Tecniche di Progettazione: Design Patterns

Laurea Magistrale in Informatica, Pisa

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Some reviews…

▸ How hard these steps!

▸ I believe that the riser / tread ratio is one of the most uncomfortable I've ever tried! With or without heels. It seems to me that the bridge is mainly used by people in "rush", being the main link between … and … People "in the running," people who "must go" and that after the first step is forced to reduce his pace, to slow down, becoming a unique clumsiness with the trolley and bags.

Some reviews ...(cont'd)

▸ The steps of the bridge have a relationship riser and tread too variable between them that makes the transit quite annoying.

▸ The bridge has proven dangerous, since its structure with uneven steps creates perceptual barriers that deceive also the eye of those who see well and the leg of people with a good mobility, and nearly ten people ended up in the hospital.

Pattern: the step

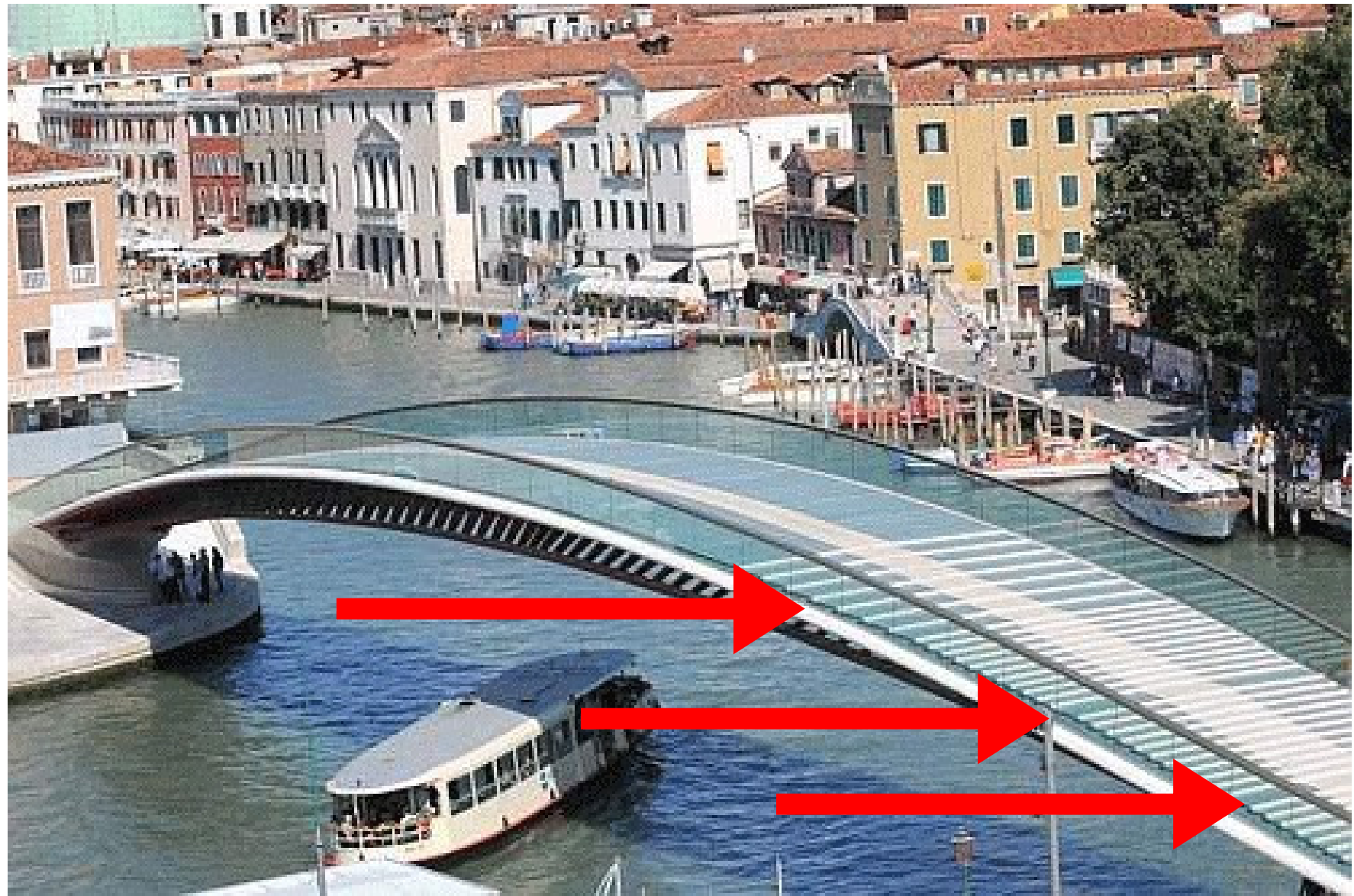▸ The riser is typically between 13 and 20 cm

▸ The tread is calculated using:

    Blondel formula: 2Raiser + Tread = 62 ÷ 64 cm

▸ By varying the slope, the user tends to change the length of the step so that the work done to overcome a step is equal to the work done to accomplish the same step on a plane.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Tread = 50, Raiser = 7 → 50+2x7=64

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Pattern: the venetian bridge

▸ The profile semiarch is characteristic of the bridges of Venice to permit the passage of boats.

