

Introduction to Design Patterns

Human Computer Interaction Research
University of Nevada, Reno



Course overview

- ★ **Teacher**
- ★ **Teaching Objectives**
- ★ **Book**
- ★ **Lecture**
- ★ **Homework**

Teacher



Eelke Folmer

SEM 208

Office hours:

W 3.30PM - 5.30 PM

Other times by appointment

Email: cs330@eelke.com

website: <http://cs330.eelke.com>

**Player Game Interaction Research
University of Nevada, Reno**

research interests

human-computer-interaction software engineering
interaction design patterns **accessibility** exergames
haptics software architecture **virtual worlds** motor
impaired usability **games** visually impaired multimodal
feedback human navigation

Past Research

★ **European Research Project: STATUS (Software Architecture that supports Usability)**

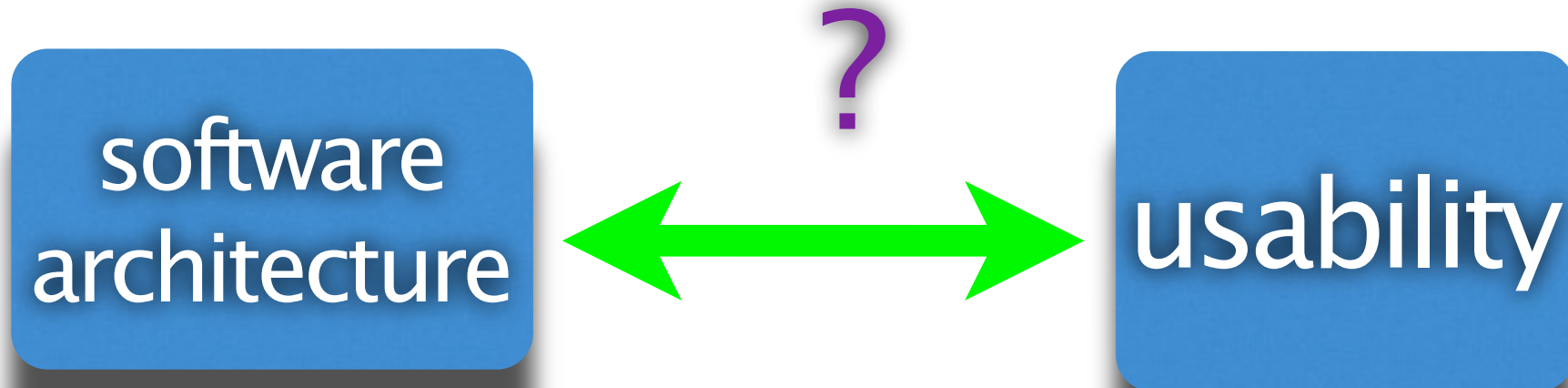
★ **Duration: 2001 to 2004**



★ **International Partners:**

- » **Technical University of Madrid (Spain)**
- » **University of Groningen (Netherlands)**
- » **Imperial College (England)**
- » **IHG (Spain)**
- » **LogicDIS (Greece)**

relationship



- ★ adding certain usability features (undo) is often expensive during late stage
- ★ can we assess an architecture for its support of usability?
- ★ which solutions affect SA & improve usability?

results

SALUTA

SAUF Framework

★ see <http://www.eelke.com/publications.html>

Topics

- ★ **Introduction to Design Patterns**
- ★ **Observer pattern**
- ★ **Decorator pattern**
- ★ **Factory pattern**
- ★ **Singleton pattern**
- ★ **Command pattern**
- ★ **Adapter and Facade pattern**
- ★ **Template method pattern**
- ★ **Iterator and composite patterns**
- ★ **State Pattern**
- ★ **Proxy Pattern**
- ★ **Compound Patterns**
- ★ **Model View Controller**
- ★ **Anti patterns, etc**

Course Objectives

★ for each pattern you will be able to:

