Design Patterns in Undergraduate Education

Presented by:

Feifei Li

Shad Gilley
Overview

• What is a design pattern?
• What are the advantages and disadvantages of teaching DPs in undergraduate classes?
• How are DPs currently being taught?
• Is it helpful to teach DPs in undergraduate classes?
• What do you think?
What is a design pattern?

• From our textbook:
  – A design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems…

• Emphasis on object-oriented solutions
• Focuses on specifying the problem and the solution through particular examples
• Includes pattern tradeoffs
What is a design pattern?

• Example pattern: Flyweight
• See book p. 198
• Notice the emphasis on object-oriented design
• Even the code example is a class!
• Notice the consequences section
What is a design pattern?

- From Joseph Bergin, CS professor at Pace University
  - A "pattern" is a solution to a problem in a context
  - a pattern is an attempt to establish "best practice" with respect to a problem or class of problems
- Focuses on communicating solutions from a master to a novice
What is a design pattern?

- Example pattern: One Liner
- Problem: code layout
- Context:

You are writing a selection structure and notice that all of the parts are short. Since your reader expects that what is on one line all goes together, you should put the entire structure on one line if it fits comfortably on one line

```c
if ( grade > 60 ) { output("passing"); } else { output("failing"); }
```
What is a design pattern?

• From Eugene Wallingford, CS professor at University of Northern Iowa
  – elementary patterns are patterns appropriate for novices first learning to design and write programs.
• Focuses on patterns that can be readily grasped by novices
• Should not require a lot of background knowledge
What is a design pattern?

• Example pattern: Linear Search

How do you find an object that meets a specific condition?

Suppose that you have a set of students, and you would like to find the first student with an "A" average. In the worst case, you will look at the whole collection before finding your target. But, if you find a match sooner, you would like to terminate the search and work with the object that you found.

Therefore, construct a Process All Items loop, but provide a way to exit the loop early should you find your target sooner.

```java
for (i = 0; i < students.size(); i++)
    if (student[i].grade().isAnA())
        break;
```
What is a design pattern?

• Pedagogical patterns
  – reusable pedagogical design patterns

• Specify ways of teaching design patterns that are repeatable and easy to adapt to different situations

• Collection of teaching patterns:
  http://www-lifia.info.unlp.edu.ar/ppp/
What is a design pattern?

- Example pattern: Mistake

Students are asked to create an artifact such as a program or design that contains a specific error. Use of this pattern explicitly teaches students how to recognize and fix errors. We ask the student to explicitly make certain errors and then examine the consequences.
What is a design pattern?

• Summary:
  – No general agreement on the exact definition of a design pattern.
  – The idea of patterns is being applied to many different domains, not just OO design
Advantages/Disadvantages

• Advantages
  – helps students to build a foundation for continued study of OO programming
  – DPs provide a good example of design for students
  – patterns can help student organize examples and code they see
  – help students learn good programming techniques
Advantages/Disadvantages

- **Disadvantages**
  - Expert patterns may be inaccessible to novices
    - students and educators may not be familiar with the necessary OO concepts
  - Study by Shankelford and Badre
    - compared the use of several patterns for devising loops in Pascal
    - presented two sets of patterns for constructing loops
    - student were better able to understand one set of patterns
Advantages/Disadvantages

Descriptive rules

- If the number of iterations of the loop is known in advance, use a for loop.
- If the loop might not require any iterations, use a while loop.
- If the loop will require at least one iteration, use a repeat loop.

Constructive rules

- If the value of the control variable is simply a count of the number of iterations, use a for loop.
- If the value of the control variable exists apart from the loop (you need only access it), use a while loop.
- If the value of the control variable exists only after computation within the loop, use a repeat loop.
Advantages/Disadvantages

• Disadvantages
  – Just giving examples of patterns is not enough for students to understand
  – Study by Mann
    • two introductory Pascal classes
    • one used a patterns based approach while the other used a traditional approach
    • the patterns class was not able to understand and apply the patterns discussed
Advantages/Disadvantages

• Patterns based approach
  – Taught with each code pattern were rules for when to use it, the function it performed, and the parts that comprised it. Solution development focused on finding patterns that would solve a part of the problem and tailoring or combining them appropriately. Reverse mapping from code to templates was the main mechanism for communicating the performance of a program or code segment.
How are DP being taught?

