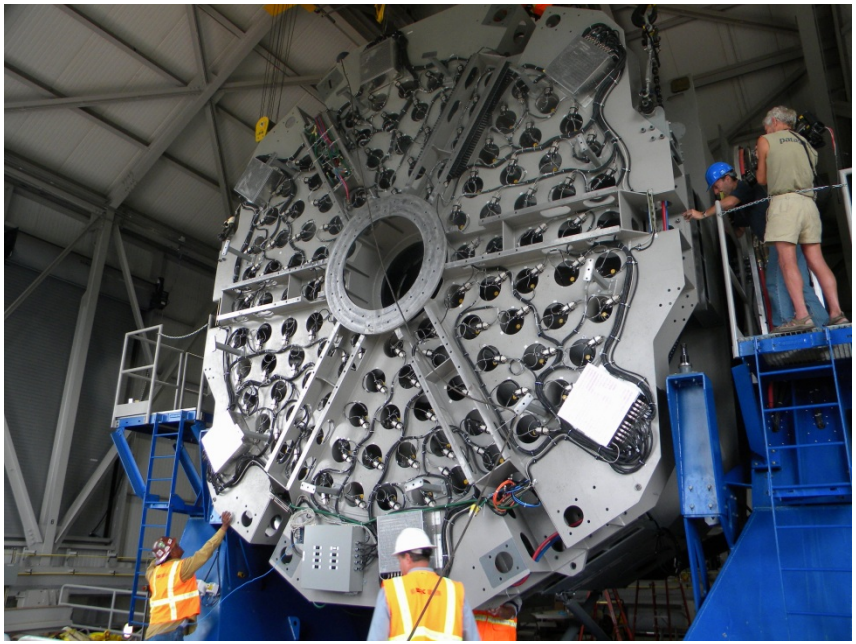
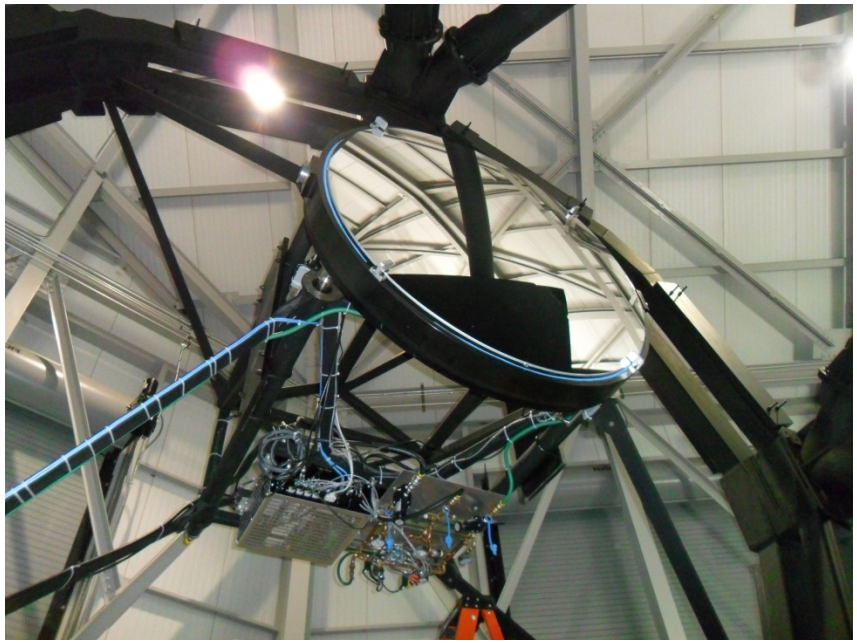
TS 8245 System Components Using Design Patterns on the Discovery Channel Telescope

Paul J. Lotz, Software Engineering Manager
with Michael J. Lacasse, Ryan C. Godwin
(Lowell Observatory)

Discovery Channel Telescope (DCT)