- » **Indicate which underlying object oriented design principle(s) it is based on.**
- » **Explain the reasoning for each object oriented design principle.**
- » **Explain what specific object oriented design problem the pattern solves.**
- » **Provide a specific context for each pattern in which it can be applied.**
- » **Draw a high level class diagram in UML for each pattern.**
- » **Explain how the different components of the pattern collaborate with each other.**
- » **List the consequences of applying each pattern to the overall software quality of a system.**
- » **List which patterns are related to this pattern and what type pattern each pattern is.**
- » **Implement this pattern in Java or C# to a real world problem.**

book



Player Game Interaction Research
University of Nevada, Reno

Typical lecture

- ★ 20 min pattern presentation
- ★ 45 min exercise I
- ★ 15 min break
- ★ 10 minute discussion solution I
- ★ 45 min exercise II
- ★ 10 min discussion solution II

Pair Programming



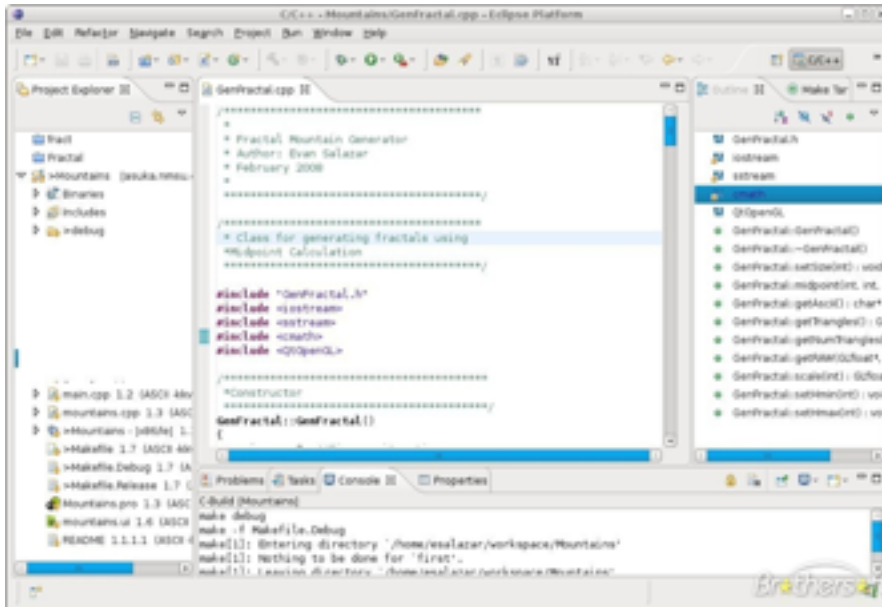
- ★ “Agile” design/programming technique
- ★ 2 Programmers 1 computer
 - » Driver: codes
 - » Navigator: reviews
- ★ Swap roles every 20 minutes

Benefits

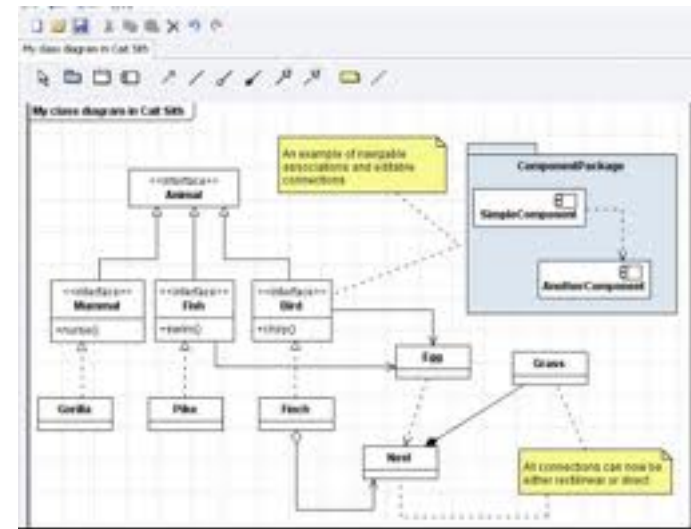
- ★ Better Quality
- ★ Higher morale
- ★ Learning & Training
- ★ Less distraction
- ★ More brainpower
- ★ Reduced cost
- ★ Lui, et al (2008) shows that pair programming outperforms for **design** tasks



