Using Patterns in the CS Curriculum

An NSF/SIGCSE-Sponsored Tutorial
at the
5th Northeastern Conference
of the
Consortium for Computing in Small Colleges

Eugene Wallingford
Department of Computer Science
University of Northern Iowa

April 29, 2000

Elementary Patterns Community

Owen Astrachan	(Duke University)
Joe Bergin	(Pace University)
Robert Duvall	(Duke University)
Ed Epp	(Intel Corporation)
Rick Mercer	(University of Arizona)

Alyce Brady	(Kalamazoo College)
Doug Dechow	(Oregon State Univ.)
Dwight Deugo	(Carleton University)
Javier Galvé-Frances	(University of Madrid)
Zung Nguyen	(Rice University)
Viera Proulx	(Northeastern Univ.)
Richard Rasala	(Northeastern Univ.)
Stephen Wong	(Oberlin College)

<your name here!>

Outline of the Presentation

What are patterns?

- history
- definitions
- examples

How can patterns help us teach CS?

The future of elementary patterns

Questions and answers

A Little Pre-History

The architect Christopher Alexander has been exploring the nature of form in the “built world” for over 40 years.

He adopted the term **pattern** to denote the structural relationships that recur in “good” houses and other “good” spaces.

We use “pattern” to mean three distinct but related ideas:

- a thing
- a description of the thing
- a description of how to build the thing

Light on Two Sides of Every Room

A thing ...

any configuration in a building in which a room receives exterior light from at least two sides

A description of the thing ...

a relationship among a room in a building, other interior spaces, and the exterior of the building

A description of how to build the thing ...

Place rooms in corners of the building. Juxtapose small rooms with large ones. Take care not to destroy the building's roof layout. Make the windows open on something beautiful. ...

The Idea of Software Patterns

At a 1987 OOPSLA workshop, Beck and Cunningham presented their initial effort at applying Alexander's ideas about patterns to software.

Concurrently, many other folks were working on the problem of how to codify software design expertise. They gradually came into contact with patterns, and the idea struck a chord with many of them. Thus was the **software patterns** community born.

The popular birthday of software patterns is probably the release of the book *Design Patterns* by the "Gang of Four".

The Composite Pattern

A thing ...

a panel in a Java program, or
the expression `(3 + (4 * 5))`

A description of the thing ...

an object in some program that behaves like other objects but which *contains* a collection of such objects and uses them to perform its tasks

A description of how to build the thing ...

Create a class that implements the desired interface. Give the class an instance variable that is a collection of objects of the interface type. Implement its methods by delegating responsibility to the objects in its collection. ...

What Patterns Are

A structural relationship between the components of a system that brings into equilibrium a set of demands on the system.

“A solution to a problem in a context”

In order to understand why the pattern recurs, one must understand the design trade-offs that underlie it.

A description of these trade-offs is an essential part of the pattern as description.

A Software Pattern (1)

Name

Mutual Recursion

Context

You are using **Structural Recursion**, over a BNF description of the data.

Problem

What should you do when the data element you are processing has a BNF description that refers to the same inductively-specified data type?

```
<list> ::= ()
        |  (<symbol-expression> . <list>)

<symbol-expression> ::= <symbol>
                     |  <list>
```

A Software Pattern (2)

Forces

You want your code to be easy to read and modify.

Creating a single procedure focuses the reader’s attention on one piece of code but may result in code with complex case analysis.

Furthermore, a single procedure obscures the fact that your data is defined in separate expressions. This makes adding or changing data definitions difficult.

Writing multiple procedures complicates the task of reading the code but makes the relationship between the data definitions explicit.

A Software Pattern (3)

Solution

Use **Structural Recursion** on *both* data definitions.

Have each procedure invoke the other at the corresponding point in the code.

Give the helper procedure a name that indicates the data type it operates on.

Resulting Context

Use a **Local Procedure** to eliminate name clashes or undesired clutter to your set of procedure definitions.