DCT

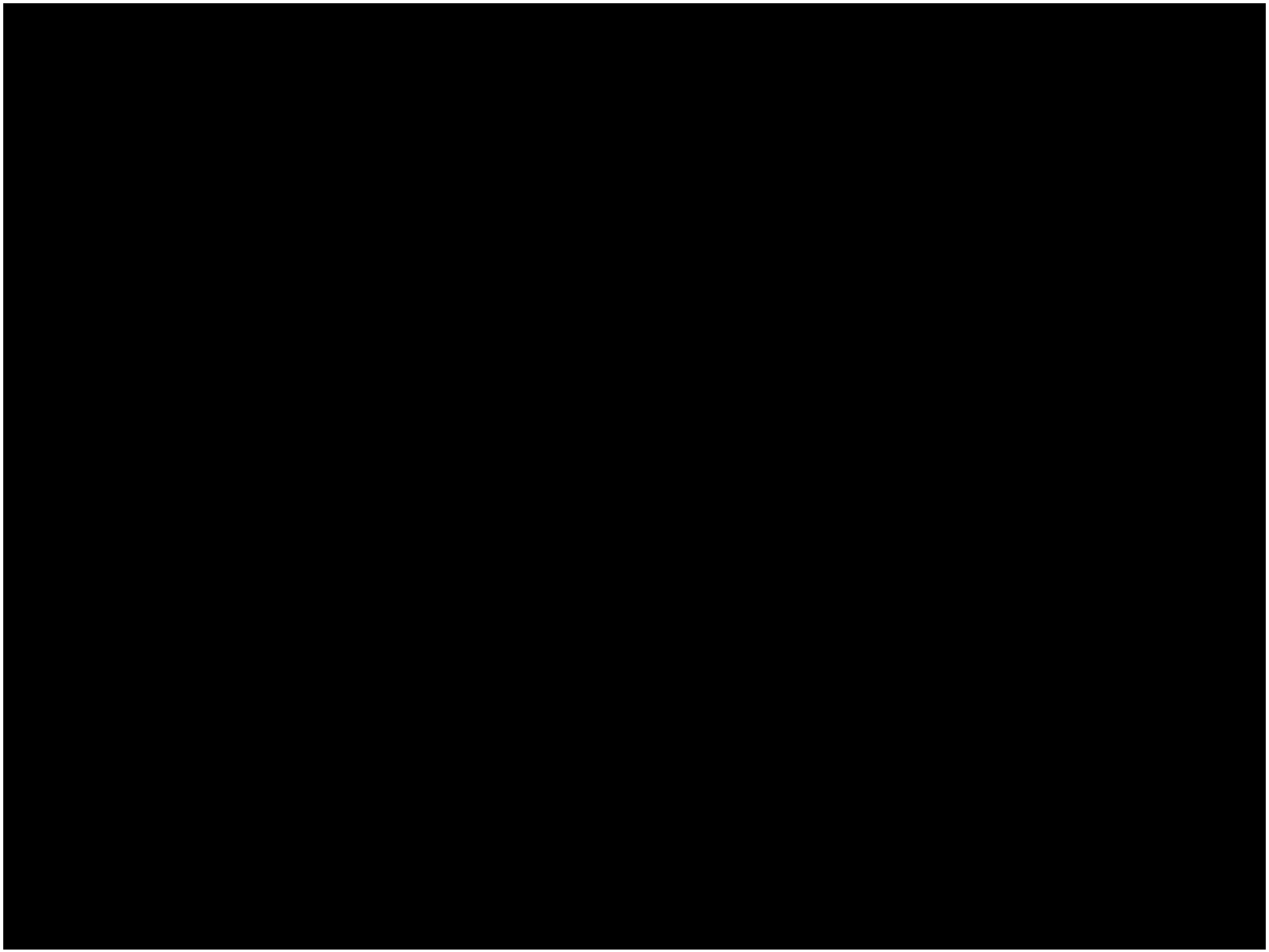




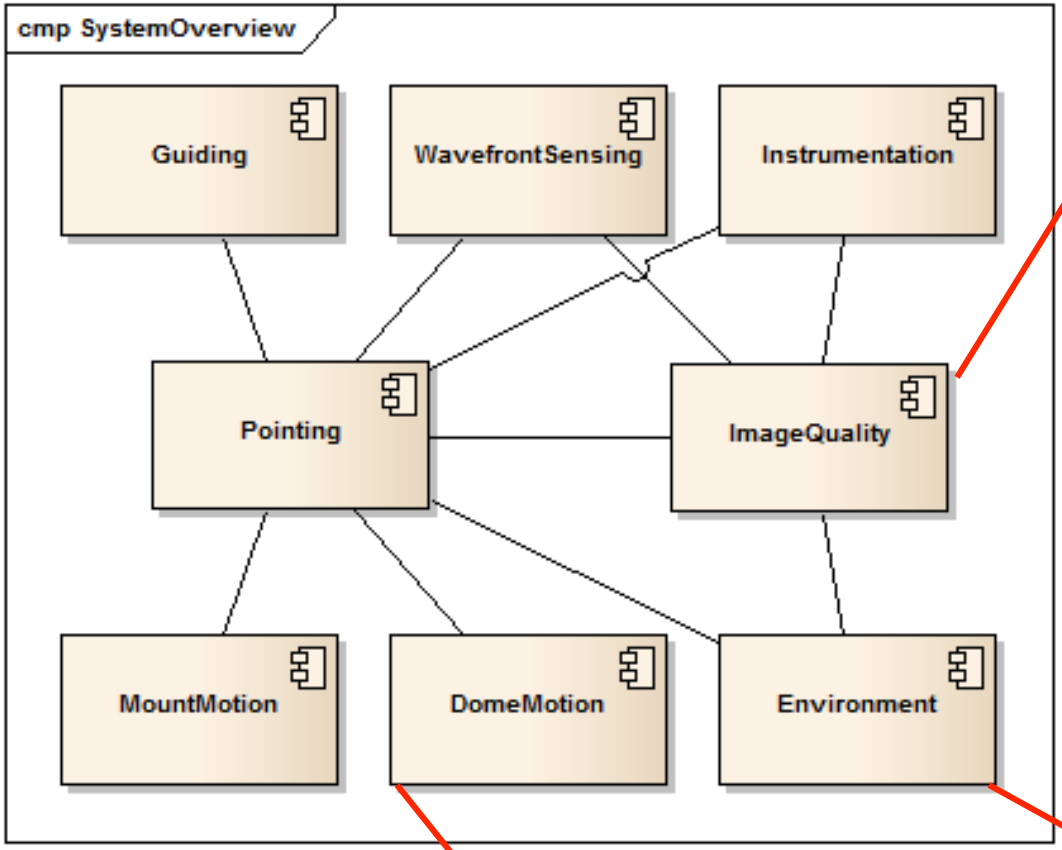
First Light Images

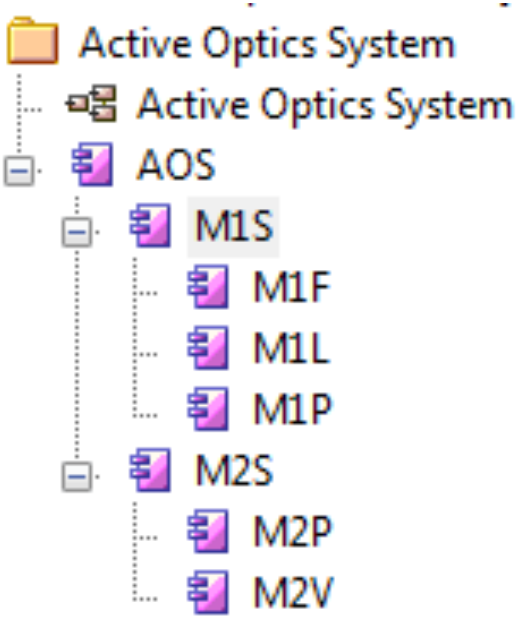
These results made use of the Discovery Channel Telescope at Lowell Observatory. Lowell is a private, non-profit institution dedicated to astrophysical research and public appreciation of astronomy and operates the DCT in partnership with Boston University, the University of Maryland and the University of Toledo.

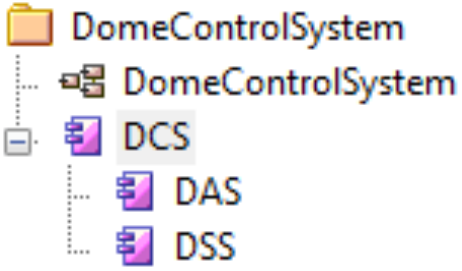
Software Team

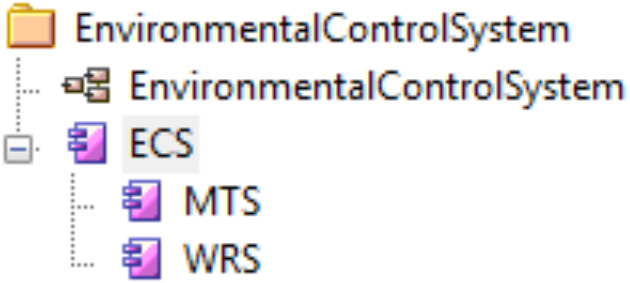


Some System Components









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.*

*Weilkiens, T., [Systems Engineering with SysML/UML: Modeling, Analysis, Design], Morgan-Kaufmann, Amsterdam, 8 (2006).

Component Definitions

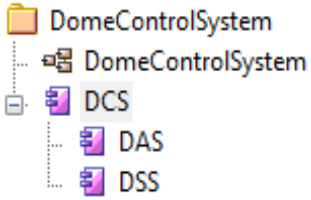
- Component*
  - Modular
  - Encapsulates contents
  - Replaceable with another implementation that satisfies interfaces
- DCT component
  - Stand-alone
  - State-based
  - Interfaces: SubData, PubData

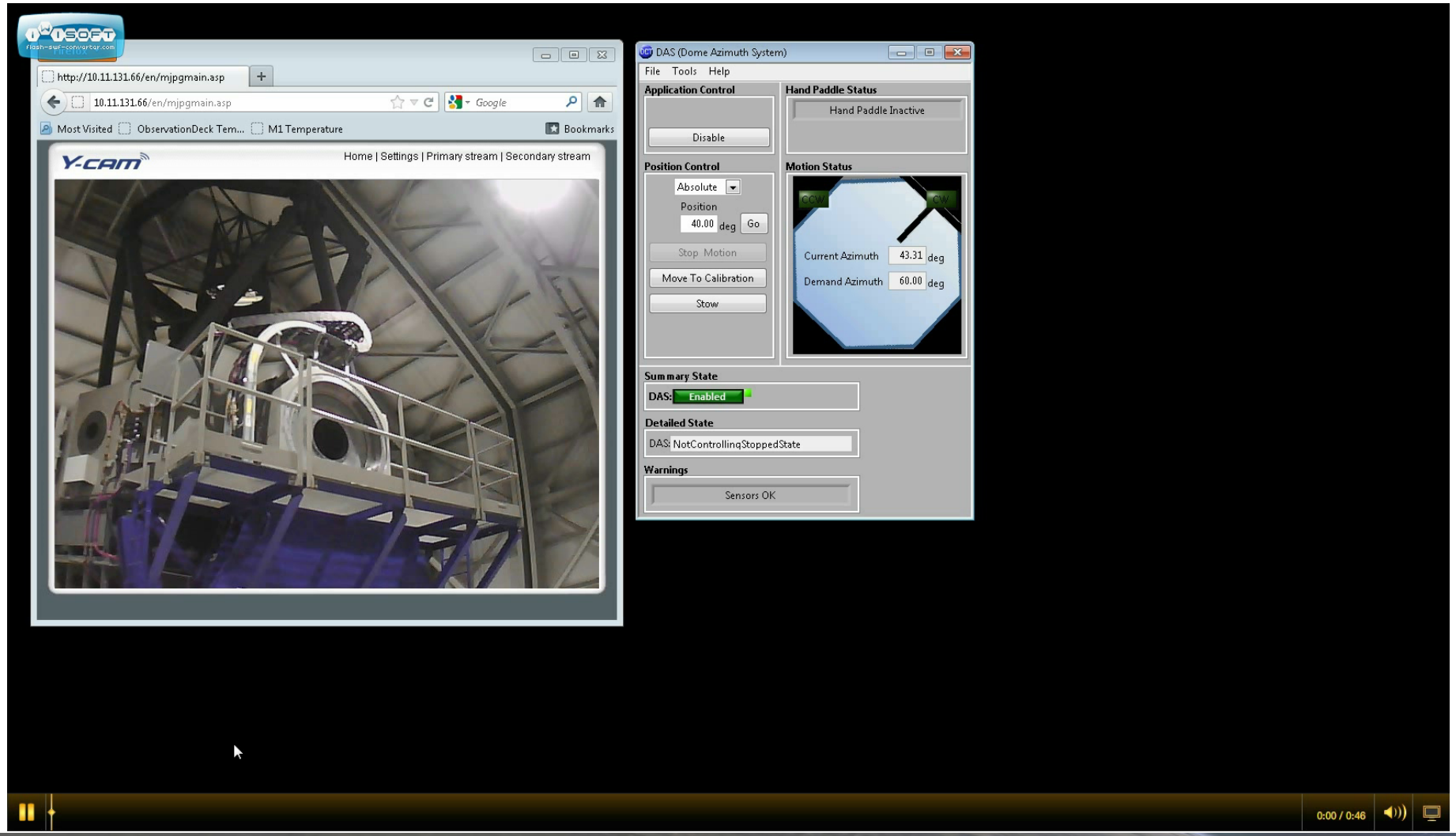
*Larman, C., [Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development], Pearson, Upper Saddle River, 654 (2005).