Tools

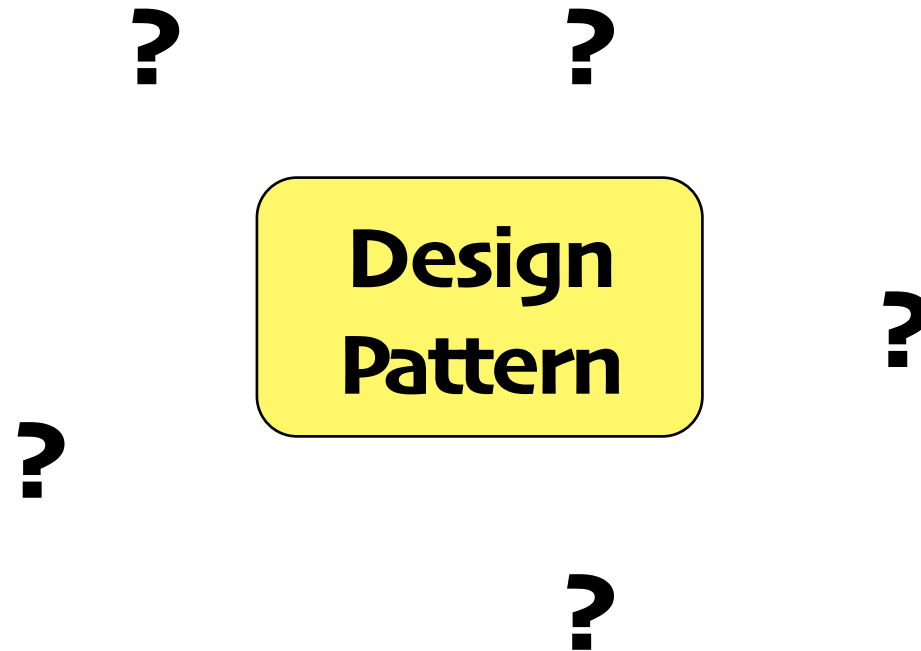


Eclipse



tinyUML

....so what is a:



what is a pattern?



- ★ Current use comes from the work of the architect Christopher Alexander
- ★ Alexander studied ways to improve the process of **designing** buildings
- ★ “Each pattern is a three-part rule, which expresses a relation between a certain **context**, a **problem** and a **solution**.”
- ★ Hence, the common definition of a pattern: “A solution to a problem in a context.”
- ★ Patterns can be applied to many different areas of e.g. business / interaction design / software development / cooking

Example



ChocolateChipRatio

- ★ **Context:** You are baking chocolate chip cookies
- ★ **Problem:** Determine the optimum ratio of chocolate chips to cookie dough
- ★ **Solution:** Use the maximum amount of chocolate chips that results in a really sturdy cookie.
- ★ **Rationale:**
 - » chocolate is the best part of the chocolate chip cookie.
 - » Too much chocolate may prevent the cookie from holding together

Pattern Language



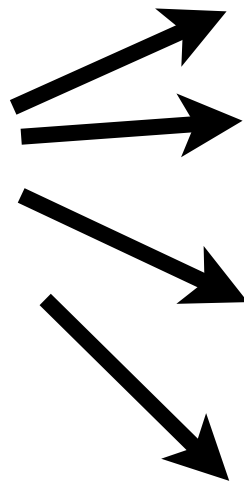
CookieHeat

DoughRatio

ChocolateChipRatio

- ★ **Context:** You are baking chocolate chip cookies
- ★ **Problem:** Determine the optimum ratio of chocolate chips to cookie dough
- ★ **Solution:** Use the maximum amount of chocolate chips that results in a really sturdy cookie.
- ★ **Rationale:**

**common
elements**



Shared Vocabulary

“use: **ChocolateChipRatio**”



ChocolateChipRatio

- ★ **Context:** You are baking chocolate chip cookies
- ★ **Problem:** Determine the optimum ratio of chocolate chips to cookie dough
- ★ **Solution:** Use the maximum amount of chocolate chips that results in a really sturdy cookie.
- ★ **Rationale:**

