Lifecycle Architecture Review: Preliminary Feedback

Lecture 10: Core Principles and Best Practices for Software Design (Part II)

"Treat design as a wicked, sloppy, heuristic process. Don't settle for the first design that occurs to you. Collaborate. Strive for simplicity. Prototype when you need to. Iterate, iterate, and iterate again. You'll be happy with your designs."
-- Steve McConnell, *Code Complete* (*2nd ed.*), Ch. 5

Outline

- Best practices for software design
- Time-tested software design principles
  - With examples

Resources

- *"Code Complete"*, 2nd ed., by Steve McConnell
  - Ch. 5: http://www.cc2e.com/docs/Chapter5-Design.pdf
- "*The Pragmatic Programmer*", by Andrew Hunt and David Thomas
  - Ch. 2 (section 7), Ch. 5 (section 26)
- *"Agile Software Development – Principles, Patterns and Practices"*, by Robert C. Martin
  - See handout
- *"Design Patterns Explained"*, by Alan Shalloway and James Trott
- *"On the Criteria to be Used in Decomposing Systems into Modules"*, by David Parnas

Best Practices for Software Design (discussed previously)

- Create at least three independent designs and choose the best one among them.
- Keep it simple (a.k.a. KISS principle).
- Ask yourself how you may test your components.
- Do not invest too much into visualizing early designs – they will change substantially.
- Learn to use design patterns.
- Consider if there are single points of failure or bottlenecks in your designs.
- Use abstractions as much as possible.
- Encapsulate changing components; fix the interfaces between them.
- Favor composition over inheritance.

Principles for Good Design: Loose/Weak Coupling

- Avoid unnecessary dependencies between modules.
- Law of Demeter: "Any method of an object should call only methods belonging to itself, to any parameters that were passed in to the method, to any objects it created, or to any directly held component objects."
  - Example: What is wrong with the following code?

```
public void showBalance (BankAccount acct) {
    Money amt = acct.getBalance();
    printToScreen (amt.printFormat());
}
```

Principles for Good Design: Single Responsibility Principle

- Also, principle of strong cohesion
- "A class should have only one reason to change."
  - "God object" metaphor
- Example 1: Not storing state in a GUI class.
  - Model-View-Controller (MVC) pattern helps to avoid this.
- Example 2: How is the principle violated below?

```
interface Modem {
   public void dial (String pno);
   public void hangup();
   public void send (char c);
   public char recv();
}
```

Principles for Good Design: Open-Closed Principle

- "Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification."
- Example: Extending an abstract class (with as many new subclasses as needed) rather than modifying an existing class to accommodate each new addition.
- The designer chooses what changes to anticipate and what parts of the system to "fix" (and assume that they won't change).