Publish-Subscribe

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

Publish-Subscribe

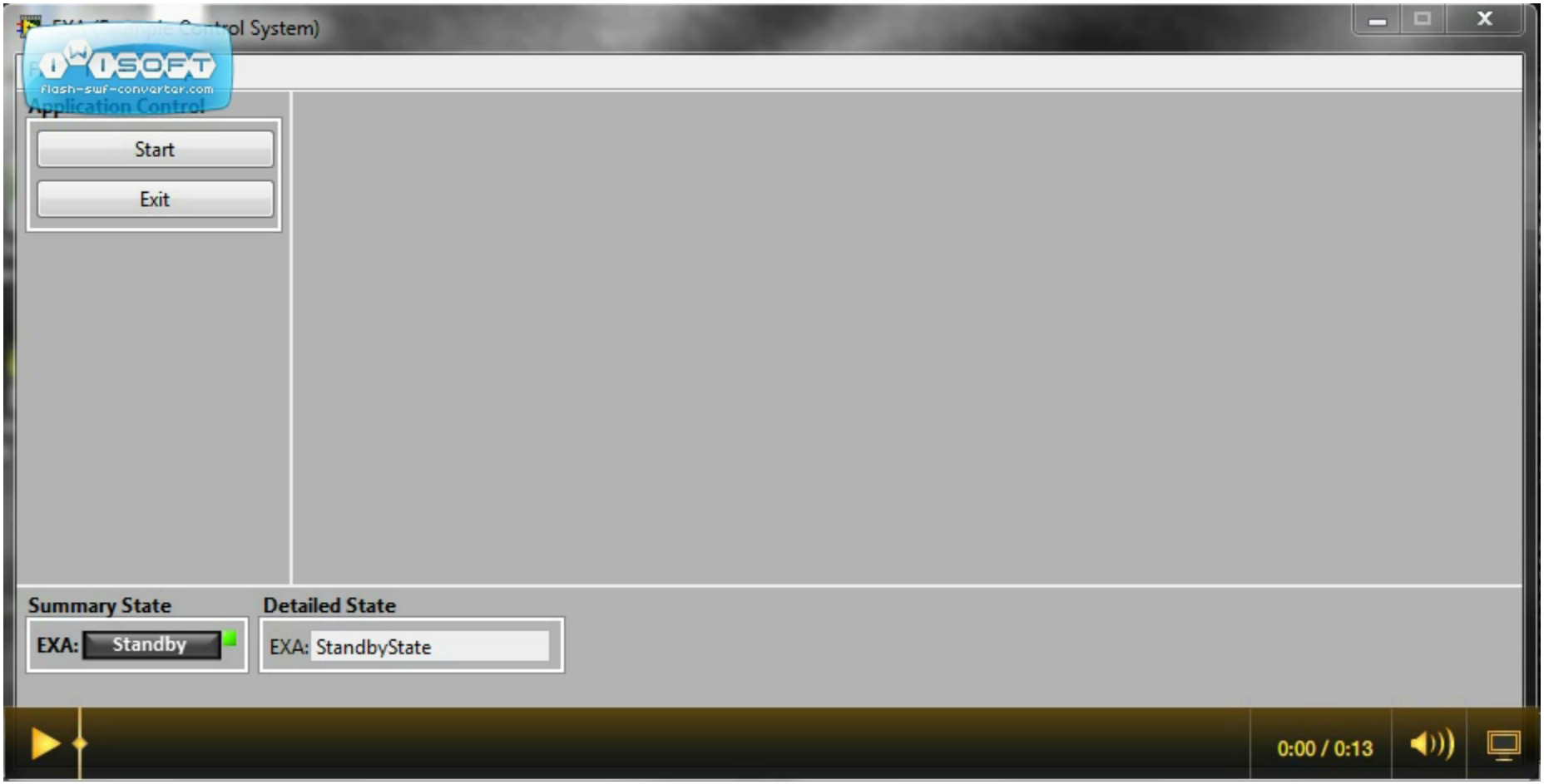




Motivations for Template

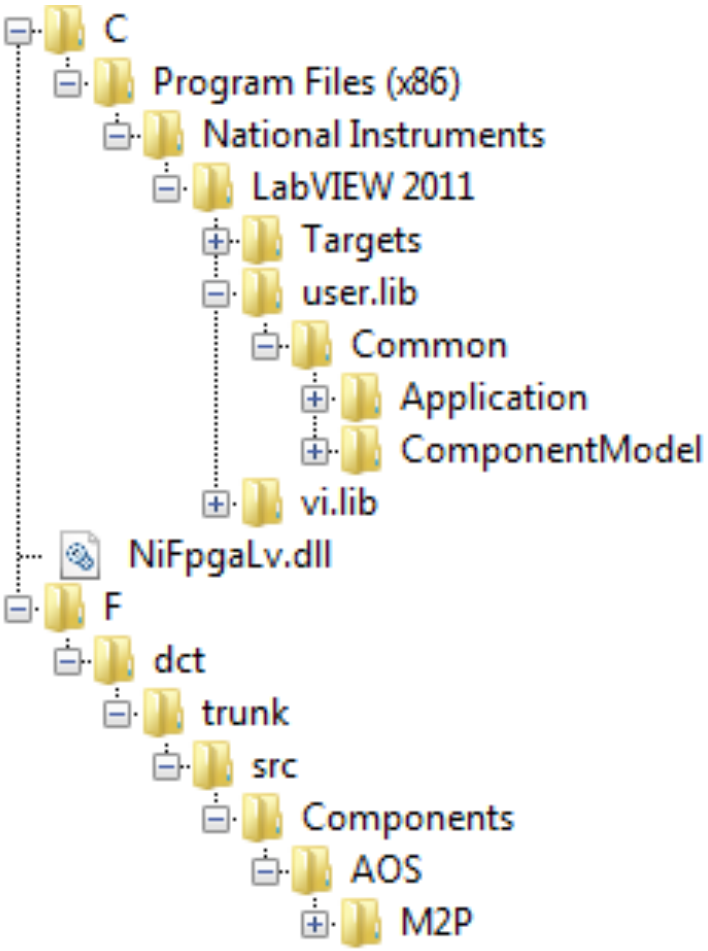
- Issues with independent implementation
  - Repetition
  - Inconsistency
  - Mistakes
- Considerations
  - Uniqueness
  - Flexibility
  - Embraceability

Template Demonstration



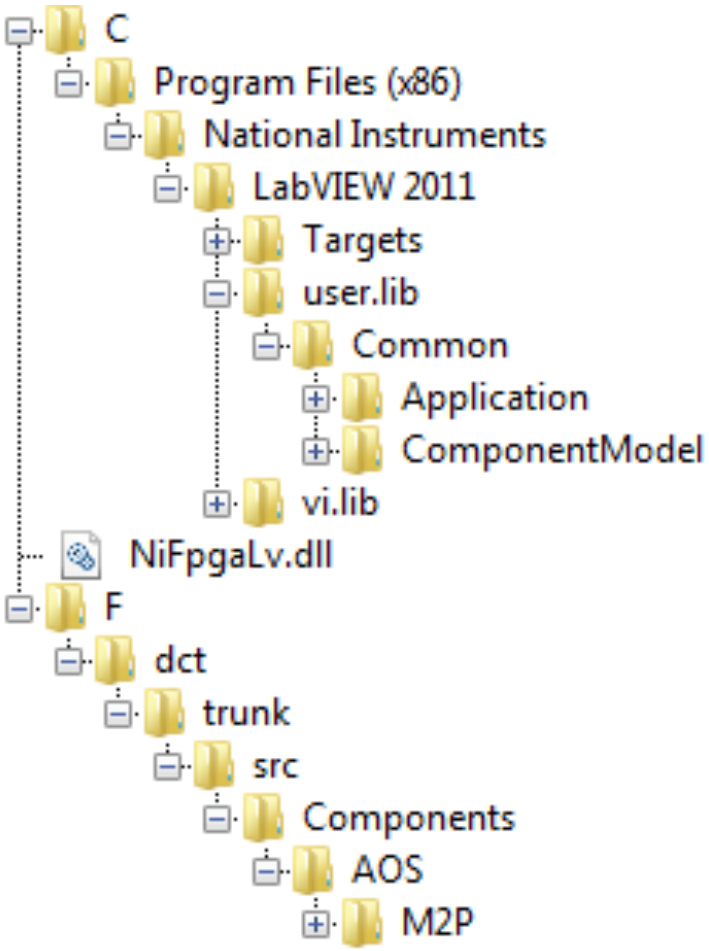
Template Characteristics

- Common vs. modifiable
- Best practices
  - Design patterns
  - Existing implementations
  - Collaboration
- Simplicity
- Completeness