▸ If Calatrava had been more humble would have asked any mason (carpenter) in Venice how they have always solved the problem"

▸ The answer would be: because the bridges follow a "fixed law" centered on the "Venetian step", an ancient unit of measurement used on purpose to make sure that we can go through a bridge with the same frequency, so they are the steps themselves that try to facilitate us.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

The use of Istrian stone alternated with dark trachyte to mark the step is a good solution already identified by the Venetians in the fifteenth century.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

                Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.
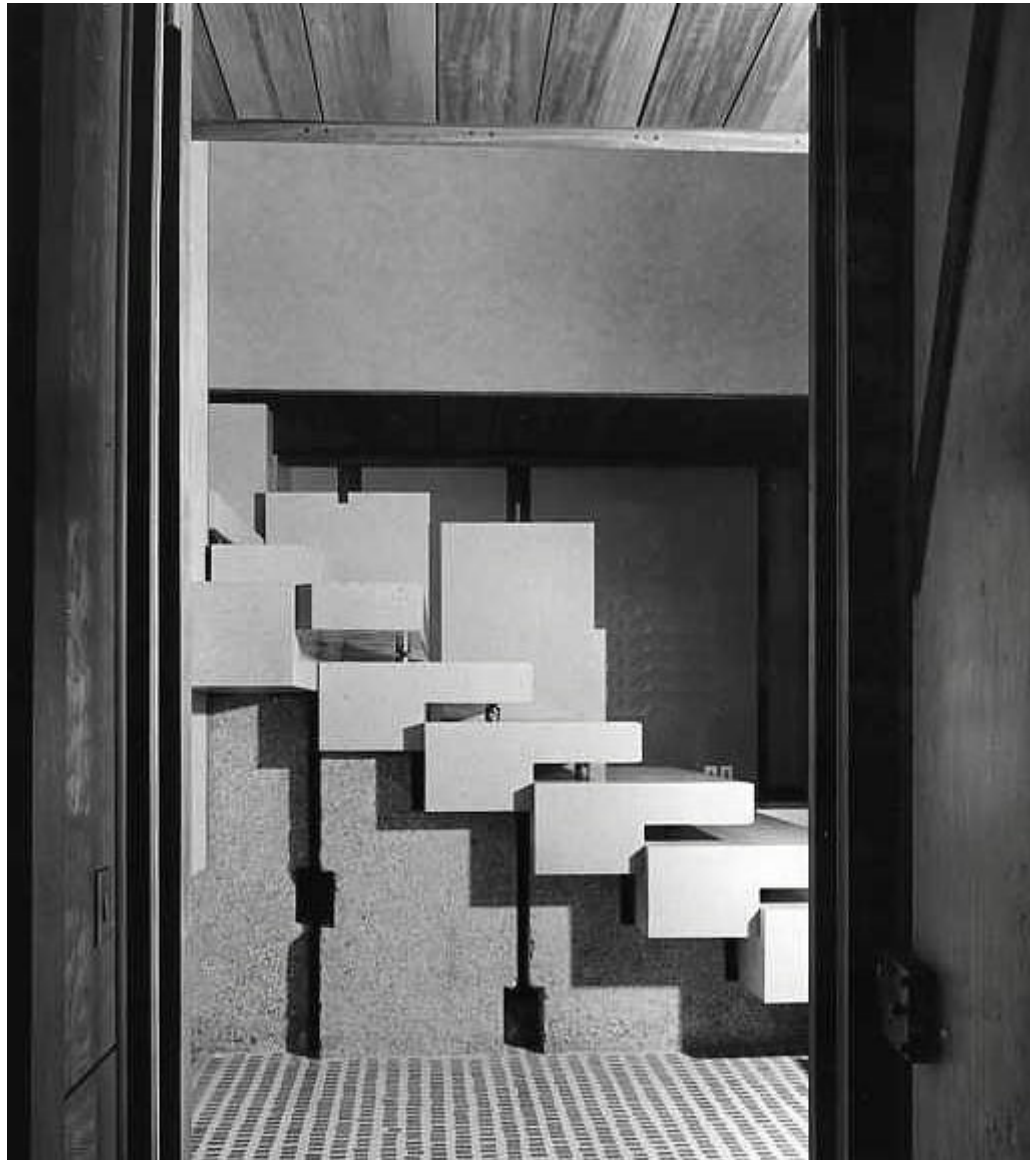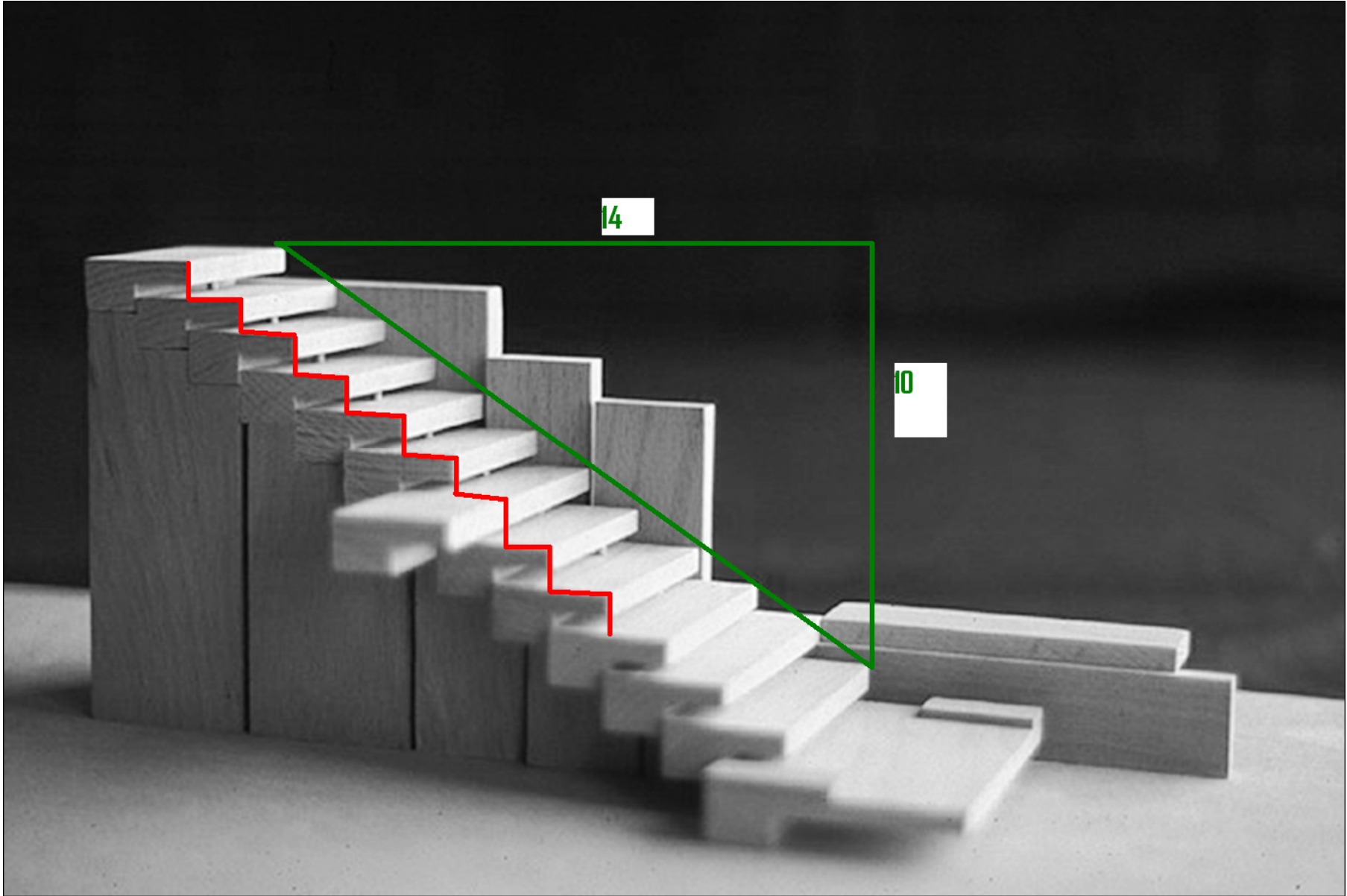
Patterns built from observation

