Tutorial 3
More on Design patterns

Study Several Examples of Design Patterns
Explain its relation to our course projects
Last lecture…
On design patterns

• We explained what are patterns, what are design patterns
• How are they categorized?
• How to apply them?
• How to identify them?
• How to assess them?
Today…

1. Design patterns structures
   Creational patterns
   Structural patterns
   Behavioural patterns

2. How are they related to each other?

3. Design patterns by examples
   Some special design in OpenOME

4. Their relation to your course project
1. The GOF Catalogue

• Creational
  Abstract Factory, Builder, Factory method, Prototype, Singleton

• Structural
  Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy

• Behavioural
  Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor
Adapter

Composite

Spring 2005

ECE450H1S

Software Engineering II

Decorator
• Template Method

• Visitor

Spring 2005 ECE450H1S Software Engineering II
2. Relation among patterns

Ladan Tahvildari and Kostas Kontogiannis. “On the Role of Design Patterns in Quality-Driven Re-engineering”

Spring 2005    ECE450H1S    Software Engineering II
A layered version
3. Some Special design patterns in our legacy software

1. MVC patterns
classic design pattern from SmallTalk
Most editors follows the pattern

2. Plugin patterns
OpenOME, Protégé, Eclipse

3. Meta-modelling patterns
Telos, EMF, UML, Protégé
3.1 MVC

Model updates Views when data changes

Event is passed to the Controller

Controller changes Model or View(s)

Views get data from Model

Subject (abstract)
- +Attach(observer Observer): void
- +Detach(observer Observer): void
- +Notify():
  for all o on observers {
    o.Update();
  }

Observer (interface)
- +Update():

Concrete Subject
- +subjectState: State
- +GetState(): State
- +SetState(state: State): void

Concrete Observer
- +Update():

Spring 2005 ECE450H1S Software Engineering II
3.2 Plugin patterns

AbstractPlugin
attribute
<<abstract>> plugin_method()

ConcretePlugin
attribute = value
plugin_method()
3.2.1 OpenOME

- AbstractPluginMethod.java
  - PluginMethod.java

- OMEPlugin.java
  - OMEDefaultPlugin.java
  - A bunch of methods
  - Extended by ...

- *edu.toronto.cs.ome.plugins*
  - ERPlugin.java
  - NFRPlugin.java
  - IStarPlugin.java
  - ...

- Plugin is selected at run-time, depending on the input class. ForName( ... )
3.2.2 Protégé

- **ClsWidget**, **ExportPlugin**, **ImportPlugin**, **ProjectPlugin**, **SlotWidget**, **TabWidget**, **Widget**

- Plugins are packaged into a JAR file, under the “plugins” subdirectory

- OMETab.java is a TabWidget plugin packaged as
  
  `plugins/edu.toronto.cs.ome/OpenOME.jar`
3.2.3 Eclipse

http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html
And many articles on its plugin developments ... plugin.xml, feature.xml

Spring 2005    ECE450H1S    Software Engineering II
4. Think about these …

• How would you classify the classes in \texttt{edu.toronto.cs.ome.OME} into the MVC pattern?

• Which design pattern is used by Web-Service projects?

• Which basic design patterns are used by the aforementioned Plugin patterns?
5. Relation to your project

• Opportunities:
  – You may add junit test cases to the code base to reveal bugs (publish it to the bug tracking system) and fix them (+5%)
  – You may apply design patterns, refactoring techniques on this legacy code base, showing as an improved complexity metrics (+2.5%)
  – You may tune the performance of the system to speed up the display, load/save for scalable graphs (+2.5%)

• Don’t forget your major project task (up to 100%!)
  – To study the editor methods in the OpenOME and adapt them to the OmniGraphEditor web service.