CSC407
Concluding remarks

Is that all there is?

• Design and Architecture is about communicating between people.
• The machine doesn’t much care.
• Clean, elegant abstractions are easier to describe, modify, reuse.
• Sometimes subtle.

OO is a good thing?

• OO might represent progress because:
  • OOA allows us to (crudely and informally) model the world.
  • OOP helps us build software that incorporates a model of the world.
  • Thus, the innards of our software reflects the concepts that experts use to relate to their world.

UML is a good thing?

• UML represents progress because:
  • It standardizes many competing bubble and line diagram formats and tools.
  • Programmers like bubbles and lines.
  • You can use it for concepts (OOA) as well as software specification (OOD)
**Design is really hard**

- ...so hard that a tremendous proportion of engineers don’t even *try* to do it.
- We need to save our strength for the unique designs we must create by reusing other people’s designs when possible.
- Patterns help us use designs created by good engineers who preceded us.

**Composition**

- Creating complicated objects by composing potentially many simple objects seems to be a critical feature of good designs.
- Hence we focused on patterns that helped us do this.
- Linton et al invented several important patterns.
- Frameworks rely on composition.

---

**OOA**

- Use UML to model the static concepts that a analyst deems important to understanding the domain.
- Static, not time varying.
- Write use cases to record the interactions various actors have with the system.
- Use cases “drive” the concepts through the (time varying) transformations that occur as the system is used to get real work done.

**UML is really big**

- (In the sense that it contains many different diagrams already.)
- But nevertheless I follow Rosenberg’s suggestion to draw yet another diagram for preliminary design.
- Robustness diagrams let the use cases take the concepts through real required workflows.
- Eases jump to sequence diagrams.
- We discover new concepts, messages and attributes in the process.
Patterns

<table>
<thead>
<tr>
<th>Scope</th>
<th>Creational</th>
<th>Structural</th>
<th>Behavioral</th>
<th>Storage</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Factory method</td>
<td>Adapter, Template Base</td>
<td>Interpreter, Template Method</td>
<td>Object File, RDBMS</td>
<td>Proxy, Attribute Factory</td>
</tr>
<tr>
<td>Object</td>
<td>Abstract Factory, Builder, Prototype, Singleton</td>
<td>Adapter, Bridge, Composite, Decorator, Facade, Proxy</td>
<td>Chain of Responsibility, Command, Iterator, Mediator, Moehan, Flyweight, Observer, State, Strategy, Visitor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

lexi

- Is interesting because it was an extreme example of how objects can be lightweight, abstract and get the work done.
- Shows how flyweight can work
  - Points out importance of thinking about extrinsic vs intrinsic state
- Puts creational patterns to work
- Shows visitor at work

Creational Patterns

- are important because:
- Now that we propose composing complex objects out of many abstract ones it is a complex task to connect them all up!

Behavioral Patterns

- Are important because they illustrate how to allow parts to work together
  - Observer
  - Visitor
- Yet how to avoid tying pieces together so closely that reuse won't be possible.
Architecture

- Our understanding of architecture is less developed (even) than design.
- Architectural Design Languages are research projects at the moment.
- Experience has shown that if a system is described from several perspectives a reasonable understanding can be conveyed.
- Logical, implementation, process, deployment, data, performance, quality, etc.

Architecture styles

- Shaw tutorial points out that a similar approach can be used to reuse architecture as design.
- Architectural styles are to architecture as patterns patterns are to design.
- Data flow, call-and-return, data centred, vm, independent communicating components are examples.

Common business system architecture

- This course compares monolythic, client-server and triple-tier systems.
- Not as general as Shaw's styles but a very pragmatic classification that distinguishes many modern systems.
- And most business systems.
- Many IT practitioners probably imagine that's all there is.

The exam

- Quite different style than historical.
- Q1 5 x 5 mark small answers.
- Q2 3 x 5 mark really small answers.
- Q3 30 mark OOD (&robustness).
- Q4 30 mark pattern and design.
The End

- Thanks for your patience and good humor.
- I particularly enjoyed the conversations with the office hours “regulars”.
- Please fill in the review form.
- The results, especially constructive comments or astute criticisms, are typed in (by a service bureau) and carefully studied.