▸ Patterns are not "created"

▸ They arise as accumulation of knowledge

▸ At some point, in the solution to a problem, there may be a "creative" step, which leads to a "revolution".

▸ Ex: elevator:

▸ With the experience of designing elevators then you get to the definition of patterns for lifts
Etc ...

▸ Revolutionary science of Thomas Kuhn
Patterns are not created or invented: they are discovered (or "mined") from empirical observation.

The design as a process that is not only creative

▸ **Carlo Scarpa told to a young architect:**

"Read a hundred pages of architecture per day!"

Christopher Alexander

Who's Who
Archives
Buildings
Paintings
Books
Film
C Vitae
Computing
Wiki
Software
Patterns

© 2001 patternlanguage.com

Christopher Alexander is Professor in the Graduate School and Emeritus Professor of Architecture at the University of California, Berkeley.

He is the father of the Pattern Language movement in computer science, and *A Pattern Language*, a seminal work that was perhaps the first complete book ever written in hypertext fashion.

He has designed and built

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 and urban areas

  - "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 human endehavour, including software development

- *A Pattern Language* was written by C. Alexander and five colleagues at the Center for Environmental Structure in Berkeley, California in the late 1970s.

What is a Design Pattern?

- "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 A Pattern Language, 1977

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Why Patterns in 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

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Software Patterns 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 & Vlissides) begin compiling a catalog of design patterns

- 1991 - First Patterns Workshop at OOPSLA

- 1993 - Kent Beck and Grady Booch sponsor the first meeting of what is now known as the Hillside Group

- 1994 – 1st Pattern Languages of Programs (PLoP) conf.

- 1995 - The Gang of Four (GoF) *Design Patterns book*

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

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

🔓 **Login**   Text Size ⊖ ⊕

THE
HILLSIDE GROUP

Site Search

Search...

MAIN MENU

🏠 **HIlltop** ⟩ History

Hillside History

HIlltop

- News
- **History**
- Hillside Europe
- Membership
- Mission Statement
- Vision
- Board

Conferences

Patterns

In August of 1993, Kent Beck and Grady Booch sponsored a mountain retreat in Colorado where a group converged on foundations for software patterns. Ward Cunningham, Ralph Johnson, Ken Auer, Hal Hildebrand, Grady Booch, Kent Beck and Jim Coplien struggled with Alexander's ideas and our own experiences to forge a marriage of objects and patterns. The Group agreed that we were ready to build on Erich Gamma's foundation work studying object-oriented patterns, to use patterns in a generative way in the sense that Christopher Alexander uses patterns for urban planning and building architecture. We then used the term "generative" to mean "creational" to distinguish them from "Gamma patterns" that captured observations. The Group was meeting on the side of a hill when all this occurred, hence the name.

Since then, the **Hillside Group** has been incorporated as an educational non-profit. It has sponsored and helped run various conferences (PlopConference, EuroPlop, ChiliPlop, KoalaPlop, Mensore PLoP, SugarloafPLoP, and UP97) and has been responsible for getting the PatternLanguagesOfProgramDesign series of books put together and published.

