Reuse at Design Level: Design Patterns

Reuse in Software Engineering

- Reuse at code level is common in software development e.g. C standard libraries such as math.h and stdio.h;

- User defined libraries such as "serverutils", "library.h"

- What about reusing old design solutions and not just the code ?

Christopher Alexander's Work

- Two books for building architects
  - The timeless way of building: Alexander 1977
  - A Pattern language: Alexander et al. 1977
- He classified the problems that occurred again and again and described core solutions to them that could be used again and again
- Examples: main entrance, sequence of sitting spaces, public outdoor room, interior windows

Patterns in Software Engineering

- Studies in other disciplines is helpful in software engineering other than computer science, its basic discipline

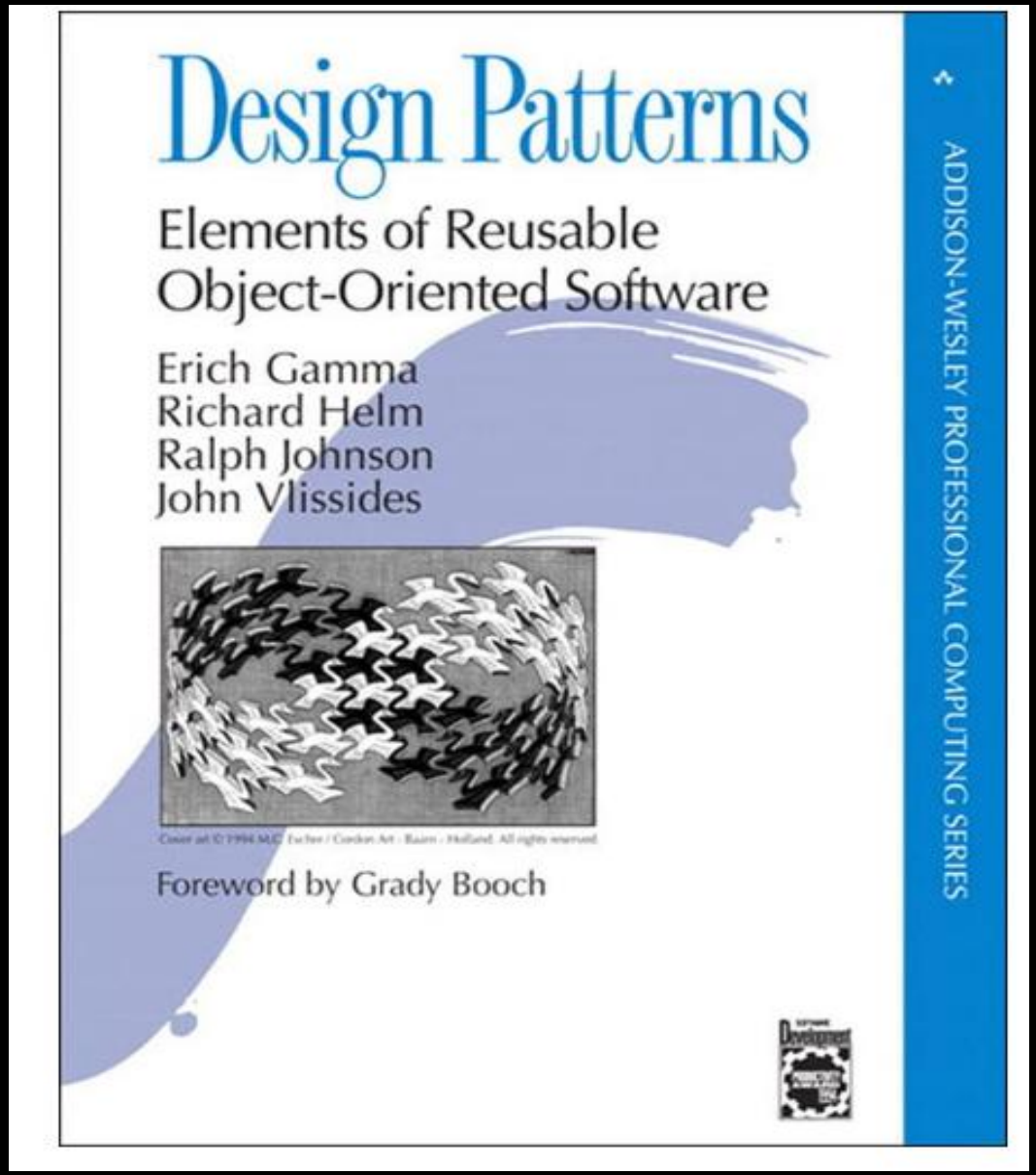
- Researchers in Object Oriented Software Engineering now find that design patterns can be formulated to represent commonly occurring problems in design and also the solutions to them