- Where can design patterns be introduced in undergraduate studies?
  - CS1
  - CS2
  - Object Oriented Programming
  - Software Engineering
How are DP being taught?

• CS1
  – Introduction to programming
  – Integration of design patterns
    • Elementary patterns
    • Pattern languages
    • Stylistic patterns
  – Examples
How are DP being taught?

• Elementary patterns
  – elementary patterns are patterns appropriate for novices first learning to design and write programs

• Pattern languages
  – set of related patterns that work together in some context
  – different contexts produce different languages
How are DP being taught?

Example elementary pattern language

- Whether or Not
- Alternative Action
- Return Not Else
- Conditional Expression
- Range of Possibility
- Sequential Choice
- Unrelated Choice
- Independent Choice
- Partial Dependence
How are DP being taught?

• Stylistic patterns
  – patterns for improving the readability of programs
• Example
  – Brace All

if (measuredHeat() > BoilThreshold) { shutDownGenerator(); }
How are DP being taught?

• CS2
  – Data Structure/Algorithms
  – Integration of design patterns
    • Patterns from the Go4 book
    • Most of the papers on this subject simply present examples of how to use these patterns
    • No studies done to see if this is effective!
  – Example: State and Singleton pattern
How are DP being taught?

Singleton pattern to represent empty state

```java
Class EmptyListNode extends AListNode{
    private static AListNode _unique=null;
    static AListNode singleton() {
        if ( null == _unique)
            _unique= new EmptyListNode ();
        return _unique;
    }

    Public List () {
        _link = EmptyListNode.singleton();
    }
}```
How are DP being taught?

- Object Oriented Programming
  - We could not find any papers describing the use of design patterns in OO
  - This was surprising since this seems like a logical place to introduce this topic
  - Class survey: How many learned about DP in OO course?
  - Perhaps this is too much to learn when shifting to the OO paradigm
How are DP being taught?

- Software Engineering
  - No specific papers, but we found a several courses that integrate DP with software engineering
  - Each course required CS1, CS2, and OO as prerequisites
  - Does this imply these courses are necessary knowledge for understanding DPs?
Helpful or not?

• Very little empirical evidence to answer this question
  – Wallingford paper: “I have done no formal analysis of the affects of the approach…”
  – Kendall paper: “the DP curriculum described here has not been formally evaluated.”
  – Quote from Preiss paper
Helpful or not?

• Only one study that actually measured the effect of using design patterns
  – Semester-long study in two introductory Pascal programming classes
  – Class using patterns did not perform as well as class using traditional approach

• Should we be concerned with verifying our assumptions about teaching DPs?
Helpful or not?

- Most of the papers discuss personal opinions on how design patterns can be used.
- Many authors seem to imply that just using patterns will be a panacea.
- The “Before and After” model
  - analyze anonymous student solutions
Helpful or not?

• From Bergin
  – First time: only mention general design principle
  – Next time: recall first use and note similarities
  – Third or fourth time: explicitly describe the pattern

• Use a single pattern as the theme for the course (example of scaling)
Helpful or not?

- Many papers give examples of patterns and how they can be used

<table>
<thead>
<tr>
<th>Design Pattern</th>
<th>CS2 Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapters</td>
<td>Stacks and queues</td>
</tr>
<tr>
<td>Template methods</td>
<td>Tree and graph traversals</td>
</tr>
<tr>
<td>Comparators</td>
<td>Priority queues</td>
</tr>
<tr>
<td>Decorators</td>
<td>Balanced trees, graphs</td>
</tr>
<tr>
<td>Iterators</td>
<td>Sequences, trees, graphs</td>
</tr>
<tr>
<td>Positions</td>
<td>Sequences, trees, graphs</td>
</tr>
<tr>
<td>Locators</td>
<td>Priority queues, dictionaries</td>
</tr>
</tbody>
</table>
Helpful or not?

• No conclusive evidence for the following:
  – When should design patterns be introduced?
  – Does teaching design patterns in introductory classes help students develop better programming skills?
  – Do elementary patterns provide a good foundation for more advanced patterns?
What do you think?