More history can be found on the Portland's Pattern Repository

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Types of Patterns

▶ Riehle and Zullighoven in "*Understanding and Using Patterns in Software Development*" mention three types of software patterns

▶ Conceptual Pattern

> ▶ Pattern whose form is described by means of terms and concepts from the application domain

▶ Design Pattern

> ▶ Pattern whose form is described by means of software design constructs, such as objects, classes, inheritance and aggregation

▶ Programming Pattern (Programming Idiom)

> ▶ Pattern whose form is described by means of programming language constructs

Still others

- Organizational Patterns
  - Recurring structures of relationship, usually in a professional organization, that help the organization achieve its goals.
  - The patterns are usually inspired by analyzing multiple professional organizations and finding common structures in their social networks.

- Process Patterns
  - A process pattern is a pattern which describes a proven, successful approach and/or series of actions for developing software
  - E.g. software life cycle

Design Pattern Levels Of Abstraction

▸ 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.

More Abstract
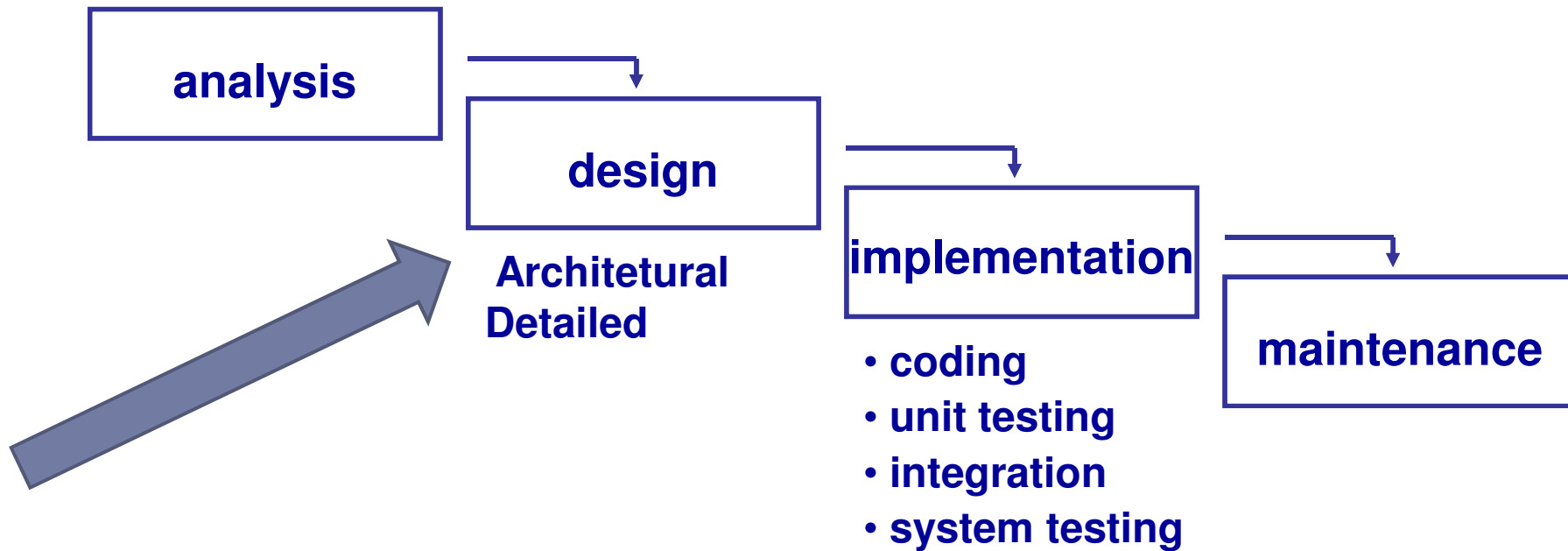
More Concrete

Architecture-Design-Code

▶ Architectural Design Patterns

▶ Design Patterns

▶ Idioms o Coding Design Patterns

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.
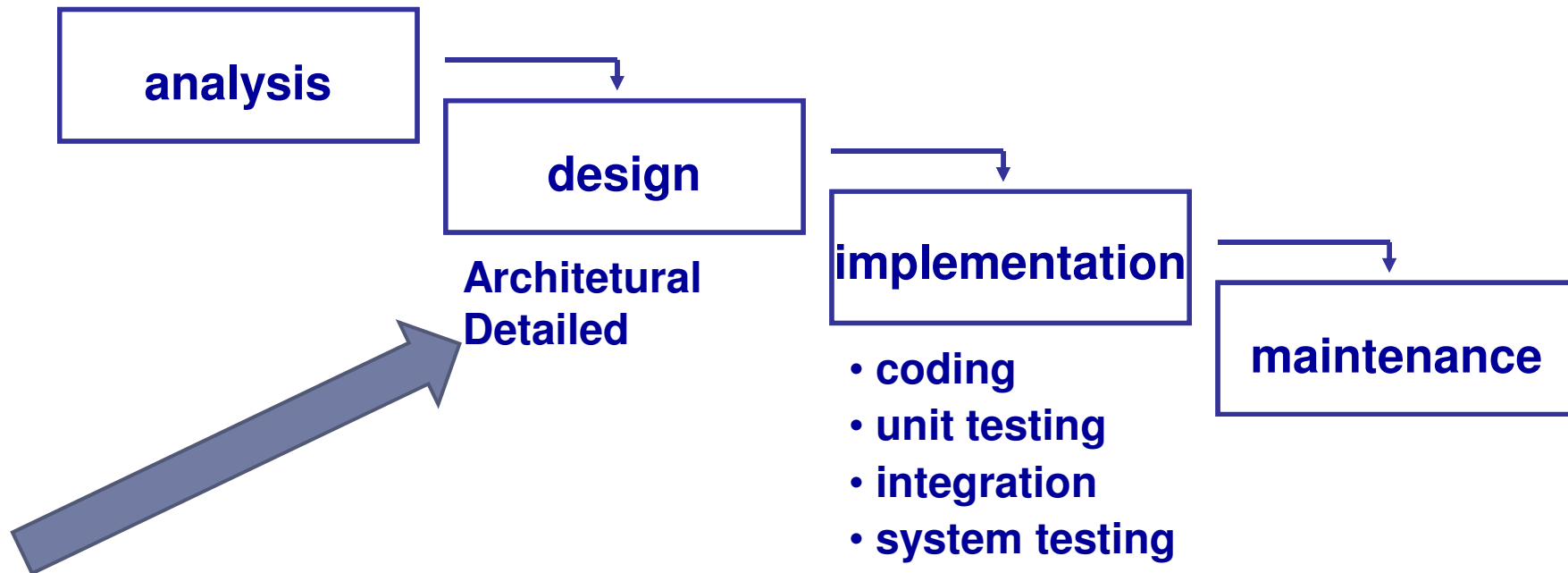
Architecture-Design-Code

▸ Architectural Design Patterns

  ▸ Thay address the architecture of a sw system

  ▸ E.g. Layers, Pipes and Filters, Blackboard, Broker, Model-View-Controller, …

```
┌──────────────┐
│   analysis   │
└──────────────┘
          ┌──────────────┐
          │    design    │
          └──────────────┘
       Architetural
       Detailed
                    ┌──────────────────┐
                    │  implementation  │
                    └──────────────────┘
                      • coding
                      • unit testing        ┌──────────────┐
                      • integration         │ maintenance  │
                      • system testing      └──────────────┘
```

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Architecture-**Design**-Code

▸ Design Patterns

    ▸ They address the design and refinement of components.

    ▸ E.g. abstract factory, decorator, …

analysis

design

**Architetural
Detailed**

implementation

• coding
• unit testing
• integration
• system testing

maintenance

Architecture-Design-**Code**

▶ Idioms o Coding Design Patterns

 ▶ An idiom is a low-level pattern specific to a programming language. An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given language.

 ▶ An Idiom is more restricted than a design pattern

  ▶ Still describes a recurring problem

  ▶ Provides a more specific solution, with fewer variations

  ▶ Applies only to a narrow context

   ☐ e.g., the C++ language

 ▶ E.g. Naming conventions, Source code formats, Memory management, Swapping variable values …