Template Application

- Copy
- Customize



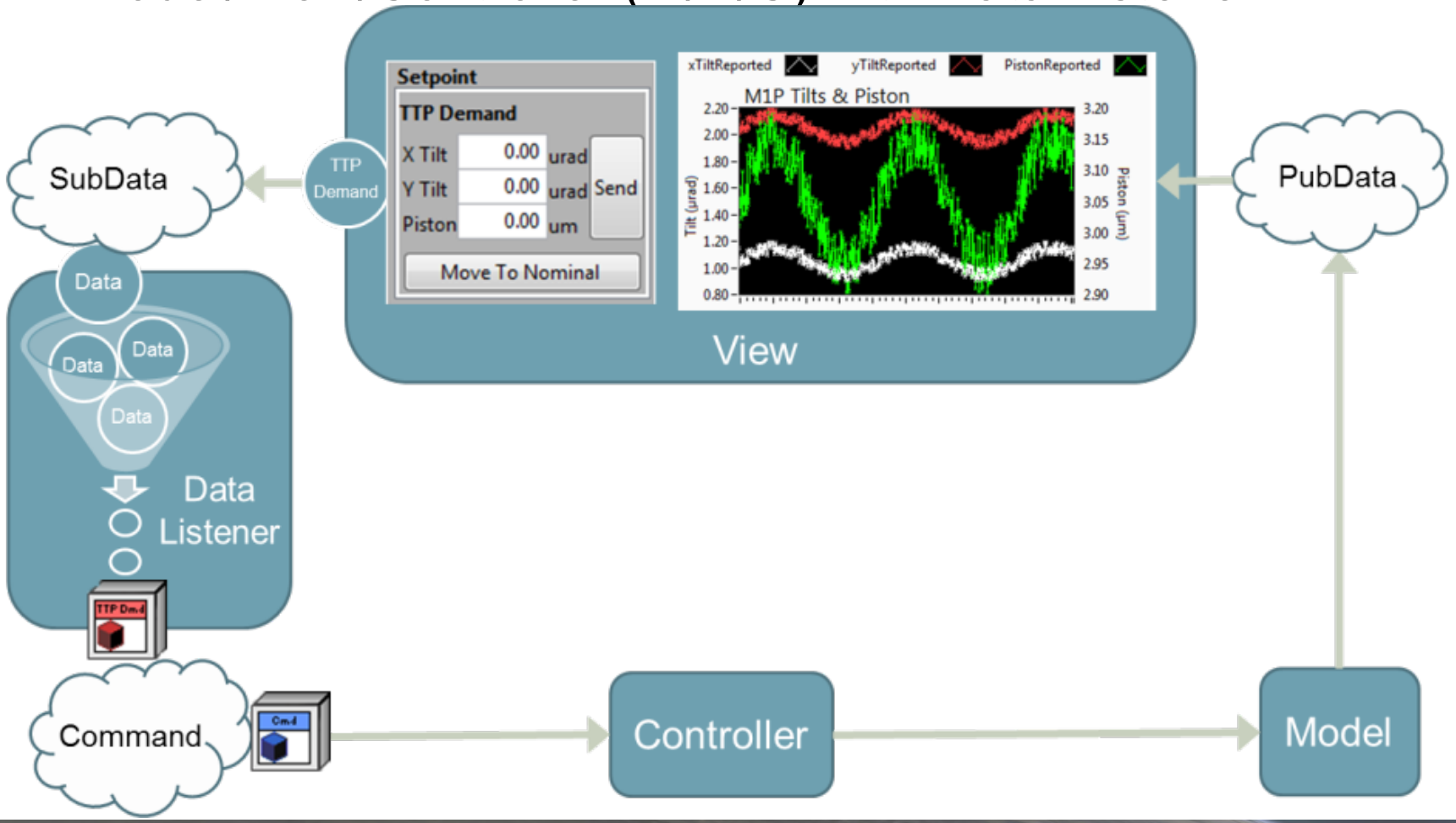
Why National Instruments LabVIEW?

- Supports our Windows, RT (VxWorks), FPGA targets
- Dataflow—easily specify parallel operations
- Graphical programming

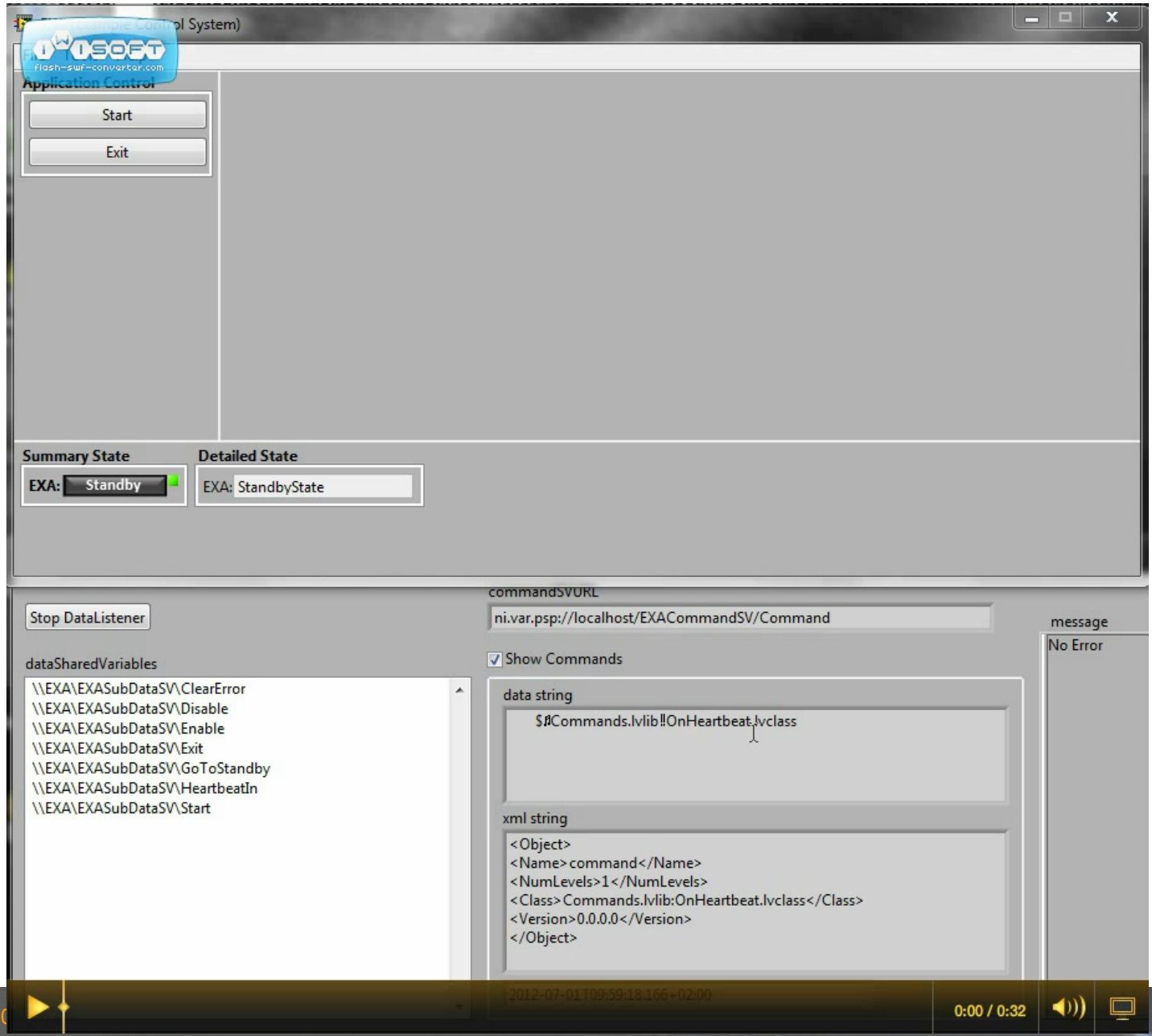
Architecture

M/V/C with D

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

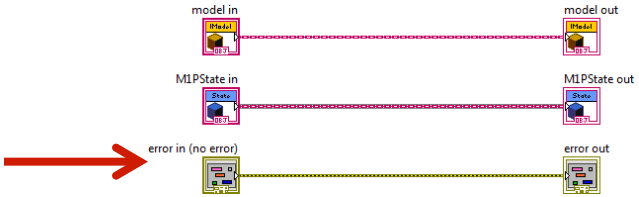


Heartbeat


Application Control
Start
Exit
Summary State
EXA: Standby
Detailed State
EXA: StandbyState
Stop DataListener
commandSVURL
ni.var.psp://localhost/EXACommandSV/Command
message
No Error
dataSharedVariables
\\EXA\EXASubDataSV\ClearError
\\EXA\EXASubDataSV\Disable
\\EXA\EXASubDataSV\Enable
\\EXA\EXASubDataSV\Exit
\\EXA\EXASubDataSV\GoToStandby
\\EXA\EXASubDataSV\HeartbeatIn
\\EXA\EXASubDataSV\Start
Show Commands
data string
$#Commands.lvlib!OnHeartbeat.lvclass
xml string
<Object>
<Name>command</Name>
<NumLevels>1</NumLevels>
<Class>Commands.lvlib:OnHeartbeat.lvclass</Class>
<Version>0.0.0.0</Version>
</Object>
0:00 / 0:32

