Software System Engineering

Dr. M.E. Fayad, Professor
Computer Engineering Department, Room #283I
College of Engineering
San José State University
One Washington Square
San José, CA 95192-0180
http://www.engr.sjsu.edu/~fayad
Lesson 6-1: Object-Oriented Design Heuristics
Lesson Objectives

- Overview of Previous Lecture
- Understand what do we mean by heuristics
- Understand what is an object-oriented design heuristic?
- Discuss the following:
  - Macho Class Problem
  - Interesting Design Problems
  - Topology Which Needs Accessor Methods
  - The Common Traps of Controller Classes
CRC Cards

- **General**
  - Each class is described on a separate 3X5 or 4X6 card
- **The cards are known as CRC card. They have 3 sections:**
  - Class
  - Responsibilities
  - Collaborations

<table>
<thead>
<tr>
<th>ATM (role)</th>
<th>Responsibility</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Access &amp; modify account balance</td>
<td>Clients</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Account</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Responsibility</td>
<td>Collaboration</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>1. Unique</td>
<td>Clients</td>
<td></td>
</tr>
<tr>
<td>2. One Responsibility</td>
<td>The clients of the named class (X)</td>
<td></td>
</tr>
<tr>
<td>3. Within Context</td>
<td>(5)</td>
<td></td>
</tr>
<tr>
<td>4. Indicate the role of the class</td>
<td>Server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 4-5 interfaces</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Services provided by the named class (X)</td>
<td></td>
</tr>
</tbody>
</table>

What are the good, bad, and ugly of the CRC cards?

Other Names: Interfaces, Commands, Requests, Functions, Methods, Procedures, Processes
Guidelines

- Pre- & Post-conditions
- Statements that guide you to do something.
- It is not a process.
- It is not a standards.
- It is not a methodology
- It is not a heuristic.
- Direction or information about something.
Object-Oriented Design Heuristics

- An OO design heuristic is a “rule-of-thumb”, not a rule.
- An OO design heuristic is something which makes a design “feel right” to its designer.
- An OO design heuristic is used to guide a designer in selecting the appropriate design choice from many possibilities.
- An OO design heuristic warns the designer of some impending doom when it is violated.
Object-Oriented Design Heuristics

- While some design heuristic certainly imply a sense of priority, there cannot be a prioritized ordering of the entire list (in general).

- It is not possible to follow all heuristics in a design. They frequently have colliding interests such as extensibility versus complexity.

- A design heuristic will tell users of design patterns when a particular pattern needs to be applied.
Design Patterns: Properties & Definitions

- [Webster] A pattern is a fully realized form, original, or model accepted, or proposed for imitation.
- [Webster] A pattern is something regarded as normative example to be copied; archetype; exemplar.
- [Alexander 79] A pattern is a solution to a problem in a context.
- [Alexander 79] A pattern has three parts:
  - Problem(s)
  - Context
  - Solution
- A pattern offers a workable solution.
- Patterns are rules of thumbs that can be used again and again -- useful, practical “how-to” guideline.
Design Patterns: More Properties & Definitions

- [Gamma 95.] Design patterns identify, name, and describe common and recurring designs appearing frequently in object-oriented systems.
- [Gamma 95] You can think of a design pattern as a micro architecture that contributes to overall system architecture.
- Each design pattern tends to be relatively small in size and scope.
- [Coplien 92] Patterns are a way of describing, documenting, and creating system architectures for software.
- Patterns tend not to be domain specific.
- Patterns are one of the primary mechanisms that people use for passing on expertise to others.
Object-Oriented Design Heuristics

- Heuristic Properties:
  - A heuristic is a “rule-of-thumb but not a rule.
  - Heuristics attempt to capture the unknown “feels right” feature of good analysis, good design, good documentation, etc.
  - Heuristics are strongly related to patterns in that they provide the rational for improving a design from a worse pattern to a better one.
  - All of the interesting heuristics are qualitative in nature.
  - There are no useful quantitative heuristics.
Object-Oriented Design Heuristics

- The absence of quantitative heuristics leads us to conclude that OO design is not precise.
- Some amount of fuzziness is required and only qualitative heuristics can accommodate this constraints.
- For an example of this fuzziness consider the following two important design heuristics that play off each other in an attempt to model the top level view of a system
  - A designer should distribute system intelligence uniformly among the top level classes in the system
  - A designer should minimize the number of collaborations on the top level of the system.
Object-Oriented Design Heuristics

- It is typical for designers to want the seemingly more useful quantitative metrics such as
  - A designer should have 4.6 top level classes per 1,000 lines of code.
  - A designer should have 35 to 45 percent of the relationships in a design be uses relationships.

- These heuristic are not useful and very misleading.
What is an Object?

- An object will always have four facets:
  1. Identity (this might be only its address in memory).
  2. Attributes of its class (usually static) & values of those attributes (usually dynamic).
  3. Behavior of its class (the implementor’s view)
  4. Public interface of its class (user’s view)
Classes & Objects: The Building Blocks of OO Paradigm

- Heuristic 2.1: All data should be hidden within its class.
- Heuristic 2.2: Users of a class must be dependent on its public interface, but a class should not be dependent on its users.
- Heuristic 2.3: Minimize the number of messages in the protocol of a class
- Heuristic 2.4: Implement a minimal public interface.
- Heuristic 2.5: Don’t put implementation details such as common-code private functions into the public interface of a class.
Classes & Objects: The Building Blocks of OO Paradigm

- Heuristic 2.6: Don’t clutter the public interface of a class with items that users of that class are not able to use or are not interested in using.

- Heuristic 2.7: Classes should only exhibit nil or export coupling with other classes, that is, a class should only use operations in the public interface of another class or have nothing to do with that class.

- Heuristic 2.8: A class should capture one and only one key abstraction.
Classes & Objects: The Building Blocks of OO Paradigm

- Heuristic 2.9: Keep related data and behavior in one place “localization”
- Heuristic 2.10: Spin off non-related information into another class (i.e., non-communicating behavior)
- Heuristic 2.11: Be sure the abstractions that you model are classes and not simply the roles objects play.
Discussion Questions

• T/F statements
  1. Objects should be intelligent agents
  2. Type names an interface.
  3. A valuable object works and plays well with others
  4. Design patterns identify, name, and describe common and recurring designs appearing frequently in object-oriented systems.
  5. Design patterns are domain independent.

• Define: legacy systems, method, message, interface
Questions for the Next Lecture

Define:

- Macho classes
- Accessor Methods
- Controller Classes
- Classes & objects in UML