GoF 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."

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Best known families of patterns

▸ GRASP

   ▸ General Responsibility Assignment Software Patterns (or Principles) [Larman]

   ▸ Information Expert, Creator, Controller, Low Coupling, High Cohesion, Polymorphism, Pure Fabrication, Indirection, Protected Variations

▸ SOLID

   ▸ Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion

▸ GoF

   ▸ 23 design patterns

▸ POSA

   ▸ A System of Patterns: Pattern-Oriented Software Architecture

   ▸ Volumes 1—5

GoF Classification Of Design Patterns

▶ **Purpose - what a pattern does**

  ▶ Creational Patterns

    ▶ Concern the process of object creation

    ▶ *Abstract Factory, Builder, Factory Method, Prototype, Singleton.*

  ▶ Structural Patterns

    ▶ Deal with the composition of classes and objects

    ▶ *Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy.*

  ▶ Behavioral Patterns

    ▶ Deal with the interaction of classes and objects

    ▶ *Chain of responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template, Visitor.*

The Sacred Elements of the Faith

the holy origins

the holy structures

the holy behaviors

107 **FM** Factory Method								**139** **A** Adapter
117 **PT** Prototype	**127** **S** Singleton					**223** **CR** Chain of Responsibility	**163** **CP** Composite	**175** **D** Decorator
87 **AF** Abstract Factory	**315** **TM** Template Method	**233** **CD** Command	**273** **MD** Mediator	**293** **O** Observer	**243** **IN** Interpreter	**207** **PX** Proxy	**185** **FA** Façade
97 **BU** Builder	**315** **SR** Strategy	**283** **MM** Memento	**305** **ST** State	**257** **IT** Iterator	**331** **V** Visitor	**195** **FL** Flyweight	**151** **BR** Bridge

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

GoF Classification Of Design Patterns (Continued)

▶ Scope - what the pattern applies to

  ▶ Class Patterns
    ▶ Focus on the relationships between classes and their subclasses
    ▶ Involve inheritance reuse

  ▶ Object Patterns
    ▶ Focus on the relationships between objects
    ▶ Involve composition reuse

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

GoF Essential Elements Of Design Patterns

▸ **Pattern 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?

GoF Essential Elements Of Design Patterns (Continued)

▶ Solution

  ▶ A 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

▶ Consequences

  ▶ The pros and cons of using the pattern

  ▶ Includes impacts on reusability, portability, extensibility

GoF Pattern Template

▶ **Pattern Name and Classification**

  ▶ A good , concise name for the pattern and the pattern's type

▶ **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

▶ **Applicability**

  ▶ Situations where the pattern can be used

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

GoF Pattern Template (Continued)

- Structure
  - A graphical representation of the pattern

- Participants
  - The classes and objects participating in the pattern

- Collaborations
  - How to do the participants interact to carry out their responsibilities?