What is a pattern and what is not

- ★ **Generic Solution**

- » Use a ratio of 1 to fournot a pattern

- ★ **Highest level of abstraction**

- ★ **Repeatable** for different contexts

- » chocolate muffin

- » chocolate bread

Software

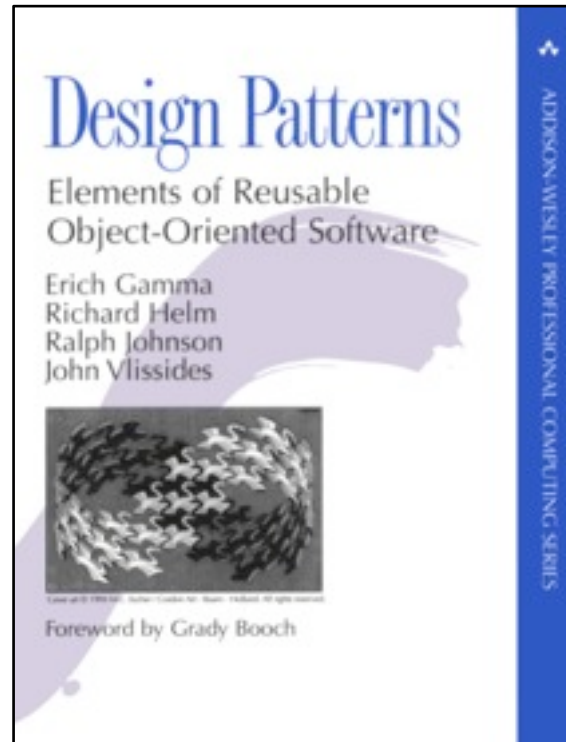


Software Design



**Human Computer Interaction Research
University of Nevada, Reno**

1995



★ **Gang of Four (GoF) publish the Design Patterns book**

Human Computer Interaction Research
University of Nevada, Reno

Design of Software

- ★ “Designing object-oriented software is hard and designing reusable object-oriented software is even harder.” - Erich Gamma
- ★ Experienced designers **reuse** solutions that have worked in the past
- ★ Well-structured object-oriented systems have **recurring** patterns of classes and objects
- ★ Knowledge of the patterns that have worked in the past allows a designer to be more **productive** and the resulting designs to be more **flexible** and **reusable**
- ★ Patterns are subject to a form of evolution

Pattern history

- ★ 1987 - Cunningham and Beck used Alexander's ideas to develop a small pattern language for Smalltalk
- ★ 1990 - The Gang of Four (Gamma, Helm, Johnson and Vlissides) begin work compiling a catalog of design patterns
- ★ 1991 - Bruce Anderson gives first Patterns Workshop at OOPSLA
- ★ 1994 - First Pattern Languages of Programs (PLoP) conference
- ★ 1995 - The Gang of Four (GoF) publish the Design Patterns book
- ★ 1995-2000 - Patterns increase in Popularity
- ★ 2001-2006 - Pattern movement slowly ``dies``
- ★ 2004 Publication of Head First Book on DP
- ★ 2008 Patterns get popular again.

Types of Software Patterns

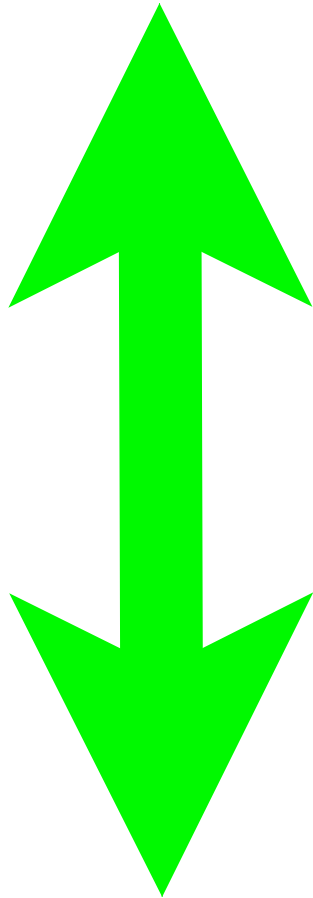
- ★ Analysis
- ★ **Design**
- ★ Organizational
- ★ Process
- ★ Project Planning
- ★ Configuration Management
- ★ Interaction Design

