NATIONAL INSTRUMENTS

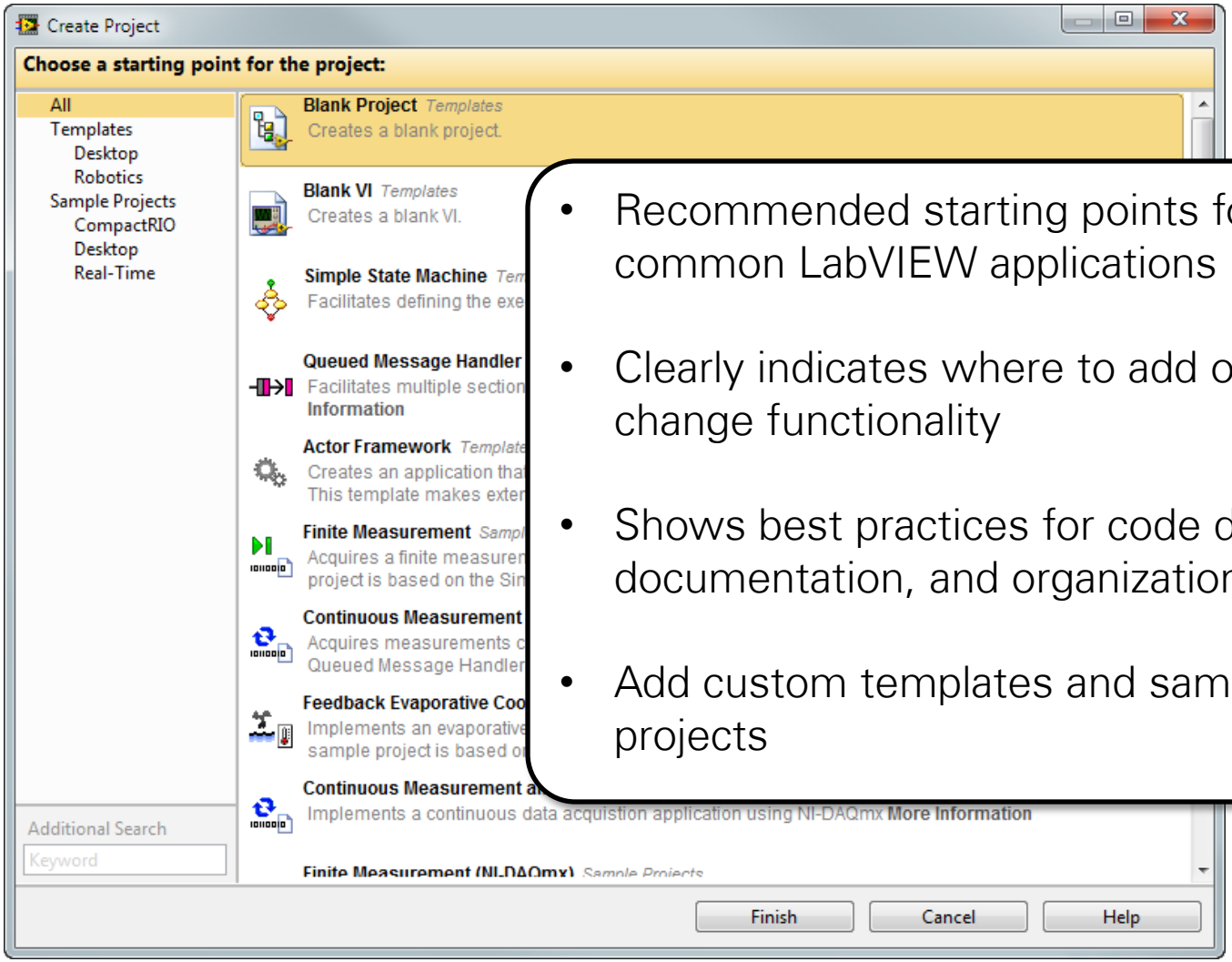
LabVIEW™ 2012

Advanced Design Templates and Sample Projects

Elijah Kerry, Certified LabVIEW Architect (CLA)

Senior Product Manager for LabVIEW, National Instruments

LabVIEW Templates and Sample Projects



- Recommended starting points for common LabVIEW applications
- Clearly indicates where to add or change functionality
- Shows best practices for code design, documentation, and organization
- Add custom templates and sample projects

