Lecture 16: Design Patterns, Cont

Viewing problems and solutions in context

Design Patterns Defined

“Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”


Pattern Principles

+ Encapsulate variance
+ Abstract the invariant
+ Favor composition

Strategy

+ Intent: Encapsulate each of a family of behaviors such that the use of the behaviors varies independently of the client.
+ Motivation: Remember the animals…
Strategy: Structure

A New Pattern
+ **Intent**: Attach additional responsibilities to an object dynamically or in response to changing requirements.
+ **Motivation**: What if we need to add additional processing steps to one of our algorithms…

A New Pattern, Continued

Decorator: Structure

Command
+ **Intent**: Encapsulate a request or action as an object.
+ **Motivation**: Sometimes we need to issue commands without knowledge of the specifics of any command – a menu is a good example.

Command: Structure
**Bridge and Adapter**

+ **Intent:** Decouple a set of implementations from the set of clients using them.
+ **Motivation:** Perhaps we want to enable different toolkits to perform the same operations...

**Resources**

+ Hillside.net ([http://hillside.net/patterns/](http://hillside.net/patterns/))
+ Patterns and Frameworks ([http://www.cs.wustl.edu/~schmidt/patterns.html](http://www.cs.wustl.edu/~schmidt/patterns.html))
+ Design Patterns: Elements of Reusable Object-Oriented Software. Gamma et al. Addison-Wesley, Boston, 1995.

**Lecture 19: Software Quality**

**Outline**

Today:

- Quality – a look back at history
- “Good enough” quality
- What is (software) quality?

Next time:

- How do we measure quality?
- How do we improve software quality?

**References**

- *Good Enough Quality: Beyond the Buzzword*, by James Bach
- *Professional Software Development*, by Steve McConnell
- *Agile Software Development with Scrum*, by Ken Schwaber and Mike Beedle
Food for Thought: Quality in Different Contexts

- The software ‘Gold Rush’ fever periods
  - Goal: being first-to-market in an unclaimed segment
  - Typical environment: two guys in a garage
  - High-risk projects, potentially high pay-off
  - Code-and-fix development, very informal processes
  - Customers are tech savvy, willing to forgive bugs
- The in-between (post-‘Gold Rush’) periods
  - Goal: sustained, productive competition with others
  - Typical environment: larger teams, formal processes
  - Lower-risk, likely lower but more predictable pay-off
  - Careful, quality-driven development with an emphasis on quality (reliability, interoperability, usability, etc.)
  - Different customer base: demands reliability

Back to Basics: The Goal of Building Software

- To deliver a product that satisfies the customer(s)
  - on time
  - on budget
  - with good quality
- But wait...

"Good Enough" Quality

- At some point, one needs to stop and decide it is all good enough to ship. Under what conditions?

Criteria for “good enough” quality:
1. There are clear benefits (of the software).
2. There are no critical problems.
3. Overall, the benefits outweigh the problems.
4. In the present situation and all things considered, further development would be more harmful than helpful.

Question: Is your product good enough now?

"Good Enough" Quality (cont.)

Important questions to consider:
- Good enough for whom?
  - You? Your team? The customer?
- Good enough for what?
  - A demo? A beta release? Selling it? Capturing market share?
- Have you agreed on ...
  - team standards for acceptable quality?
  - what would constitute success for your team in the end?
    - These are some of the team conversations we did earlier.

Take a Step Back: What is Quality?

- Quality is in the eyes of the beholder (customer).
  - First, define what quality means (for the customer!).
  - You and the customer must agree on the expected level of quality.
- What constitutes good quality in one situation may not be considered good quality in another.
  - E.g.: in a toy project vs. in a safety-critical system
- A contract must have at least the following components:
  - Who promises to do
  - What for Whom
  - by When
  - with what Quality Criteria/Standards, and
  - with what Notification Mechanism upon completion
Components of Quality

Quality comprises (but is not limited to):
- Requirements quality
- Design quality
- Code quality
- Test quality
- Documentation quality

Given limited resources, which of these do you consider more important to pay attention to? Why?