What is Software Design?

Gail Murphy

Slides available at:
www.cs.ubc.ca/~murphy/courses.htm

Objectives

- After this lecture you will be able to:
  - Describe the context (goals and constraints) of the activity of software design
  - Describe the kinds of information we must capture in a software design
  - Describe the kinds of effects that software design choices have on the final system
  - Use Observer design pattern even better than before

What is Clothing Design?

- Why might a designer decide to design such a jacket?
- What might have influenced the designer?

Photo from www.mec.ca

What is Building Design?

- Crystal Palace built for the Great Exhibition of 1851
- Inputs:
- Constraints:

Photo from http://65.107.211.206/1851/1851os.html
What is Bridge Design?

- Brooklyn Bridge
- Inputs:
- Constraints:

Photo from www.endex.com/gi/buildings/bbridge/bgallery/bbgallery3.htm

What is Design?

- What is design? What makes something a design problem? It’s where you stand with a foot in two worlds—the world of technology and the world of people and human purposes—and you try to bring the two together.
  

Kapor goes on to say...

- Design disciplines are concerned with making artifacts for human use. Architects work in the medium of buildings, graphic designers work in paper and other print media, industrial designers on mass-produced manufactured goods, and software designers on software. The software designer should be the person with overall responsibility for the conception and realization of the program.

Kapor’s Vision of Software Design

- Software design is not user interface design
- Software designer is concerned with overall product conception (e.g., Bricklin’s VisiCalc)
- Software designers should have strong technical grounding
- Software designer works in conjunction with developers
Software Design in 310

- In 310, you will largely be focusing on the technological (developer/engineer) view of software design
- How do we realize the conceived product?
- Inputs include requirements (functional and non-functional), developer’s experience
- Constraints include development organization, technical platform

An (Object-oriented) Design Problem

- An integrated development environment (like Eclipse)
- As I edit a Java source file (e.g., define a new attribute), a number of changes occur:
  - the text I add is highlighted appropriately
  - the Outline view changes
  - the Package Explorer view changes
  - A lightbulb might appear
  - And so on...

A Solution to the Design Problem

- Applying principles such as decomposition, abstraction, and information hiding, I may have iteratively arrived at the following classes

A Solution to the Design Problem...

- What do I need to make these classes interact to do the appropriate updating such that:
  - Performance is “good” (usually this would be quantified in some way)
  - New views can be added “easily”
Use Known Good Design Practices

• The Observer design pattern applies here:
  – Recall, the Observer pattern models a one-to-
    many dependency between objects so that when
    one object (the subject) changes state, all its
    dependents (the observers) are notified and
    updated automatically.
• We’ll go through in-class how we can apply
  Observer to this problem

What is the Software Design for the Problem?

• The design consists of multiple views of the
  software structure we have constructed and
  computations we have defined
  – Static view (e.g., class diagram) shows how the
    problem has been decomposed into parts (e.g.,
    classes) and the static relationships between
    those classes (e.g., associations and
    aggregations)

What is the Software Design for the Problem?

– Dynamic view (e.g., sequence diagrams) shows
  how the parts interact to solve the problem
• We can analyze these views to see if they
  support the requirements
  – Modifiable (i.e., adding new views)?
  – Performance?
• Some iteration necessary in this case…

Software Design (Software Engineer’s Perspective)

• Is an iterative process
• A design consists of multiple views (both
  static and dynamic)
• A design is evaluated against goals
  (requirements), often using standard
  properties (e.g., coupling and cohesion)