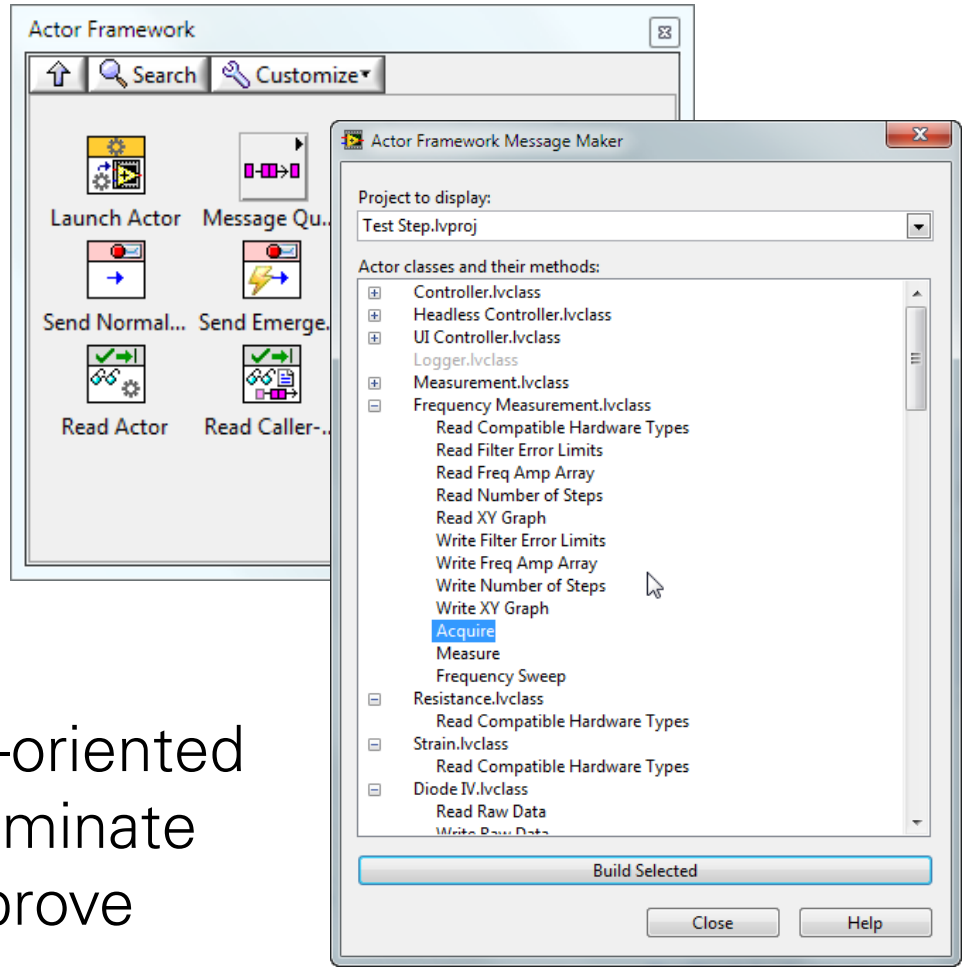
NATIONAL INSTRUMENTS

New Framework for Multi-Process Systems

The Actor Framework designed for large multi-process applications

Includes utility for generating messages and invoking actor methods

For more information:
ni.com/actorframework



Makes heavy use of object-oriented programming in order to eliminate duplication of code and improve system scalability

