ECE450 – Software Engineering II

Today: **Closing notes – A birds' eye view on patterns**

Where do we stand?

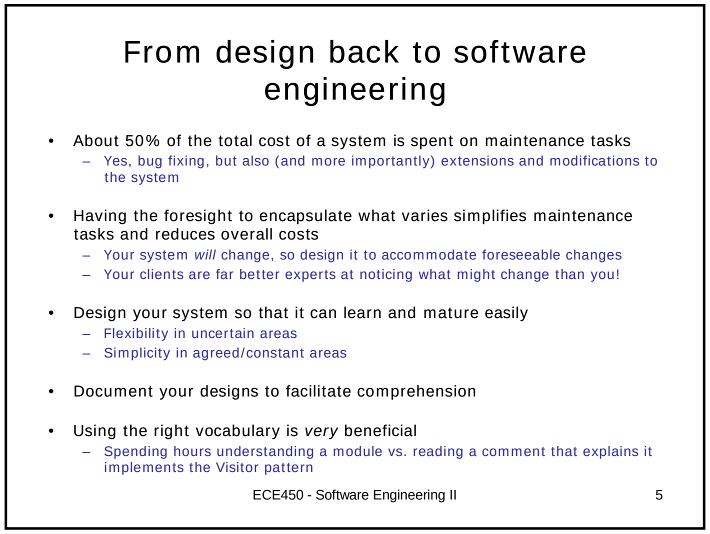
- We have discussed the 23 original design patterns
  - Many of them may seem irrelevant to you at this point
  - Some of them may seem more cumbersome than the problem they intend to solve
    - Perhaps you have not experienced the problem yet?
  - In any case, the patterns may be just a blur in your brain right now
    - What was the difference between Decorator and Strategy again...?
- Additional problem: We've only explored the surface of the field
  - More creational, structural, and behavioral patterns
  - Concurrency patterns
  - Systems analysis patterns
  - Reengineering patterns from code
  - Refactoring code to adjust to patterns
  - ...

Your approach to patterns

- Most people that are learning patterns follow a similar path:
- Step 1:
  - Know enough to be dangerous, not enough to be useful
  - Including patterns like crazy
  - Factory method for a Hello World application
  - Annoying, but perhaps necessary
    - ...since expertise comes from practice
- Step 2:
  - Expert in a few patterns, know to tell when to use them
  - Designs with one or two well-fitted patterns
  - Masters the vocabulary and communicates with the right terms with others
- Step 3:
  - Know the topic in depth, enough to tweak implementations and know when to ignore other experts' advice
  - Masters the principles *behind* pattern use
  - Using patterns as an extension of their craft, not following a template

Design principles distilled from patterns

- Encapsulate what varies
- Depend on abstractions, not on concrete classes
- Open for extension, closed for modification
- Minimize coupling
- Maximize cohesion
- Favor composition over inheritance



From design back to software engineering

- About 50% of the total cost of a system is spent on maintenance tasks
  - Yes, bug fixing, but also (and more importantly) extensions and modifications to the system
- Having the foresight to encapsulate what varies simplifies maintenance tasks and reduces overall costs
  - Your system *will* change, so design it to accommodate foreseeable changes
  - Your clients are far better experts at noticing what might change than you!
- Design your system so that it can learn and mature easily
  - Flexibility in uncertain areas
  - Simplicity in agreed/constant areas
- Document your designs to facilitate comprehension
- Using the right vocabulary is *very* beneficial
  - Spending hours understanding a module vs. reading a comment that explains it implements the Visitor pattern

ECE450 - Software Engineering II 5