Framework Cookbooks

- Frameworks such as Smalltalk's MVC were available in 80's
- But using a framework for a specific application needed the knowledge of classes and class interactions in the framework
- e.g Krasner and Pope's cookbook on MVC framework (1988)
- e.g. Ralph Johnson's cookbook: HotDraw for implementing graphical editor (1992)

The Growth of Pattern Community

- Gamma described patterns in ET++ framework in his Ph.D. thesis in 1992
- Peter Coad published an article on design patterns in an issue of CACM in 1992
- Coad organized OOPSLA workshops on patterns in 1992 and 1993
- The pioneering book on design patterns by the gang of four
- Since then patterns have been discussed widely in the OO Software community

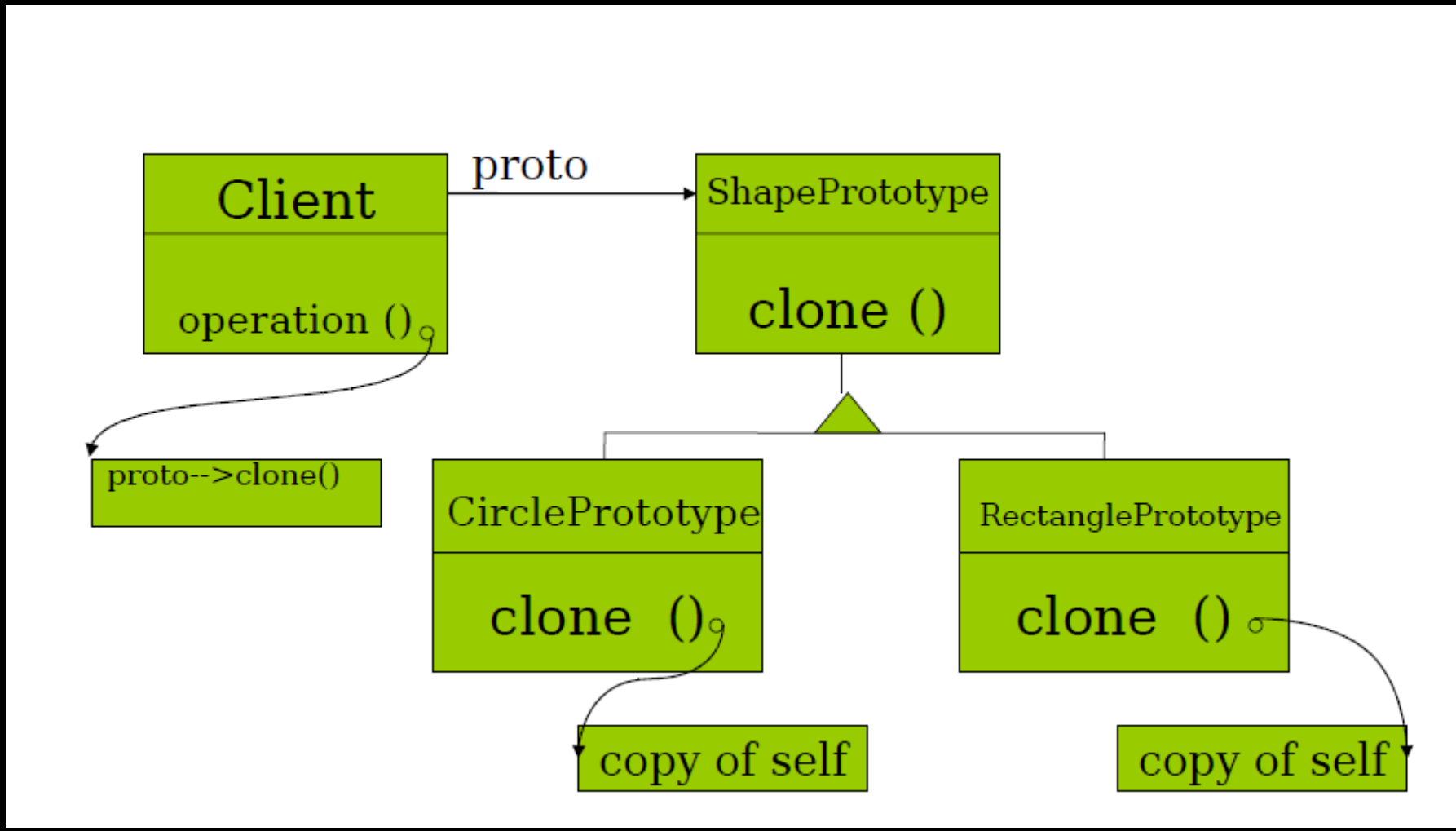

Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

A Problem

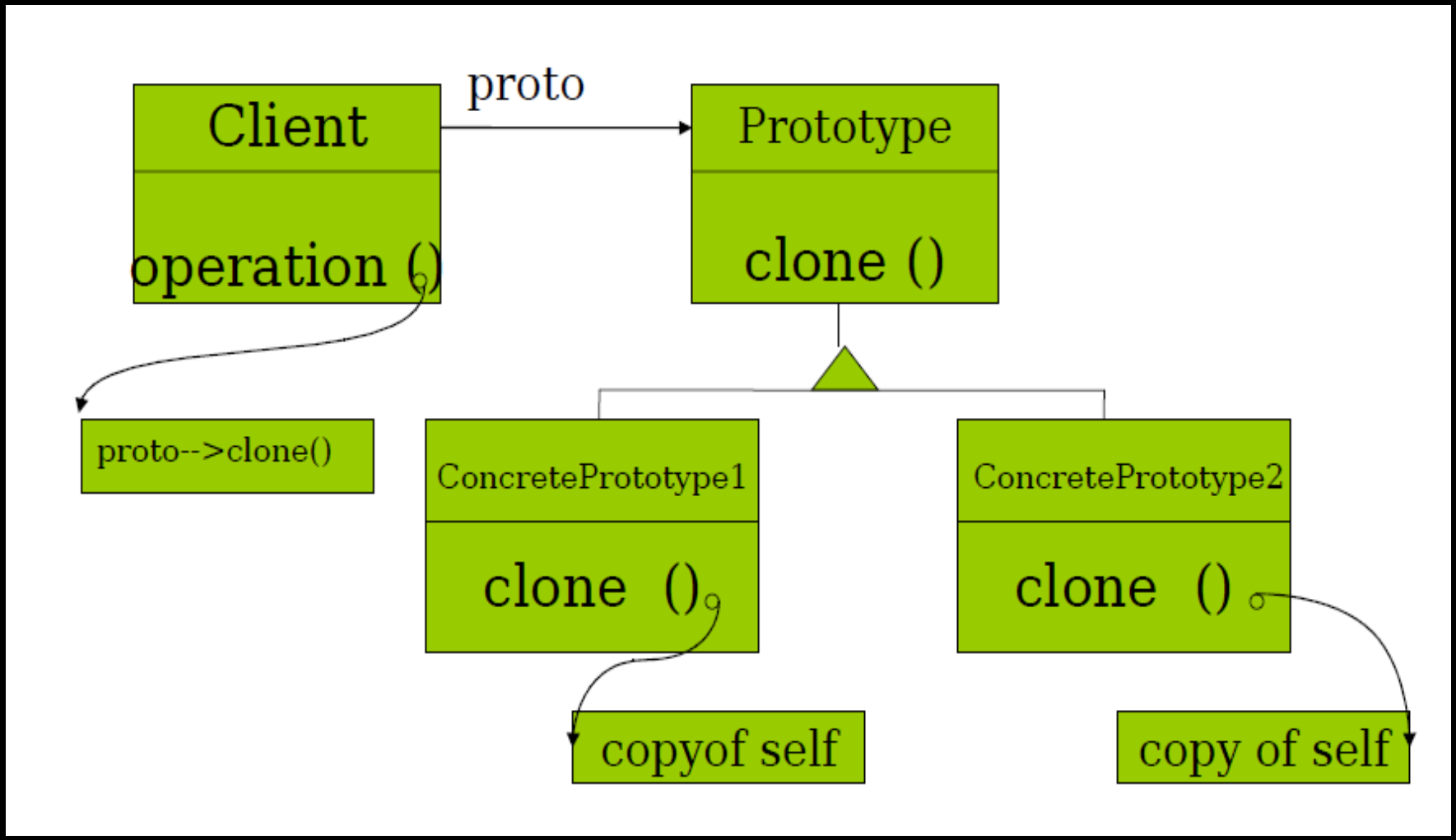
- In a graphical editor, by clicking on an object, one can obtain a copy of the original object. Obtaining a copy of an existing object is a common design problem in on-line compositions.

- We can provide a design solution to solve this problem, and reuse this design whenever similar situation arises.

The Solution



The Solution



The Solution

```
cloneHandler(Shape s) {
      Shape temp = s.copy();
     insert temp in work space
}
x copy() {
     temp = new x();
     temp.attribute1=
     temp.attribute2=
     return temp;
}
obj.copy();
```

Describing a Design Pattern

- Specify the generic problem that is solved
- Motivate the design pattern solution with the help of an example
- Provide the structure for the pattern
- Discuss collaborations between classes
- Discuss other issues related to the pattern such as trade-offs, implementation techniques etc.