NATIONAL INSTRUMENTS

Creating a Multi-Process System

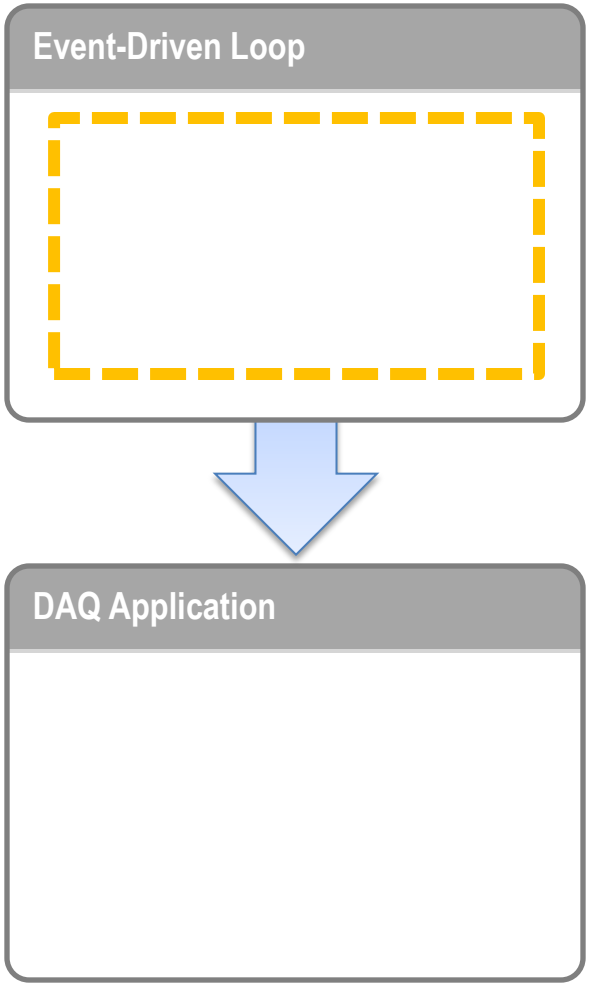
Best Practices

1. Identify data scope
2. Delegate actions to appropriate process
3. Do not poll or use timeouts*

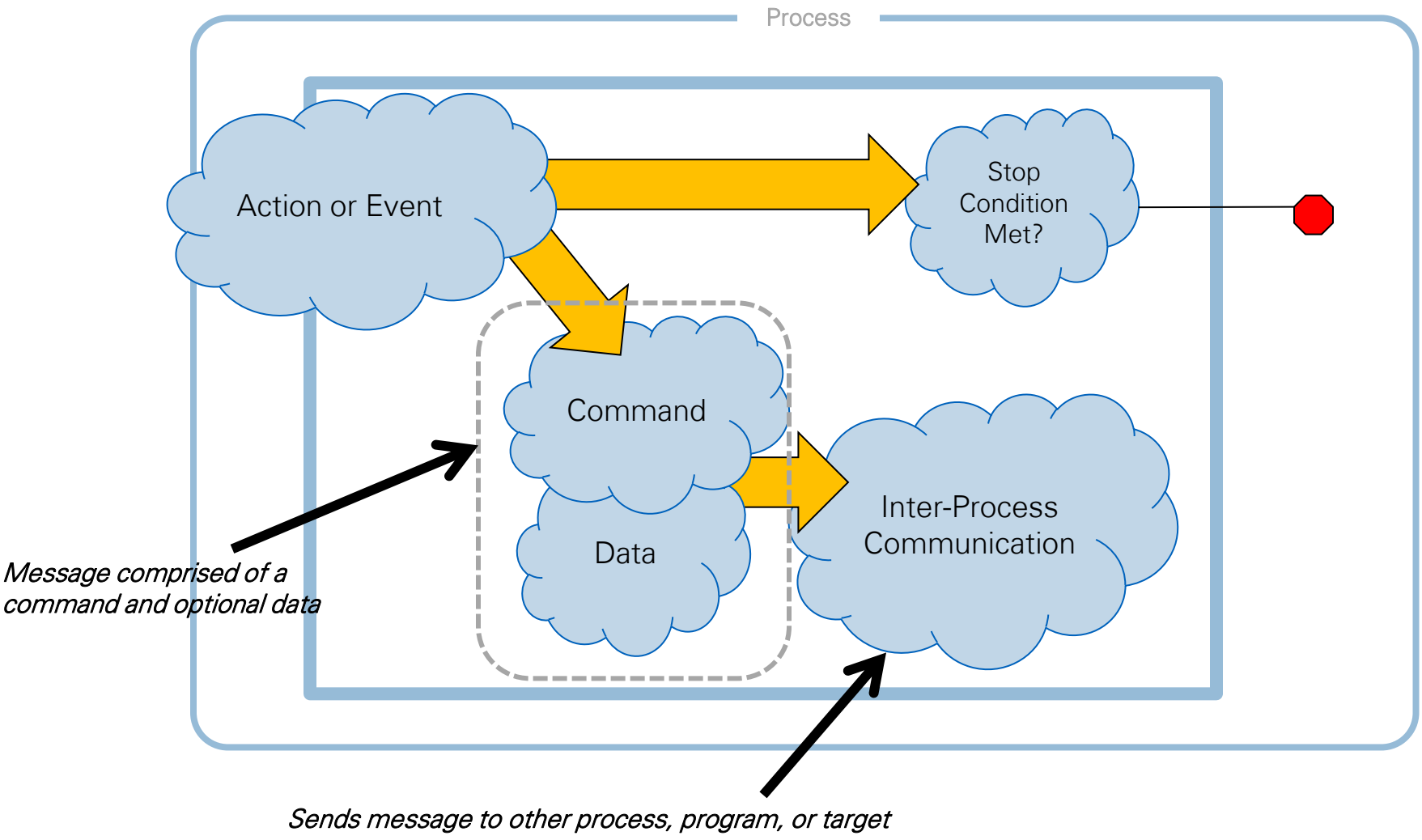
**except for code that communicates with hardware*

Considerations

1. How do you send commands?
2. How do you send data?
3. Which processes can communicate with each other?
4. How do you update the UI?



