CmpE 202 – Software Systems Engineering

Team Project #3

(Optional)

Implement one of the architectural patterns from Team project #2 with one or more of the object-oriented languages of your desire, such as Java, C++, etc.

Ask the instructor for more details on the project

• **Design & Implementation Issues**: For each EBT, discuss the important issues required for linking the analysis phase to the design phase and for each BO, discuss the important issues required for linking the design phase to the implementation phase, for example hooks.

<table>
<thead>
<tr>
<th>Length: 1/4 – 1 Pgs Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Issues (EBT)</td>
</tr>
<tr>
<td>For example hooking issues</td>
</tr>
<tr>
<td>Implementation Issues (BO)</td>
</tr>
<tr>
<td>For example why using aggregation or delegation rather than inheritance</td>
</tr>
<tr>
<td>For example hooking, hot spots problems</td>
</tr>
<tr>
<td>Can show code here.</td>
</tr>
</tbody>
</table>

Here is a list of Analysis Issues:

• Divide and conquer
• Understanding
• Simplicity
• One unique base that suitable to many applications
• Goals
• Fitting with business modeling
• Requirements Specifications Models
• Packaging
• Components
• Type (TOP) (A)
• Actors/Roles
• Responsibility and Collaborations
• Generic and Reusable models
• Etc,

Here is a list of design and implementation issues:

• Framework models (D)
• Static Models (D)
• Classes (TOP) (D)
• Collaborations (D)
• Refinement (D)
• Generic and Reusable Designs (D)
• Precision (I)
• Hooks (I)
• Pluggable Parts (I)
• Navigation (I)
• Object Identity (I)
• Object State (I)
• Associations/Aggregations (I)
• Collections (I)
• Static Invariants (I)
• Boolean Operators (I)
• Collection Operators (I)
• Dictionary (D) (I)
• Behavior models (D) (I)
• Pre-Post-conditions specify actions (I)
• Joint Actions (Use Cases) (D)
• Localized Actions (I)
• Action Parameters (I)
• Actions and Effects (I)
• Concurrent Actions (I)
• Collaborations (I)
• Interaction Diagrams (D)
• Sequence Diagrams with Actions (D) (I)
• Pattern 1: Continuity
• Pattern 2: Performance
• Pattern 3: Reuse
• Pattern 4: Flexibility
• Pattern 5: Orthogonal Abstractions
• Pattern 6: Refinement
• Pattern 7: Deliverables
• Pattern 8: Recursive Refinement
• Package (D) (I)

Here is a list of Java Patterns:
• Fundamental Design Patterns
  1. Delegation (When not to use Inheritance)
  2. Proxy
• Creational Patterns
  1. Abstract Factory
  2. Builder
  3. Factory Method
  4. Object Pool
  5. Prototype
  6. Singleton
• Partitioning Patterns
  1. Composite
  2. Filter
  3. Layered Initialization
• Structural Patterns
  1. Adaptor
  2. Bridge
  3. Cache Management
4. Decorator
5. Dynamic Linkage
6. Façade
7. Flyweight
8. Iterator
9. Virtual Proxy

• Behavioral Patterns
  1. Chain of Responsibility
  2. Command
  3. Little Language / Interpreter
  4. Mediator
  5. Null Object
  6. Observer
  7. Snapshot
  8. State
  9. Strategy
 10. Template Method
 11. Visitor

Here is a list of UML Patterns:

• GRASP Patterns
  1. Controller
  2. Creator
  3. Expert
  4. Law of Demeter
  5. Low Coupling/High Cohesion
  6. Polymorphism
  7. Pure Fabrication

• Organizational Coding Patterns
  1. Accessor Method Name
  2. Anonymous Adapter
  3. Checked versus Unchecked Exceptions
  4. Client Socket
  5. Composed Method
  6. Conditional Compilation
  7. Covert Exceptions
  8. Define Constants in Interfaces
  9. Extend Super
 10. Intention Revealing Method
 11. Server Socket
 12. Switch
 13. Symbolic Constant Name

• Code Optimization Patterns
  1. Double Checked Locking
  2. Hashed Adapter Objects
  3. Lazy Initialization
  4. Lookup Table
  5. Loop Unrolling

• Code Robustness
  1. Assertion Testing
  2. Copy Mutable Parameters
  3. Guaranteed Cleanup
  4. Maximize Privacy