- Consequences
  - What are the pros and cons of using the pattern?

- Implementation
  - Hints and techniques for implementing the pattern

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.
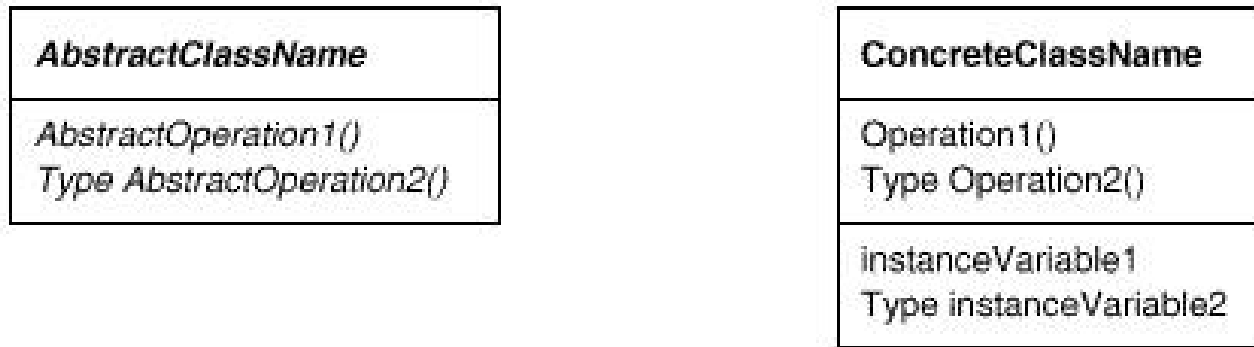
GoF Pattern Template (Continued)

▸ **Sample Code**

  ▸ Code fragments for a sample implementation

▸ **Known Uses**

  ▸ Examples of the pattern in real systems

▸ **Related Patterns**

  ▸ Other patterns that are closely related to the pattern

GoF Notation

▶ The GoF book uses the Object Modeling Technique (OMT) notation for class and object diagrams:

▶ OMT has proposed three main types of models:

  ▶ *Object model* : represents the static phenomena in the modeled domain. Main concepts are classes and associations, with attributes and operations, aggregations and generalizations (with multiple inheritance).

  ▶ *Dynamic model* : represents a state/transition view on the model. Main concepts are states, transitions, and events.

  ▶ *Functional model* : handles the process perspective of the model, corresponding roughly to data flow diagrams. Main concepts are process, data store, data flow, and actors.
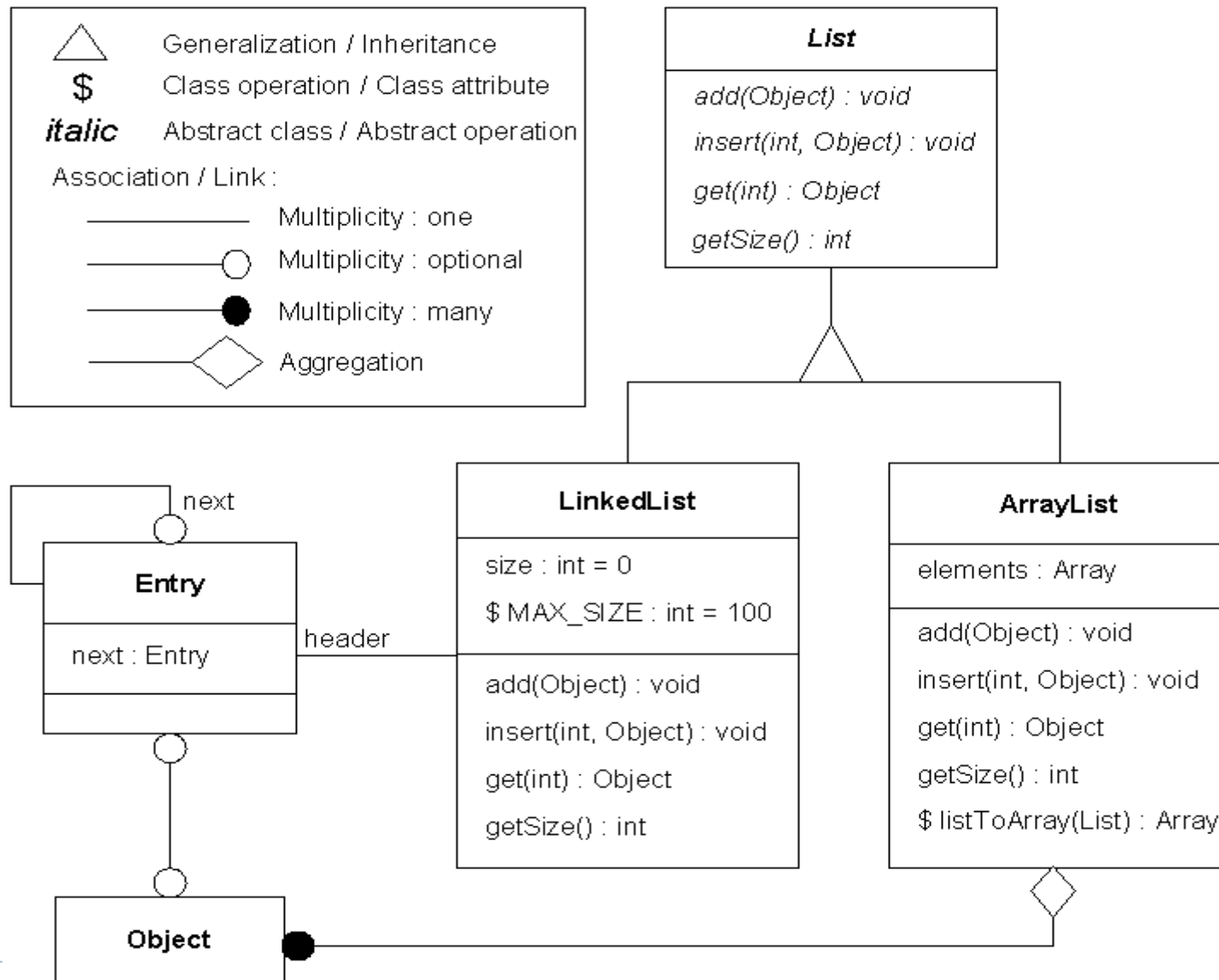
OMT object model