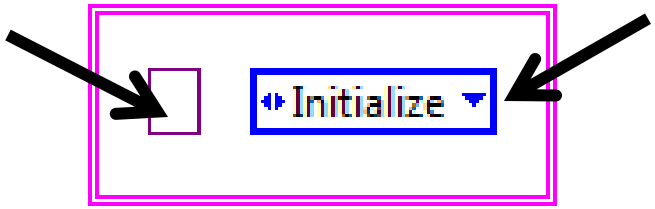
Anatomy of a Message Producer Process



Constructing a Message

Data

Variant allows data-type to vary. Different messages may require different data



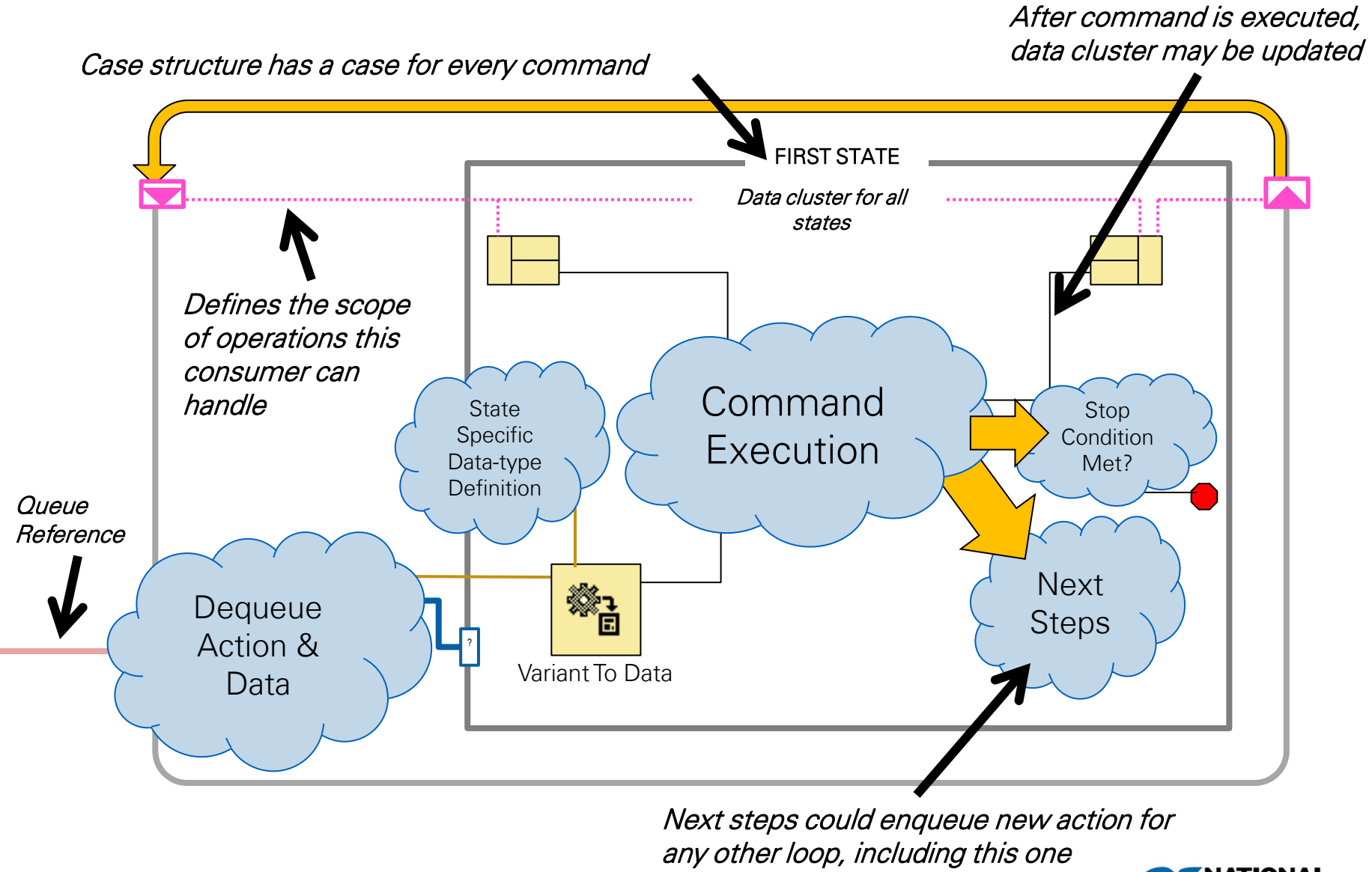
Command

Enumerated constants list all of the options

Examples