Template provided by Erich Gamma et. al

- Pattern name, its classification
- Intent, Motivation, Applicability
- Structure, Participants, Collaborations
- Consequences, Implementation, Sample code,
- Known uses
- Related patterns

Classification of Patterns

- Creational Patterns
  - concerned about ways to create new objects
- Structural Patterns
  - concerned about the composition of objects and classes
- Behavioral Patterns
  - concerned about ways in which objects interact

Classification of Patterns

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	**Factory Method**	**Adaptor Class**	**Interpreter** **Template Method**
	Object	Abstract Factory Builder Prototype Singleton	Adaptor (Object) Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Creational Patterns

- Singleton
  - To create a sole instance of a class
- Prototype
  - To create objects by cloning existing objects
- Builder
  - Build an object from existing representation

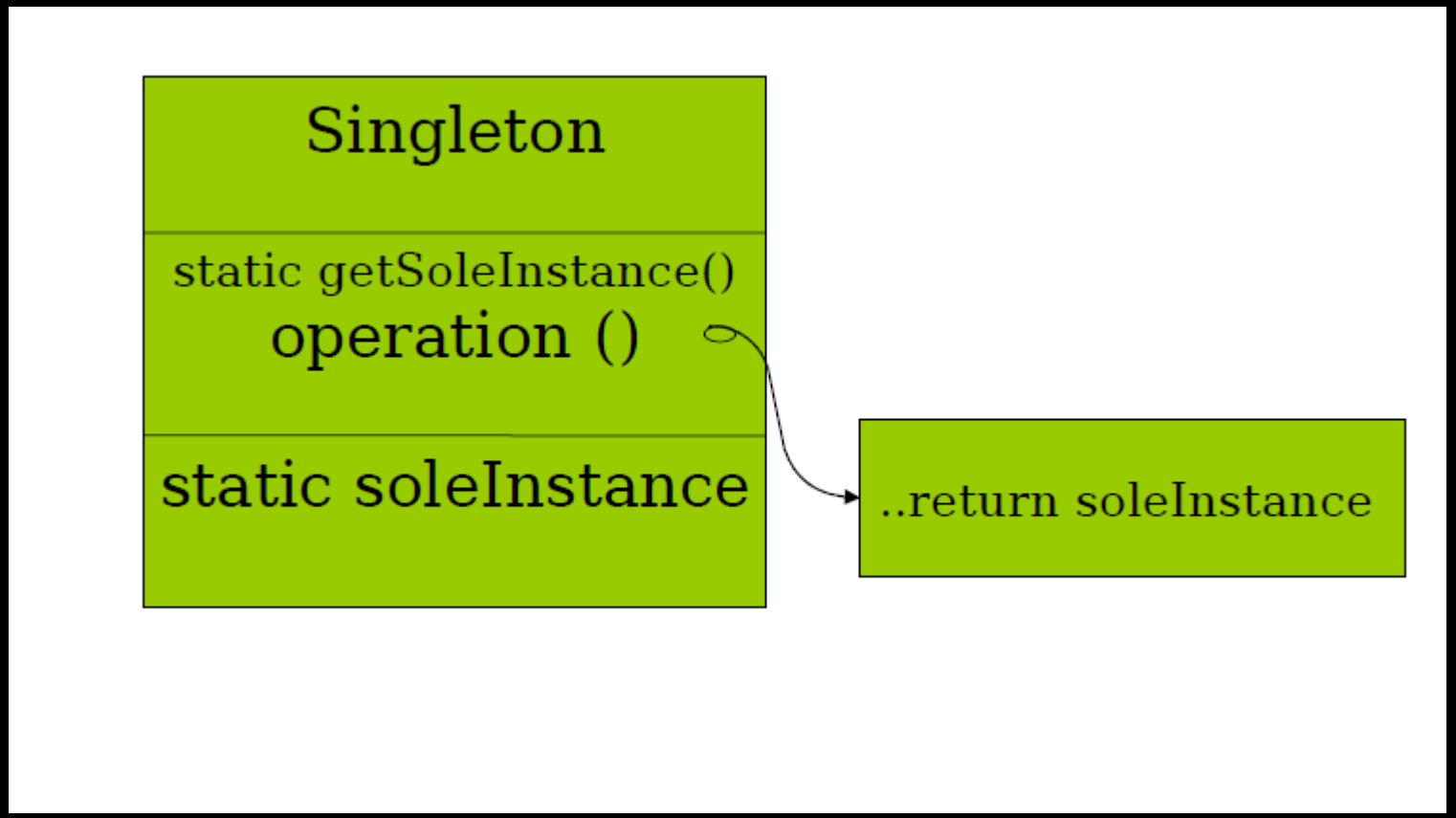
Creational Patterns

- Factory Method
  - Defer instantiation to subclasses

- Abstract Factory
  - Provides interface to create families of objects without specifying the concrete classes of the objects

Singleton

- A class that creates only one instance at most



Implementing Singleton

- Make the constructor private
  - Prohibit normal creation mode
- A new instance can only be created through a class method
- The class method is the static method in our case
- Return the unique instance crated

Singleton.c

```
Class Singleton {
        private Singleton () ;
        private static Singleton soleInstance;
        public static Singleton getSoleInstance () {….};
}
```