Applying Design Patterns to SCA Implementations

Adem ZUMBUL (TUBITAK-UEKAE, ademz@uekae.tubitak.gov.tr)
Tuna TUGCU (Bogazici University, tugcu@boun.edu.tr)

SDR Forum Technical Conference, 26-30 October 2008, Washington, USA
Outline

✓ Design Patterns
  ➢ Creational
  ➢ Structural
  ➢ Behavioral

✓ Applying Design Patterns to SCA
  ➢ Factory Method
  ➢ Chain of Responsibility
  ➢ Adapter
  ➢ Singleton
  ➢ Facade

✓ Conclusion
Design Patterns

✔ Design Patterns

➢ Optimum solutions to common software engineering problems

➢ Applying Design Patterns
  ❖ Speed up development process
  ❖ Provides tested and proven design paradigms
  ❖ Improves code readability

➢ Become famous after
  ❖ “Design Patterns: Elements of Reusable Object-Oriented Software”, Gang of Four (GoF)
Design Patterns

✓ Design Patterns

➤ Creational Patterns
  ❖ Abstract Factory
  ❖ Builder
  ❖ Factory
  ❖ Prototype
  ❖ Singleton

➤ Deals with object instantiation problems
➤ Solves object creation problems
Design Patterns

✓ Design Patterns

➢ Structural Patterns
  ❖ Adapter
  ❖ Bridge
  ❖ Composite
  ❖ Decorator
  ❖ Facade
  ❖ Flyweight
  ❖ Proxy

➢ Define ways to compose objects

➢ Can be applied
  ❖ Design from scratch
  ❖ Modify existing systems
Design Patterns

 ✓ Design Patterns
  ➢ Behavioral Patterns
    ❖ Chain of Responsibility
    ❖ Command
    ❖ Interpreter
    ❖ Iterator
    ❖ Mediator
    ❖ Memento
    ❖ Observer
    ❖ State
    ❖ Strategy
    ❖ Template
    ❖ Visitor
  ➢ Solves object communication problems
  ➢ Depicts how objects act together
General Rules about Design Patterns

- All client objects should always call the abstraction
- Future changes should not impact the existing systems
- Change always what is changing
- Have loose coupling between objects
✓ Factory Method

- Defines an interface for creating objects
- Lets subclasses to decide which class to instantiate
Design Patterns

✓ Applying Factory Method
  ➢ Solves portability problems
    ❖ Separate OS and ORB specific codes with the rest of the systems
  ➢ Manages configuration specific issues
    ❖ Change only the configuration of the factory class
  ➢ Deals with future changes
  ➢ Allows insertion of new derived subclasses
Design Patterns

✓ Chain of Responsibility
  ➢ Avoids coupling of the sender of a message to its receiver
  ➢ Allows more than one object to handle the request
  ➢ It chains the receiving objects and passes the request along the chaining until an object handles it
Design Patterns

✓ Applying CoR

- Can be applied in conjunction with Port concept
- Components can be chained to handle incoming data or not
- Each component decides by checking
  - Permissions
  - Capacity values
  - Priorities
  - Dependencies
  - Performance reqs
  - Props of incoming data
- Allows insertion or removal of new components without affecting existing codes.
- Can be life saving for applications where requirements frequently change.
Design Patterns

✓ Adapter
  ➢ Converts the interface of a class to another interface
  ➢ Adapter lets classes to work together that otherwise can’t
  ➢ It simply adapts old interface to the new one
Design Patterns

✔ Applying Adapter
  ➢ Can be applied to adapt nonSCA code to SCA classes
  ➢ Typically not used to design from scratch but rather applied to modify existing design
Design Patterns

✓ Singleton

➢ Ensures a class to have only one instance
➢ Provide a global point of access
➢ Relatively simple pattern to apply
➢ From SCA point of view

❖ Device classes that wraps a hardware should have only one instance
  - To have only one capacity manager
  - Device cannot be initialized more than once.

❖ ORB classes can implement Singleton pattern to ensure central policy manager

❖ Can be applied in conjunction with Factory Method pattern to ensure returned concrete classes has only one instance.
Design Patterns

✓ State

➢ Allows an object to alter its behaviour when it’s internal state changes.

➢ Benefit of State pattern is specific code is localized in the class that represents that state.

➢ Reduces complexity by seperating state depending code with the rest of the system.

➢ Can be applied in conjunction with the SCA state variables

   ❖ ENABLED, DISABLED, SHUTTING_DOWN, LOCKED, UNLOCKED, IDLE, ACTIVE, BUSY
Design Patterns

✓ Facade
  ➢ Provides a unique interface to a set of interfaces
  ➢ Makes the subsystem easier to use
  ➢ Beautifies an existing cumbersome class by behaving a door to its complex interface
  ➢ Facade classes can be used as a wrapper to nonSCA classes to prevent re-implementing the existing codes
  ➢ It can also be used to collect separate CORBA interfaces into a single interface
Conclusions

✓ We have summarized Design Pattern concept and provided example application areas to some of them

✓ SCA tells developers
  ➢ “What to implement”
  ➢ Not “How to implement”
  ➢ Applying OOP concepts is essential to maximize the benefits of SCA
    ✷ Portability
    ✷ Reconfigurability
    ✷ Reusability

✓ Applying Design Patterns helps developers to implement better SDR applications