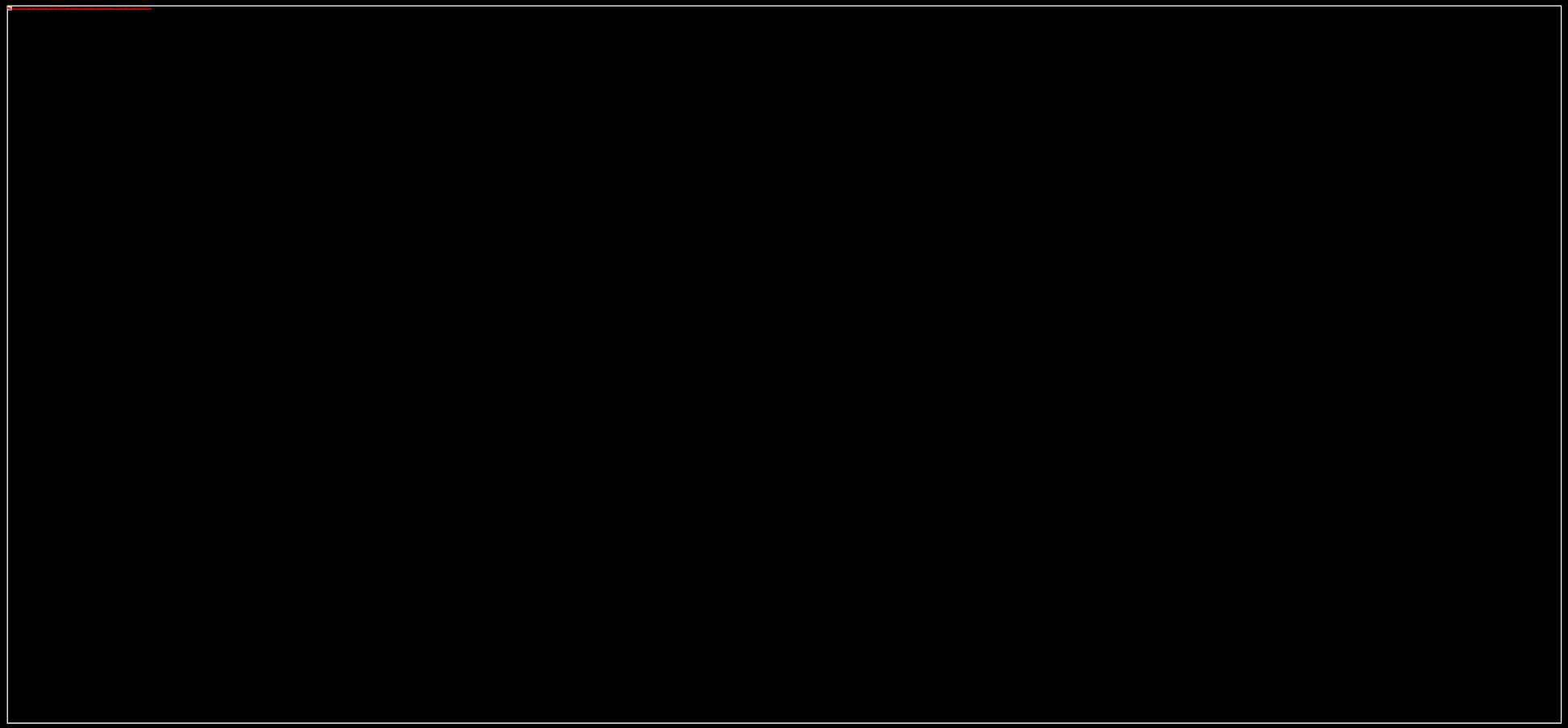


Inheritance

- Inherit attributes and behaviour
- Type hierarchy: subclass is-a superclass



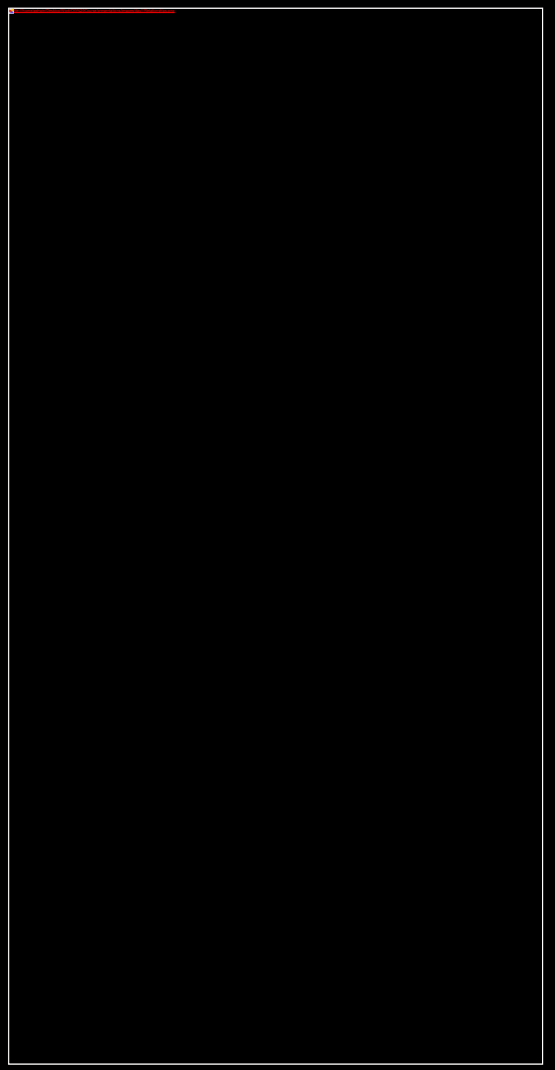
Abstraction and Polymorphism



Ashwin J Mathew Spring 2008
School of Information, UC Berkeley

Association, Aggregation, Composition

- Association: “uses-a”
- Aggregation: “has-a”
- Composition
  - Strong aggregation
  - Part lives and dies with whole



"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, Architect

Next Class

- More Log4J
- Read “A Short Introduction to Log4J”
- Get familiar with Log4J source code (available on course website)
- Background readings on GoF patterns from http://www.vincehuston.org/dp/
- Course website: http://courses.ischool.berkeley.edu/i290-1/s08