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

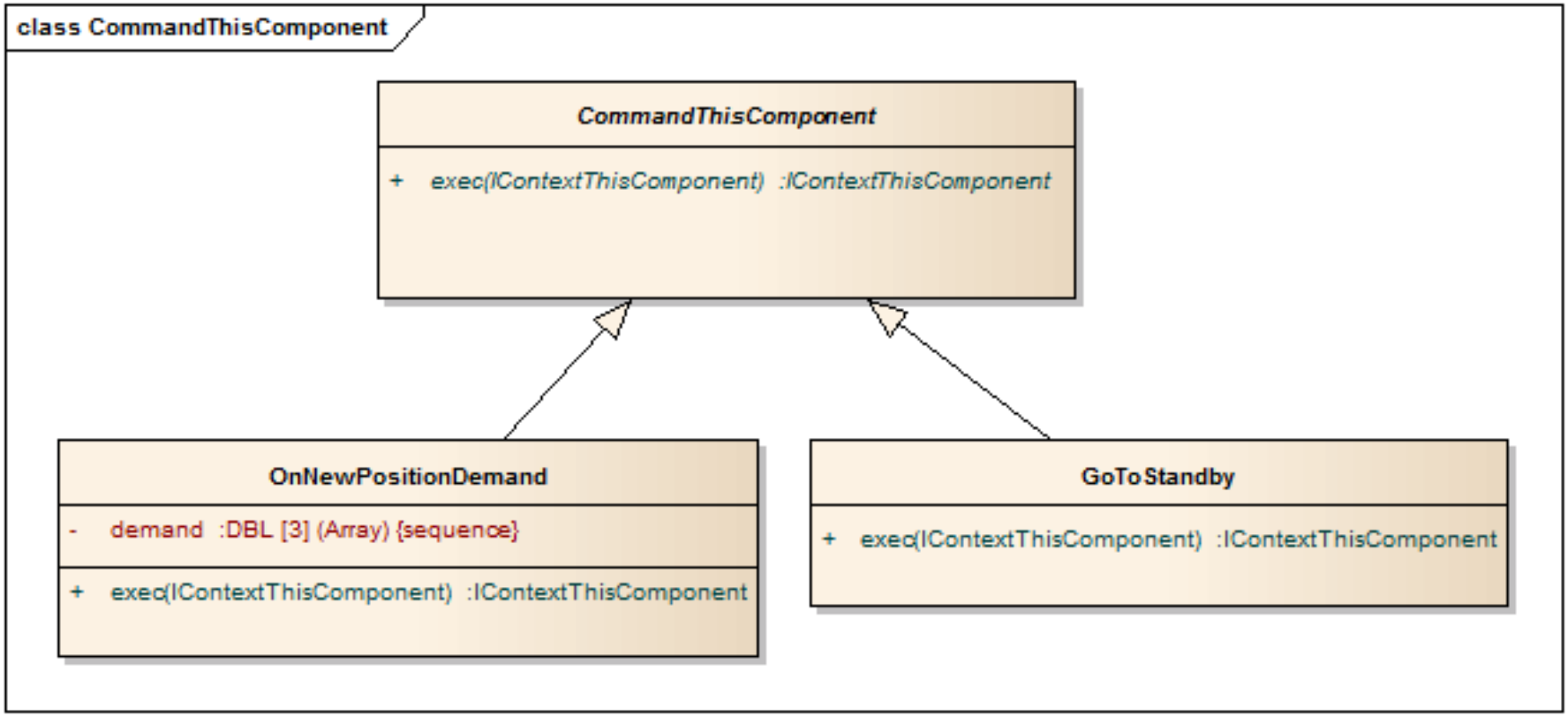
*Cf. Budd, T. [An Introduction to Object-Oriented Programming, 3d ed.], Addison-Wesley, Boston, 170 (2002).

Design Patterns

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

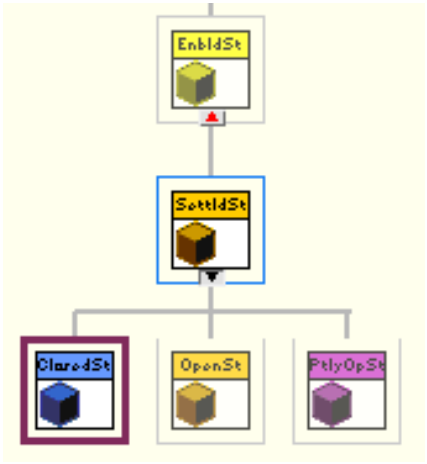
*Gamma, E., Helm, R., Johnson, R., and Vlissides, J., [Design Patterns: Elements of Reusable Object-Oriented Software], Addison-Wesley, Boston, (1995).

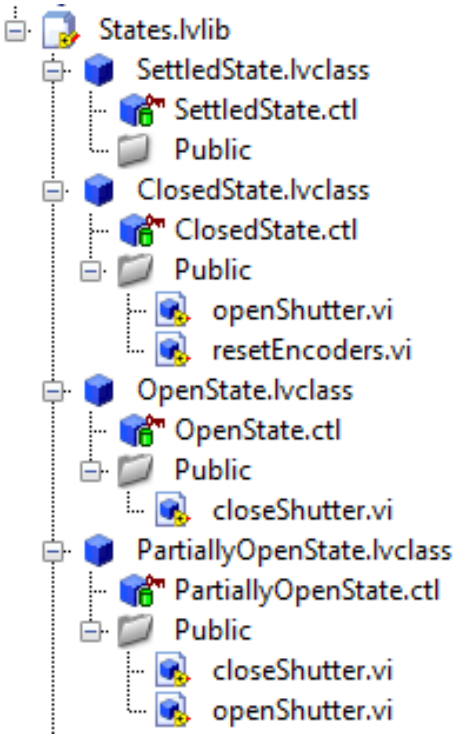
Command Pattern

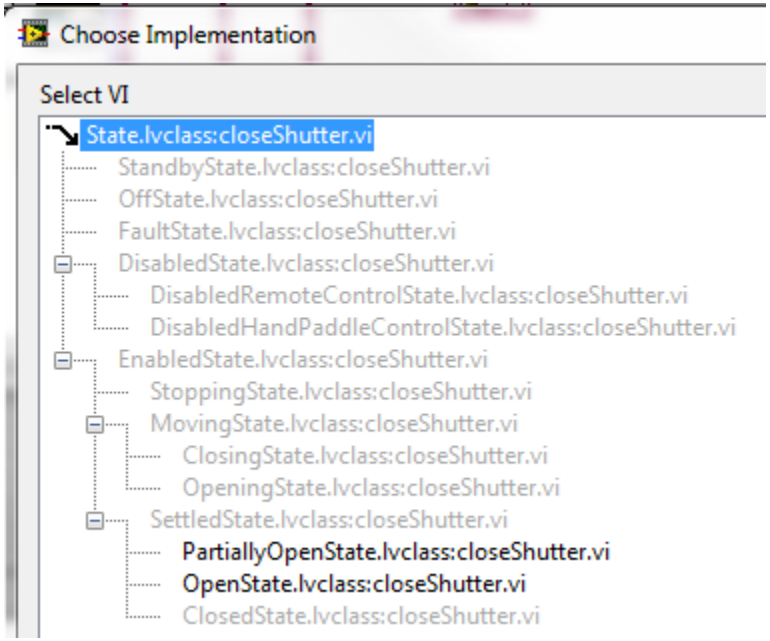


State Pattern

- Objects for States

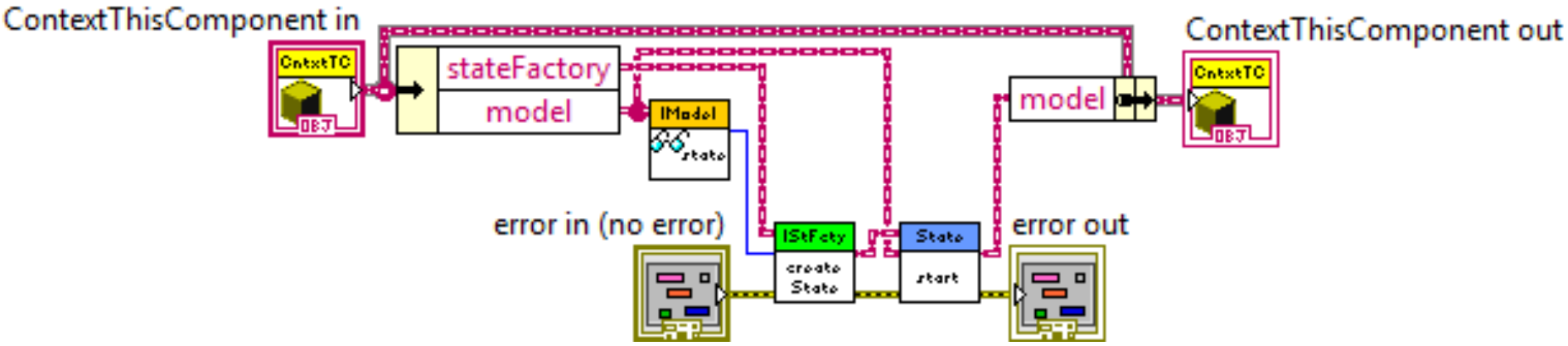






Attend my session TS8237 at 16:45 today to learn more about the State Pattern!