types of design patterns

- ★ Riehle and Zullighoven in “Understanding and Using Patterns in Software Development” mention three types of design patterns:
 - » **Architecture Pattern** whose form is described by means of terms and concepts from the application domain
 - » **Object oriented Design Pattern** whose form is described by means of software design constructs, such as objects, classes, inheritance and aggregation
 - » **Programming Pattern** (Idiom) whose form is described by means of programming language constructs (linked list)

Levels of Abstraction

Abstract



Specific

★ **Complex design for an entire application or subsystem**

★ **Solution to a general design problem in a particular context**

★ **Simple reusable design class such as a linked list, hash table, etc.**

Human Computer Interaction Research
University of Nevada, Reno

Gang of 4 Design Patterns

- ★ **The GoF design patterns are in the middle of these levels of abstraction.**
- ★ **“A design pattern names, abstracts, and identifies key aspects of a common design structure that makes it useful for creating a reusable object-oriented design.”**
- ★ **The GoF design patterns are “descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context.”**

Classification of GoF patterns

★ Purpose - what a pattern does:

- » **Creational Patterns - Concern the process of object creation**
- » **Structural Patterns - Deal with the composition of classes and objects**
- » **Behavioral Patterns - Deal with the interaction of classes and objects**

★ Scope - what the pattern applies to:

» **Class Patterns**

- ▶ **Focus on the relationships between classes**
- ▶ **Involve inheritance reuse**

» **Object Patterns**

- ▶ **Focus on the relationships between objects**
- ▶ **Involve composition reuse**

Pattern Elements I

★ Name

- » Having a concise, meaningful name for a pattern improves communication among developers

★ Problem

- » What is the problem and context where we would use this pattern?
- » What are the conditions that must be met before this pattern should be used?

★ Solution

- » Description of the elements that make up the design pattern
- » Emphasizes their relationships, responsibilities and collaborations
- » Not a concrete design or implementation; rather an abstract description

Pattern Elements II

★ **Consequences**

- » The pros and cons of using the pattern
- » Includes impacts on reusability, portability, extensibility

★ **Intent**

- » Short statement about what the pattern does

★ **Also Known as**

- » Other names for the pattern

★ **Motivation**

- » A scenario that illustrates where the pattern would be useful

Pattern Elements III

- ★ **Structure**

- » a graphical representation of the pattern

- ★ **Participants**

- » the classes and objects participating

- ★ **Collaborations**

- » how do the participants collaborate?

- ★ **Consequences**

- » pros and cons for using the patterns

- ★ **Implementation**

- » Hint and techniques for implementing the pattern

Pattern Elements IV

- ★ **Sample code**

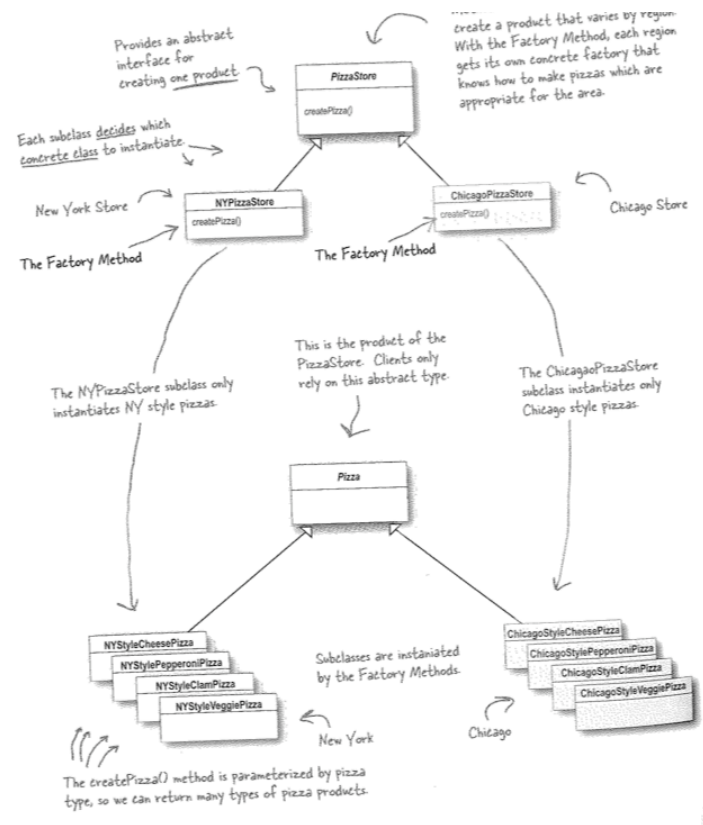
- » - Code fragments for a sample implementation