Use **Program Derivation** to reduce the cost of the extra procedure calls, especially when operating on deeply-nested lists.

Pattern Languages

The initial context and the resulting context of a pattern generally refer to other patterns.

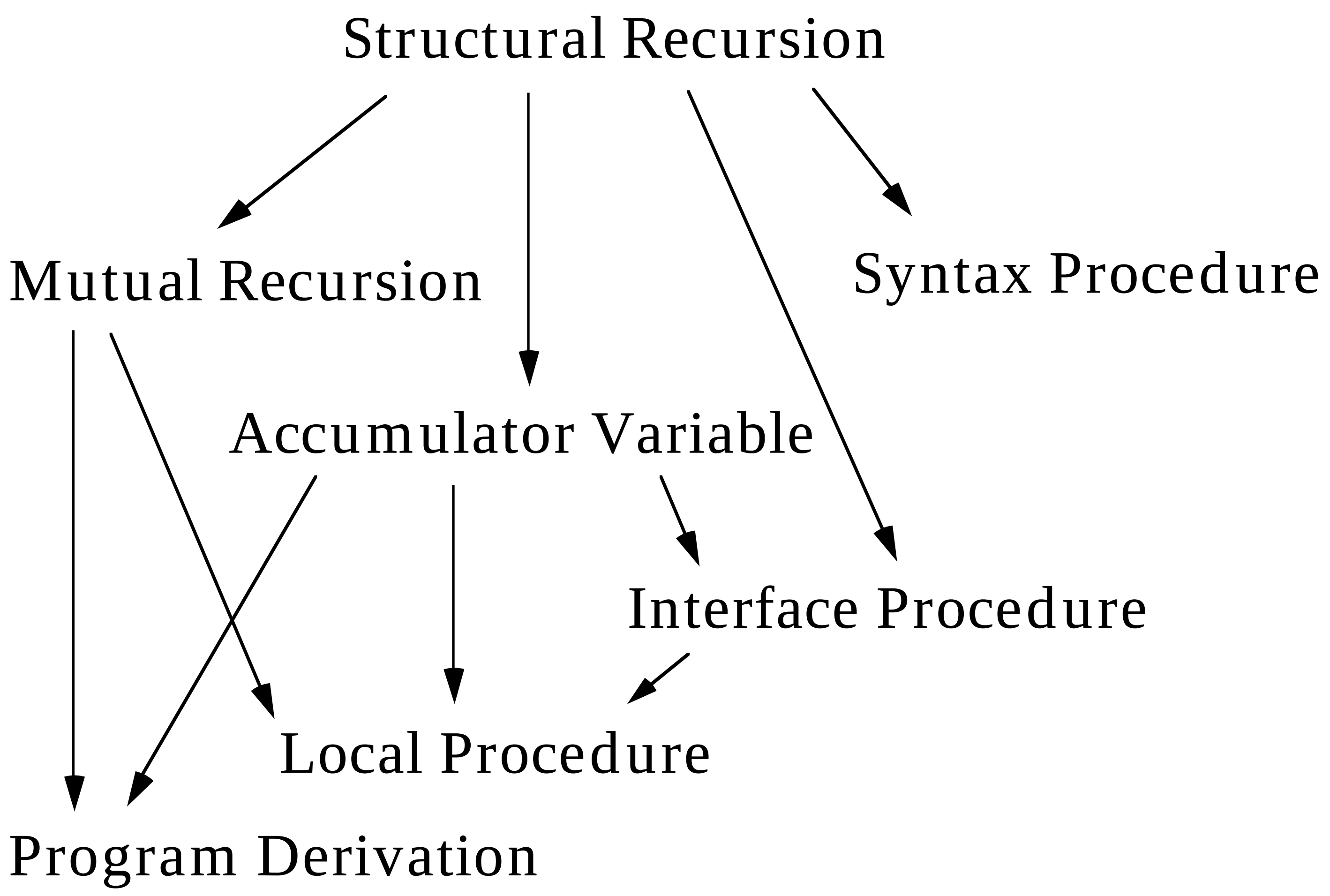
- Mutual Recursion occurs when one is doing Structural Recursion.
- When one does Mutual Recursion, one often also needs to implement a Local Procedure or a Program Derivation.