Adaptation of State Pattern

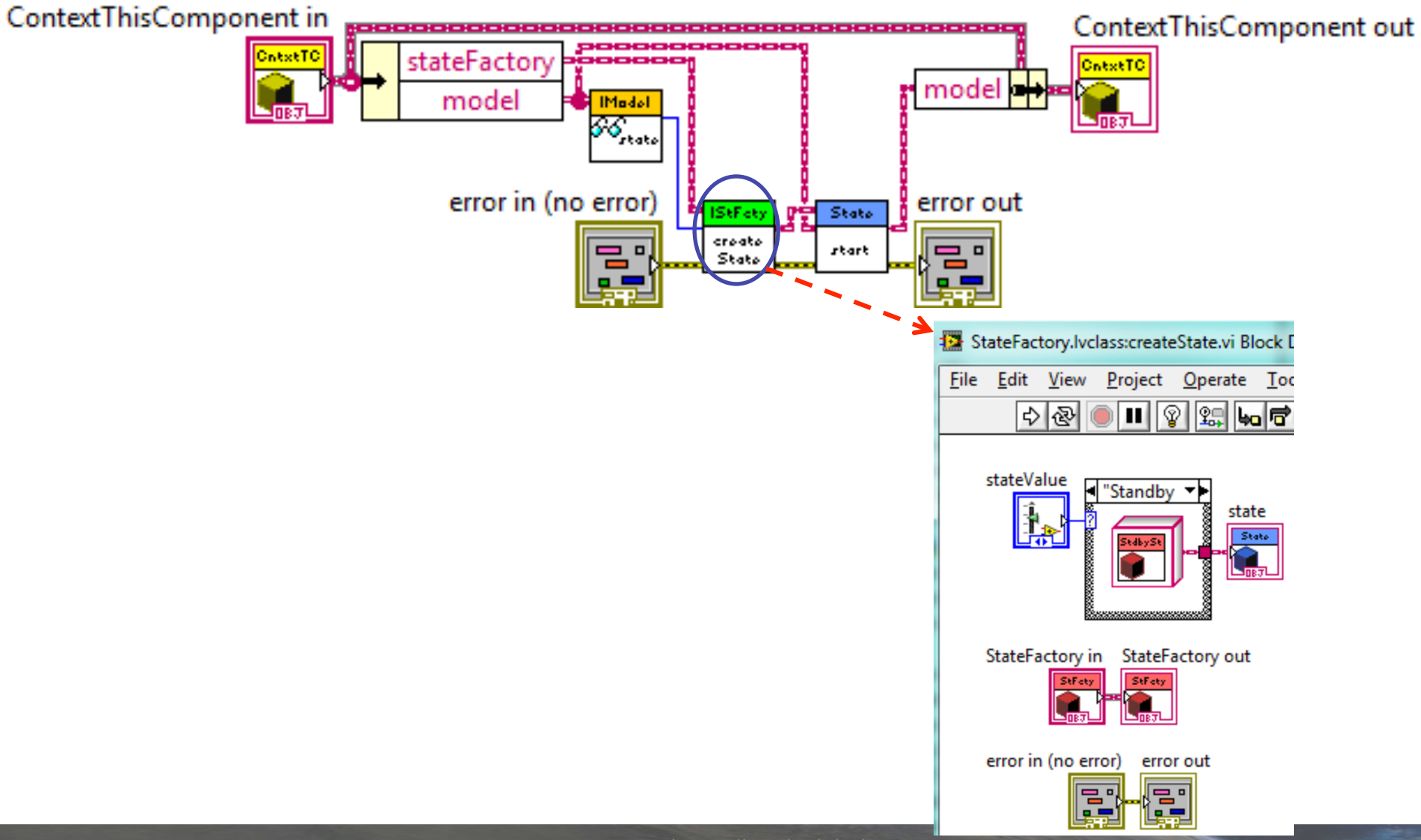


- Differentiate Context and Model
  - IContext → interface available to clients
  - IModel → interface available to state objects