- ★ **Known uses**

- » - examples of the pattern

- ★ **Related patterns**

Describing Patterns



★ GoF: OMT (precursor UML)

★ HF: UML class diagram like schemes.

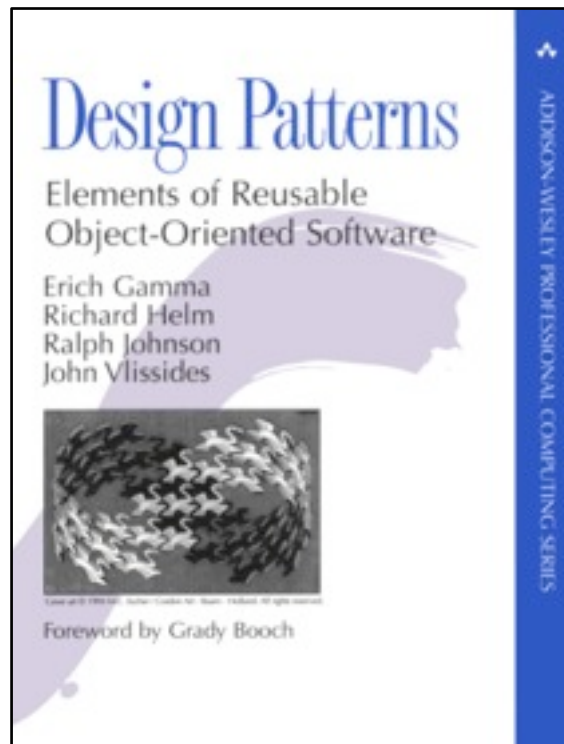
Benefits of Design Patterns

- ★ **Capture expertise and make it accessible to non-experts in a standard form**
- ★ **Facilitate communication among developers by providing a common language**
- ★ **Make it easier to reuse successful designs and avoid alternatives that diminish reusability**
- ★ **Facilitate design modifications**
- ★ **Improve design documentation**
- ★ **Improve design understandability**

So why aren't we using this book?