In complex systems, *patterns contain other patterns*, are built out of them.

A **pattern language** is a collection of patterns that, when implemented together, generate a complete structure.

Roundabout

The patterns in the previous example are part of Roundabout, a pattern language for doing recursive programming in a functional programming style.



What Patterns Are <u>Not</u>

- Patterns ≠ “Gang of Four”
- Patterns ≠ OOP

- Patterns ≠ extra material
- Patterns ≠ new material

- Patterns ≠ programming templates
- Patterns ≠ tricks

- Pattern language ≠ a pattern catalog

Elementary Patterns

Elementary patterns have a long history:

- in the mid-1980s: Soloway, Clancey, ...
- in the mid-1990s: my CS1 courses
- at the same time: others were exploring ways to increase their students' pro- gramming "literacy" through early design, apprenticeship models, ...

Then patterns hit the scene.

Patterns are consistent with all these ideas, and they add something more: a unifying framework for documenting program structure and the process that generates it.

So many of us began to: write patterns.
use patterns.

Outline of the Presentation

What are patterns?

How can patterns help us teach (CS)?

- what patterns offer teachers
- what patterns offer learners
- how to use patterns in a course

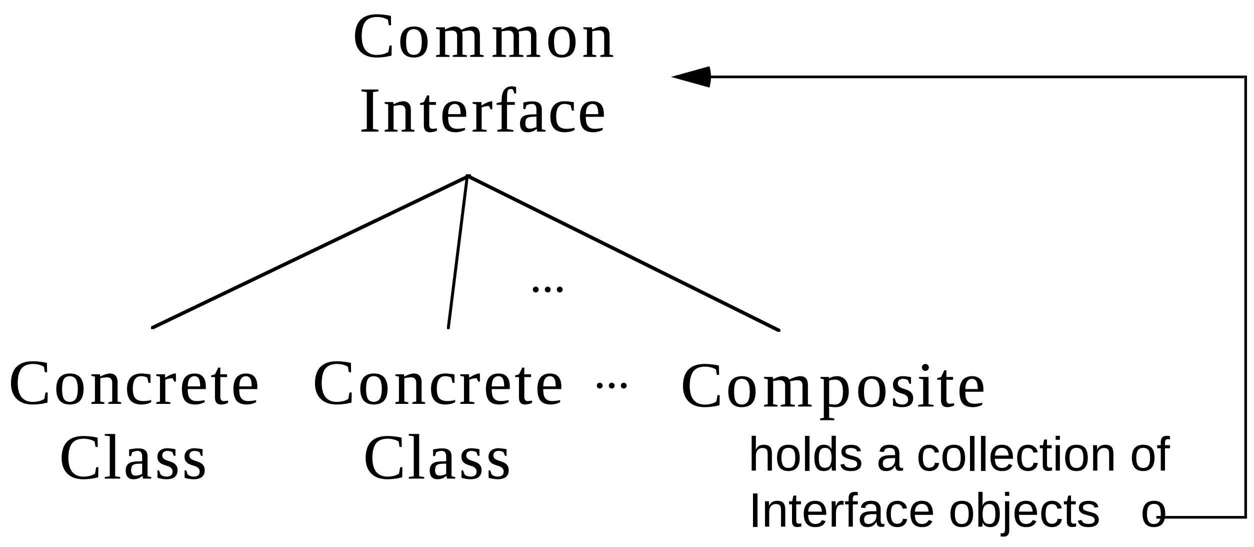
The future of elementary patterns

Questions and answers

Patterns are a laboratory for studying design principles.