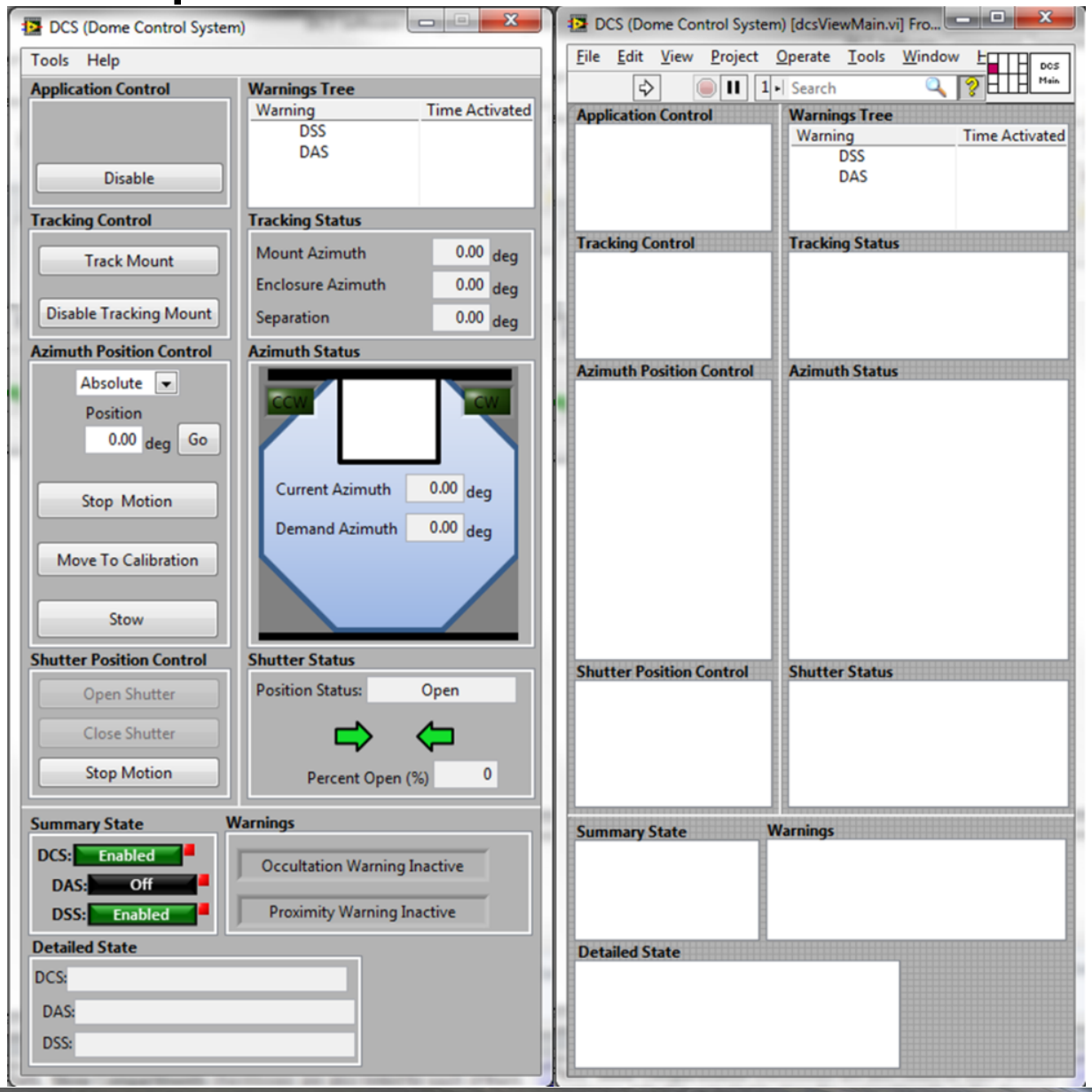
Factory Method Pattern

- Dynamic object creation
- Source code isolation

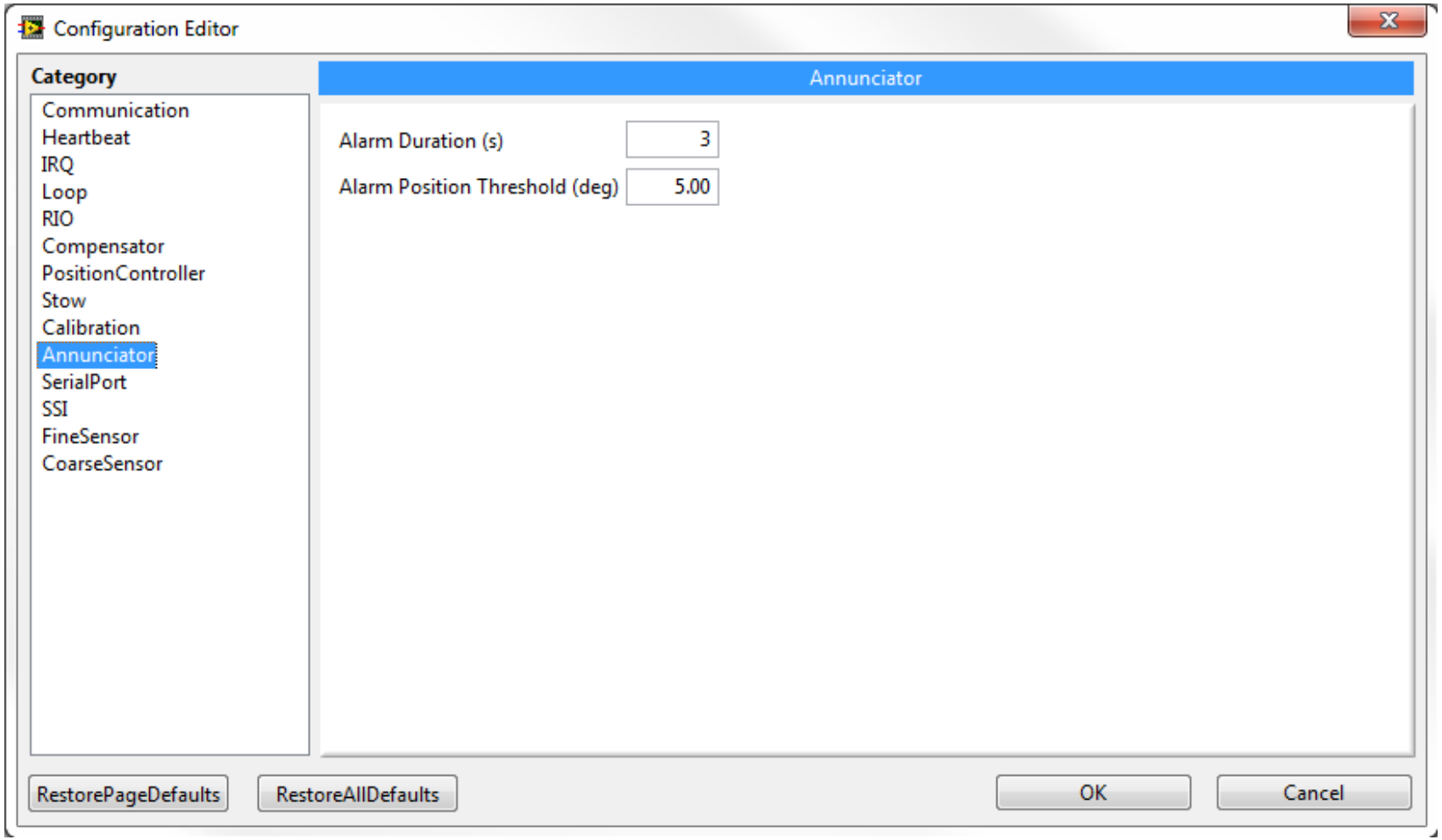
Using a Factory: Call Interface Method



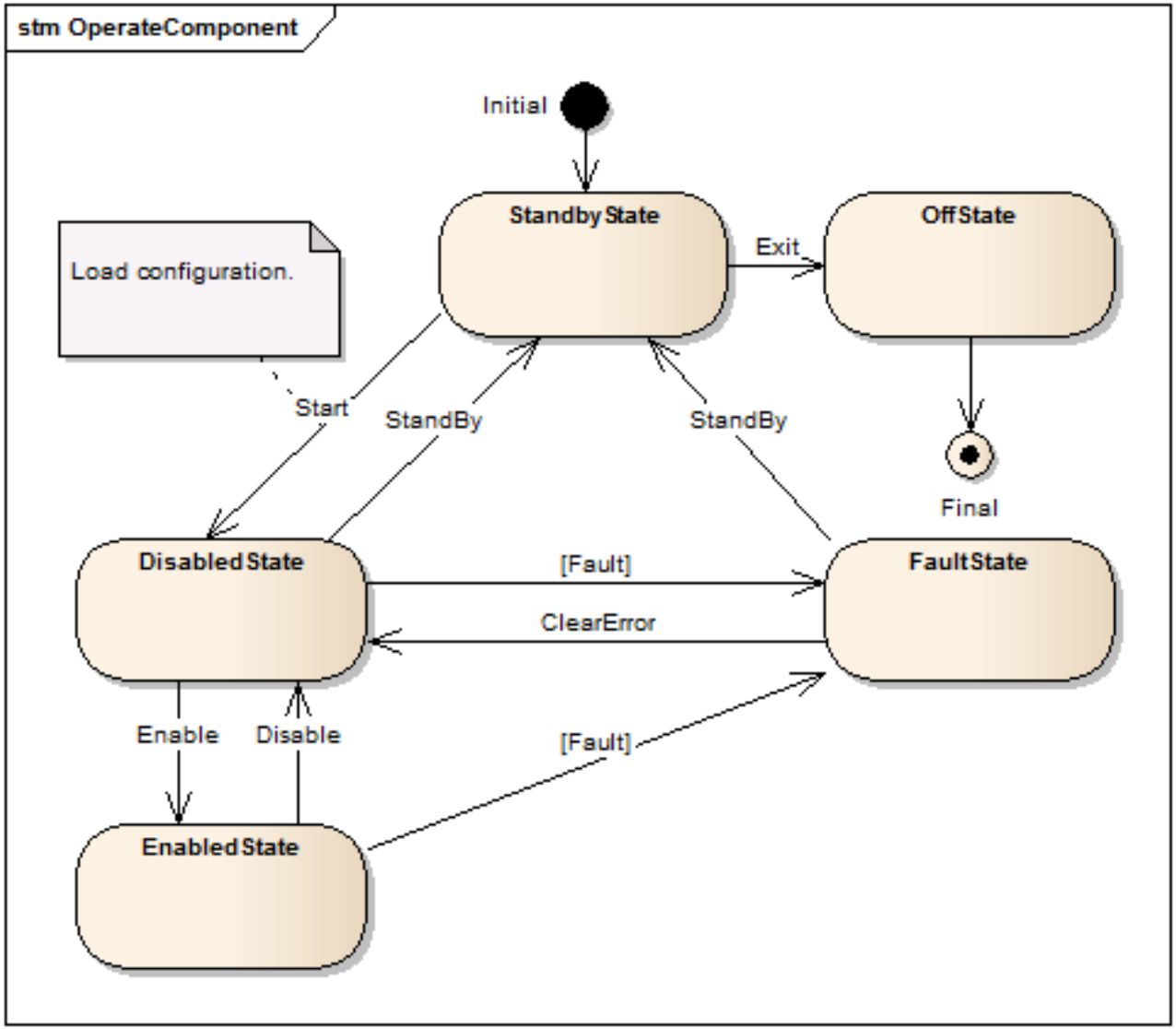
Composite View -- Demonstration



Configuration Editor -- Demonstration



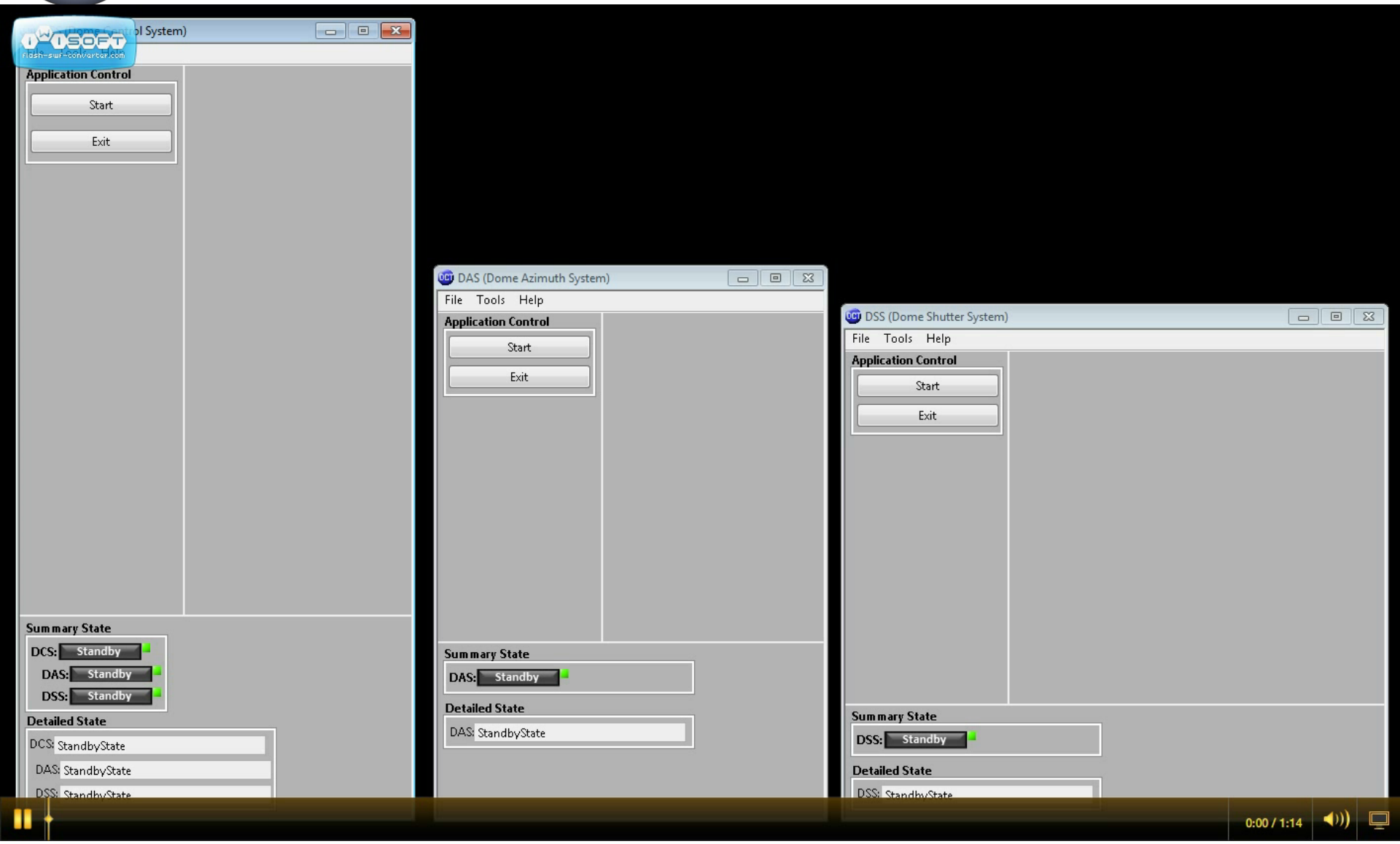
Generic Component State Machine



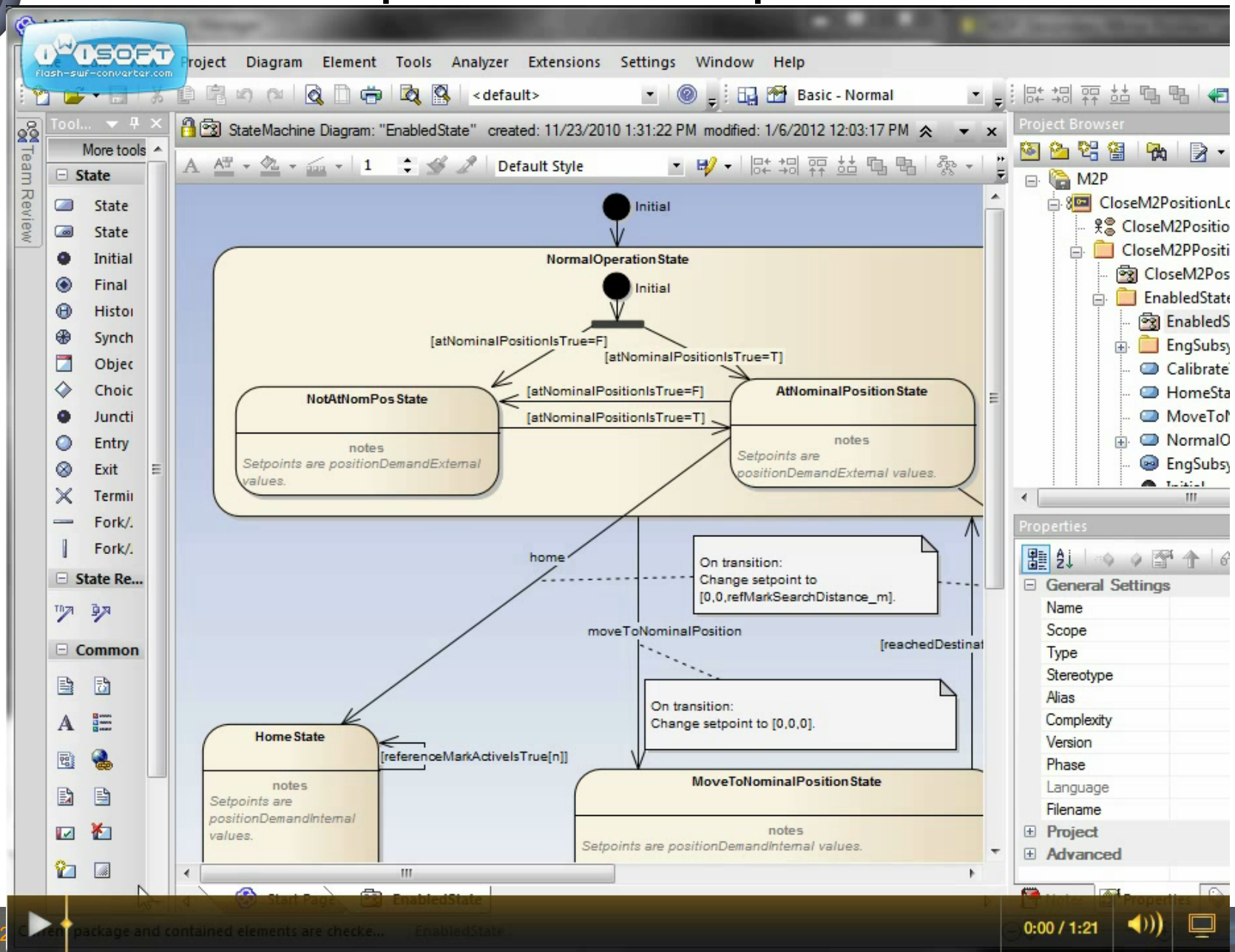
Component Relationships

- Hierarchical
- Relationship rules
  - Knowledge of child
