TS 8245 System Components Using Design Patterns on the Discovery Channel Telescope

Paul J. Lotz, Software Engineering Manager with Michael J. Lacasse, Ryan C. Godwin (Lowell Observatory)
Discovery Channel Telescope (DCT)
These results made use of the Discovery Channel Telescope at Lowell Observatory. Lowell is a private, non-profit institution dedicated to astrophysical research and public appreciation of astronomy and operates the DCT in partnership with Boston University, the University of Maryland and the University of Toledo.
A note on modeling: We use the Unified Modeling Language (UML). The modeling tool we employ is Enterprise Architect (EA).
System Concept

• Each system consists of a set of components that work together to achieve a larger goal.*

Component Definitions

• Component*
  – Modular
  – Encapsulates contents
  – Replaceable with another implementation that satisfies interfaces

• DCT component
  – Stand-alone
  – State-based
  – Interfaces: SubData, PubData

Publish-Subscribe

- Publish-subscribe (Observer Pattern)
  - Multiple subscribers
  - Multiple publishers
- Flexible deployment
Motivations for Template

- Issues with independent implementation
  - Repetition
  - Inconsistency
  - Mistakes

- Considerations
  - Uniqueness
  - Flexibility
  - Embraceability
Template Demonstration
Template Characteristics

• Common vs. modifiable
• Best practices
  – Design patterns
  – Existing implementations
  – Collaboration
• Simplicity
• Completeness
Template Application

- Copy
- Customize
Why National Instruments LabVIEW?

- Supports our Windows, RT (VxWorks), FPGA targets
- Dataflow—easily specify parallel operations
- Graphical programming
M/V/C with D

- Model/View/Controller (M/V/C) with Data Listener
Heartbeat

This image displays a software interface for managing a system, likely related to a telescope or similar equipment, given the context of the event mentioned (NI Week 2012). The interface includes buttons for starting and exiting the application, as well as various options for controlling and monitoring the system. The detailed state section shows options such as GoToStandby, Exit, and others, indicating functionalities to manage the system's state. The XML string and data string sections likely provide a view of the communication protocols or command structures used by the system.
Interfaces

• Background definitions:
  – Abstract class
  – Pure virtual method (abstract method)
  – Interface
    • As an interface in National Instruments LabVIEW we use a class with all pure virtual methods.*

• Motivations
  – Source code isolation
  – Design by contract

Design Patterns

- Proven solutions
- Cataloged* 
  - Name
  - Problem
  - Solution
  - Consequences

*Gamma, E., Helm, R., Johnson, R., and Vlissides, J., [Design Patterns: Elements of Reusable Object-Oriented Software], Addison-Wesley, Boston, (1995).
Command Pattern

```java
class CommandThisComponent {

    // CommandThisComponent
    + exec(IContextThisComponent) : IContextThisComponent

    // OnNewPositionDemand
    - demand : DBL [3] (Array) {sequence}
    + exec(IContextThisComponent) : IContextThisComponent

    // GoToStandby
    + exec(IContextThisComponent) : IContextThisComponent
```

NI Week 2012 -- Aug 05
State Pattern

- Objects for States

Attend my session TS8237 at 16:45 today to learn more about the State Pattern!
Adaptation of State Pattern

- Differentiate Context and Model
  - IContext → interface available to clients
  - IModel → interface available to state objects
Factory Method Pattern

- Dynamic object creation
- Source code isolation
Using a Factory: Call Interface Method
Composite View -- Demonstration
Configuration Editor -- Demonstration
Generic Component State Machine

stm OperateComponent

Initial

Standby State
- Exit
- StandBy

Off State
- Final

Disabled State
- Enable
- Disable
- [Fault]

Fault State
- [Fault]

Enabled State

Load configuration.
Component Relationships

- Hierarchical
- Relationship rules
  - Knowledge of child
  - Knowledge of parent
Component Relationships
Component Example Part 1
Component Example Part 2
Template Benefits

- Reuse
- Complete, functional framework
- Best practices
- Flexibility
- A place for everything
- Consistency
- Maintainability
- Scalable
- Working software!
Features LabVIEW Needs

Critical features LabVIEW must have for competitive development of large systems

• View improvements [Composite views should be the norm]
• Object serialization [Serialize in interchangeable format—possible now only with per-class methods]
• Integration with UML modeling tool [Round-trip connection to a model from standards-compliant, competitive UML tool]
Questions

paul.lotz@lowell.edu

TS8237: State Pattern Implementation for Scalable Control Systems
Date: 8/7/12
Time: 16:45-17:45
Room: 12A
Thank You!

paul.lotz@lowell.edu

TS8237: State Pattern Implementation for Scalable Control Systems
Date: 8/7/12
Time: 16:45-17:45
Room: 12A
Messaging with Objects

Sender

Prepare data → Serialize → Favorite messaging system

Data to send
Cast to parent, if necessary
Flatten to string

Serialize

Convert to byte array

(Optional: For RT-FIFO-enabled shared variable)

Deserialize

Convert to string

Unflatten from string

Use data

Original object on the wire!

Receiver

Favorites messaging system

Sender

Prepare data

Data to send

Cast to parent, if necessary

Flatten to string

Serialize

Convert to byte array

(Optional: For RT-FIFO-enabled shared variable)

Deserialize

Convert to string

Unflatten from string

Use data

Original object on the wire!