I use simple OO patterns like **Decorator** and **Composite** to help my students learn the basic principles of OOP:

- composition and delegation
- interfaces and inheritance
- dynamic polymorphism



Patterns are tools for reading programs.

Patterns provide a vocabulary for talking about the basic structures students see in programs.

The Java class libraries abound with examples of common OO patterns:

- the `Enumeration` interface
- the `MouseAdapter` class
- the `java.io` package

Patterns are tools for writing programs.

The same patterns that serve as a vocabulary for reading programs can serve as a vocabulary for writing programs.

This was my initial motivation for using procedural programming patterns in CS1:

- selection patterns
  - guarded action
  - alternative action
- repetition patterns
  - process-all-items
  - counting and accumulation loops
  - linear search

Patterns are benchmarks for evaluating programs.

Grading programs is often too subjective.

A pattern language shows the context in which each pattern appears. It guides the student through the process of generating a program from the “top” down.

When my class studies Roundabout, we all share a common vocabulary and a common understanding of when certain structures should appear in a program.

Students can write any program they want, but the burden of proof is on them whenever they choose to deviate from the pattern language!

Writing patterns will help an instructor to understand the material even better.

Writing patterns is a skill that takes time and practice to develop. But the effort to write patterns pays off many-fold in a sharpened understanding of the kinds of programs that we want our students to be able to write.

- What **is** the problem being solved?
- What **is** the solution?
- **Why** is this solution better than the alternatives?
- **How** do these pieces grow together to make a good program?

Patterns help students appreciate the beauty of programs.

Patterns provide structure to a program in a way to balances the concerns facing the program (and the programmer!).

* An individual pattern like **Decorator** gives birth to beauty by combining two tools, composition and inheritance, in a way that maximizes each's benefits and minimizes each's costs.

* A pattern language like **Roundabout** gives birth to beauty by showing how small parts can work together to create a whole that is greater than the sum of the parts—a program that is better than we would have generated otherwise.

Using Patterns in the Classroom

~~Figure It Out~~
~~The Pattern Lecture~~

Pattern Mining

- Give the students an exercise to solve in which the pattern is likely to appear.
- Give the students another such exercise, with a somewhat different flavor.
- Examine candidate solutions to the two exercises.
- Identify the pattern, and give it a name.
- Give the students another exercise to do; tell them to use the pattern explicitly.
- Assign a programming assignment in which the pattern plays a major role.

Using Patterns in the Classroom

Guided Discovery

- Give the students an exercise to solve in which the pattern is key to a good solution.
- Examine successively more satisfactory (and less obvious?) solutions, exposing the forces.
- Identify the pattern, give it a name, and discuss other applications of it.

Before and After

- Show a system designed or implemented *ad hoc*, or at least without the aid of patterns.
- Show the same system after applying the patterns.
- Discuss the pros and cons of each version.

Outline of the Presentation

What are patterns?

How can patterns help us teach (CS)?

The future of elementary patterns

- resources
- getting involved

Questions and answers

Resources

`http://www.cs.uni.edu/`
`~wallingf/patterns/elementary/`

The Elementary Patterns home page provides links to:

- information on how to participate in the elementary patterns community
- elementary patterns and pattern languages
- ideas on how to use patterns in the classroom
- conferences and workshops
- other patterns resources of interest
- other resources of interest

The web site is a community effort...

How To Get Involved

Participate in a SIGCSE workshop.

Participate in a BoF at OOPSLA.

Write a pattern and submit it to PloP™.

Come to ChiliPloP™ to learn more about patterns or to work with us elementary patterns.

Join our mailing list. Don't just lurk; ask questions and pose problems and join in!

Surf the Elementary Patterns web page.

Outline of the Presentation

What are patterns?

How can patterns help us teach (CS)?

The future of elementary patterns

Questions and answers