boring

**few code
examples**



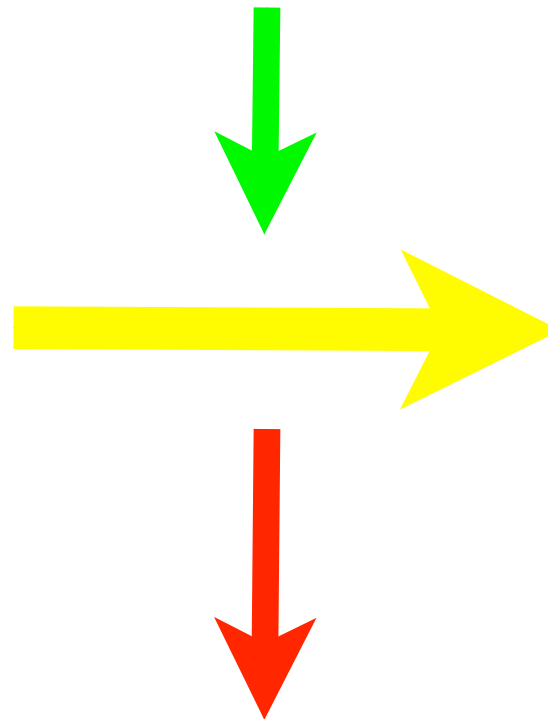
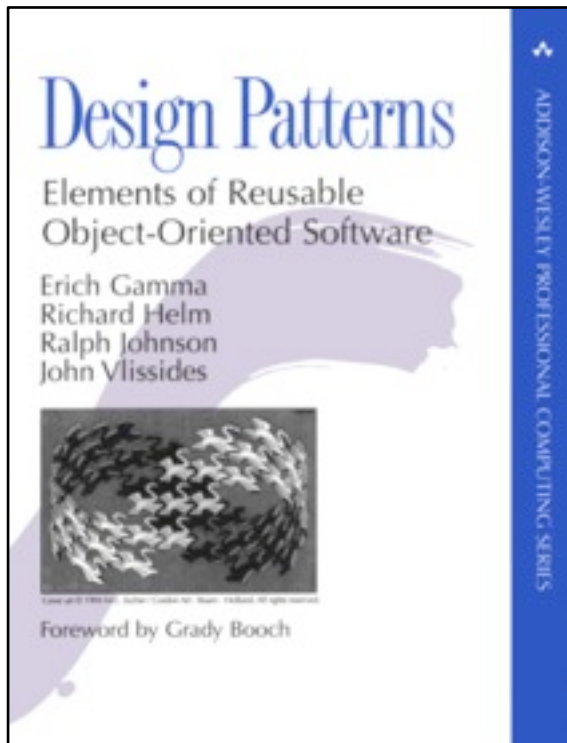
outdated

**too much
information**

Player Game Interaction Research
University of Nevada, Reno

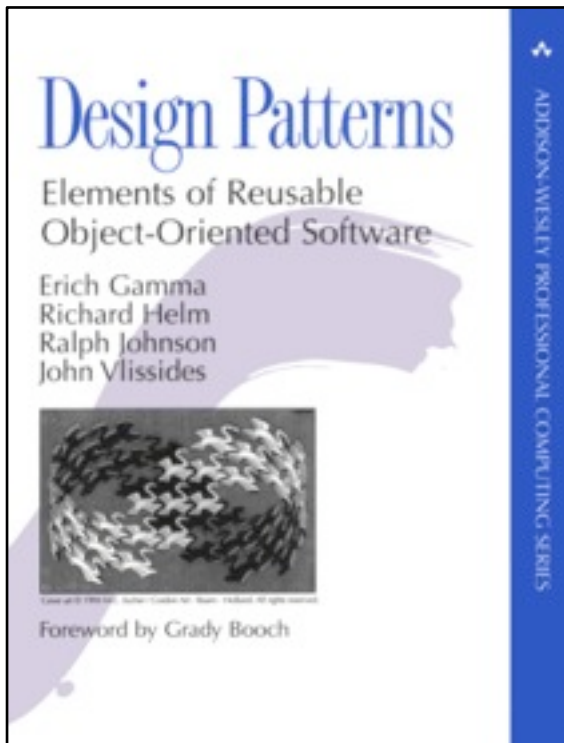
Pattern history

++FUN stuff

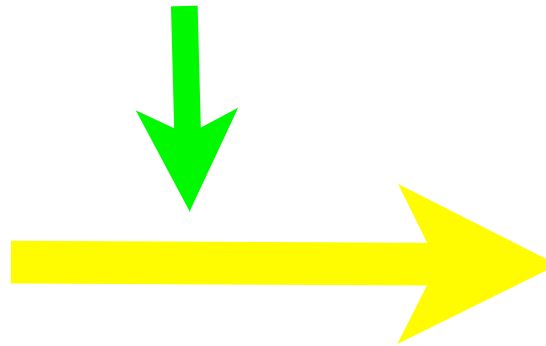


--Boring stuff

Learning based approach



- ★ **Make it Visual**
- ★ **personal style**
- ★ **challenge reader**
- ★ **keep attention**
- ★ **emotional content**



amazon: ★★★★★

amazon: ★★★★★

Player Game Interaction Research
University of Nevada, Reno