▸ Appendix B of the GoF book.

AbstractClassName
AbstractOperation1() Type AbstractOperation2()

ConcreteClassName
Operation1() Type Operation2()
instanceVariable1 Type instanceVariable2

(a) Abstract and concrete classes

Client

Client

(b) Participant Client class (left) and implicit Client class (right)

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

OMT object model (continued)

Legend:
- △ Generalization / Inheritance
- $ Class operation / Class attribute
- *italic* Abstract class / Abstract operation
- Association / Link :
  - ——— Multiplicity : one
  - ———○ Multiplicity : optional
  - ———● Multiplicity : many
  - ———◇ Aggregation

List
- *add(Object) : void*
- *insert(int, Object) : void*
- *get(int) : Object*
- *getSize() : int*

Entry
- next : Entry

(next)
(header)

LinkedList
- size : int = 0
- $ MAX_SIZE : int = 100
- add(Object) : void
- insert(int, Object) : void
- get(int) : Object
- getSize() : int

ArrayList
- elements : Array
- add(Object) : void
- insert(int, Object) : void
- get(int) : Object
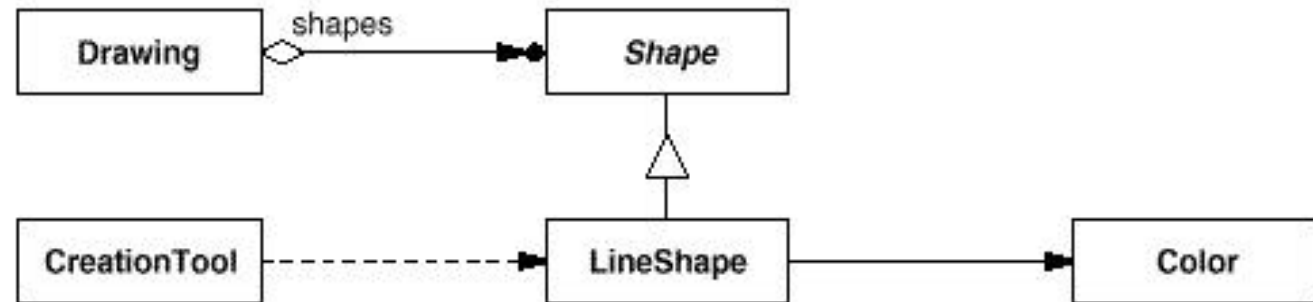- getSize() : int
- $ listToArray(List) : Array

Object

OMT object model (continued)

reference
(do not use
associations)
when
describing
DP
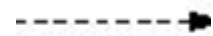
instantiation
relation