  - Knowledge of parent

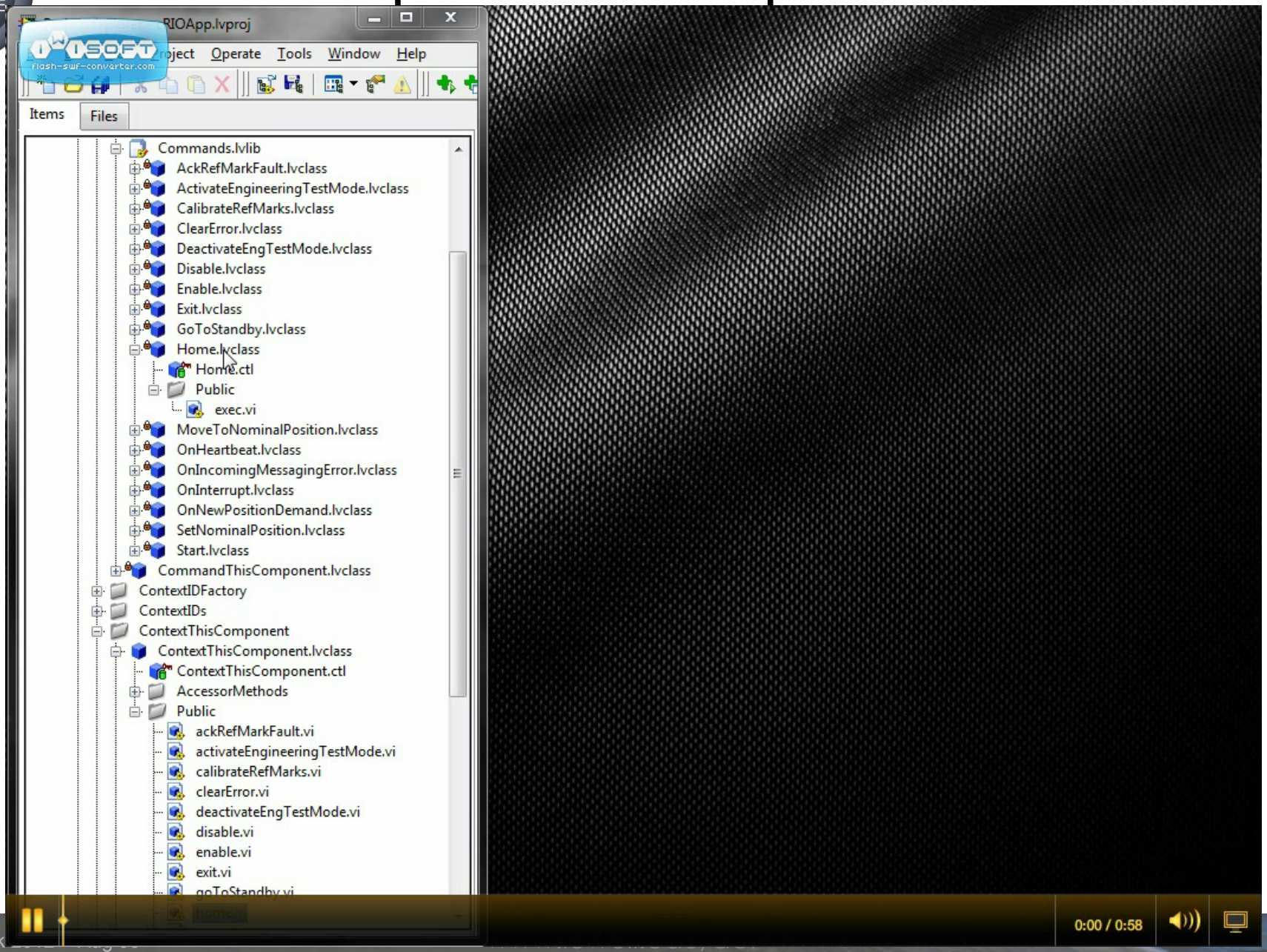
Component Relationships



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

Support Slides

Messaging with Objects

Sender

Receiver

Prepare data → Serialize → Favorite messaging system → Deserialize → Use data

Data to send

Cast to parent, if necessary

Flatten to string

Convert to byte array

Convert to string

Unflatten from string

Original object on the wire!

(or get it that way)

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