(c) Class relationships

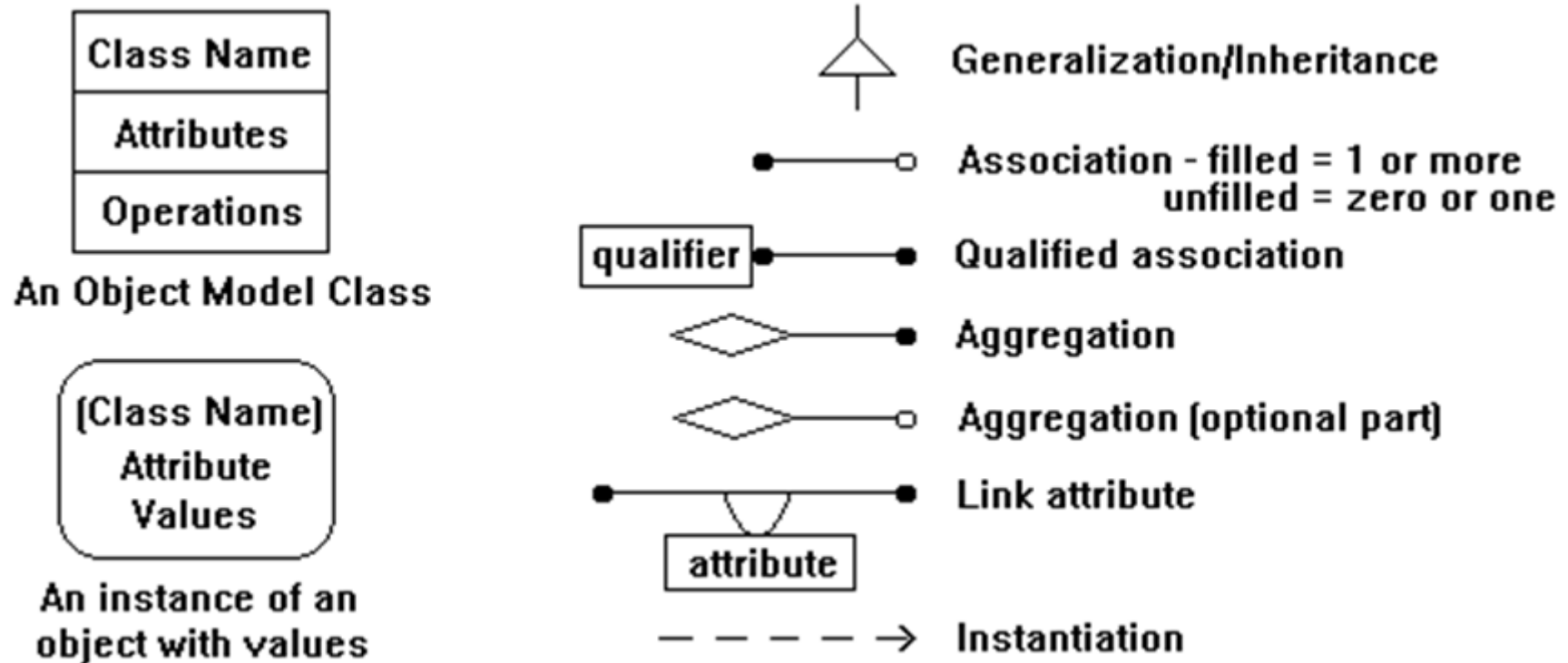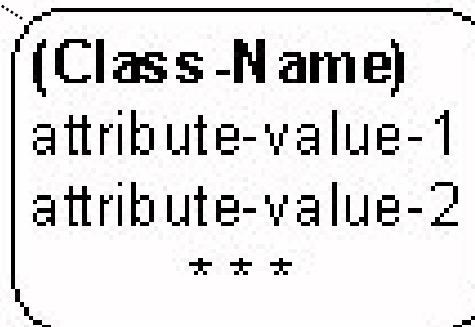**Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.**

OMT object model (continued)

anchor a note
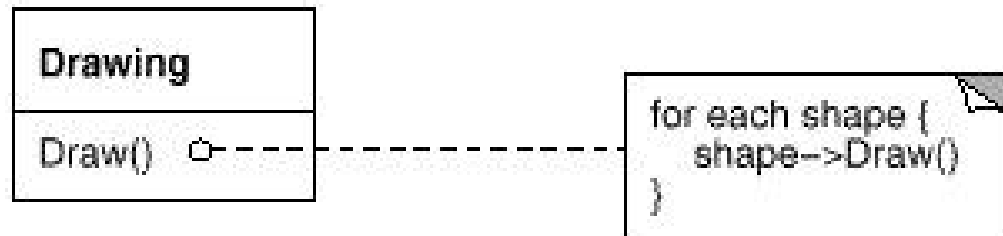
(d) Pseudocode annotation

OMT object model (continued)

An Object Model Class

An instance of an object with values

Generalization/Inheritance

Association - filled = 1 or more
unfilled = zero or one

Qualified association

Aggregation

Aggregation (optional part)

Link attribute

Instantiation

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Classes & instances (metadata & data)

Class-Name
attribute-name-1:data-type-1=default-value-1 attribute-name-2:data-type-2=default-value-2 * * *
operation-name-1(argument-list1):result-type-1 operation-name-2(argument-list2):result-type-2 * * *

(Class-Name)
attribute-value-1
attribute-value-2
* * *

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.
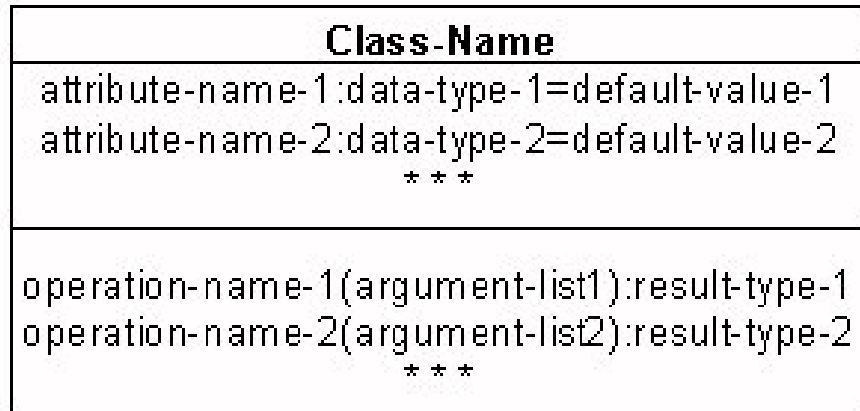
A citation

Good design and programming is not learned by generalities, but by seeing how significant programs can be made clean, easy to read, easy to maintain and modify, human-engineered, efficient, and reliable, by the application of good design and programming practices. Careful study and imitation of good designs and programs significantly improves development skills.
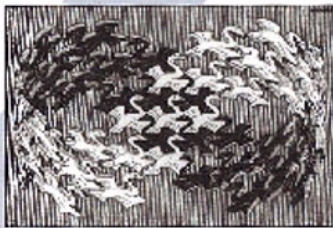
- Kernighan and Plauger

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Books

Design patterns, Laura Semini, Università di Pisa, Dipartimento di Informatica.

Hours of lesson and time available to students

▶ Any problem with lessons skedule?

▶ I'm available for talking with the students before the lectures.