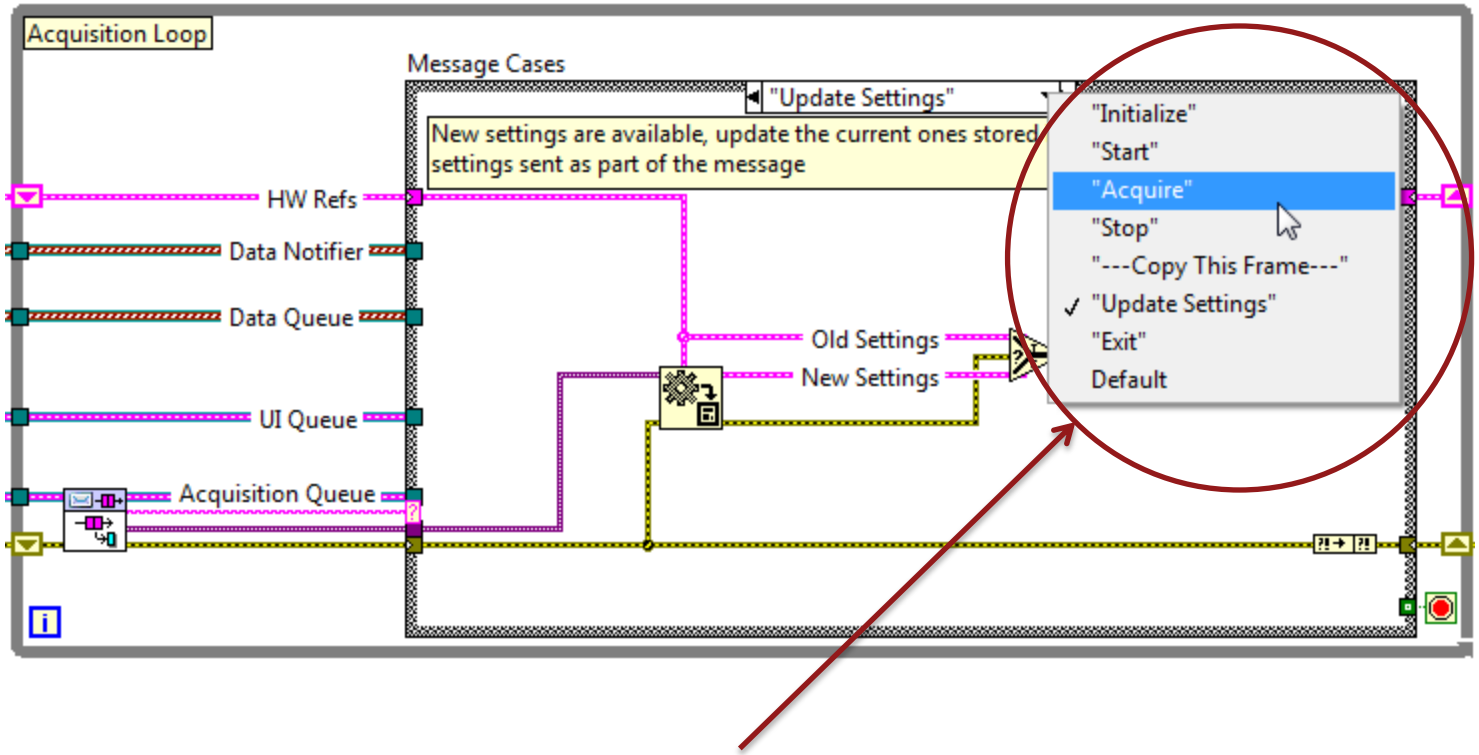
Command	Data
Initialize UI	Cluster containing configuration data
Populate Menu	Array of strings to display in menu
Resize Display	Array of integers [Width, Height]
Load Subpanel	Reference of VI to Load
Insert Header	String
Stop	-

Queued Message Handler Process



NATIONAL INSTRUMENTS

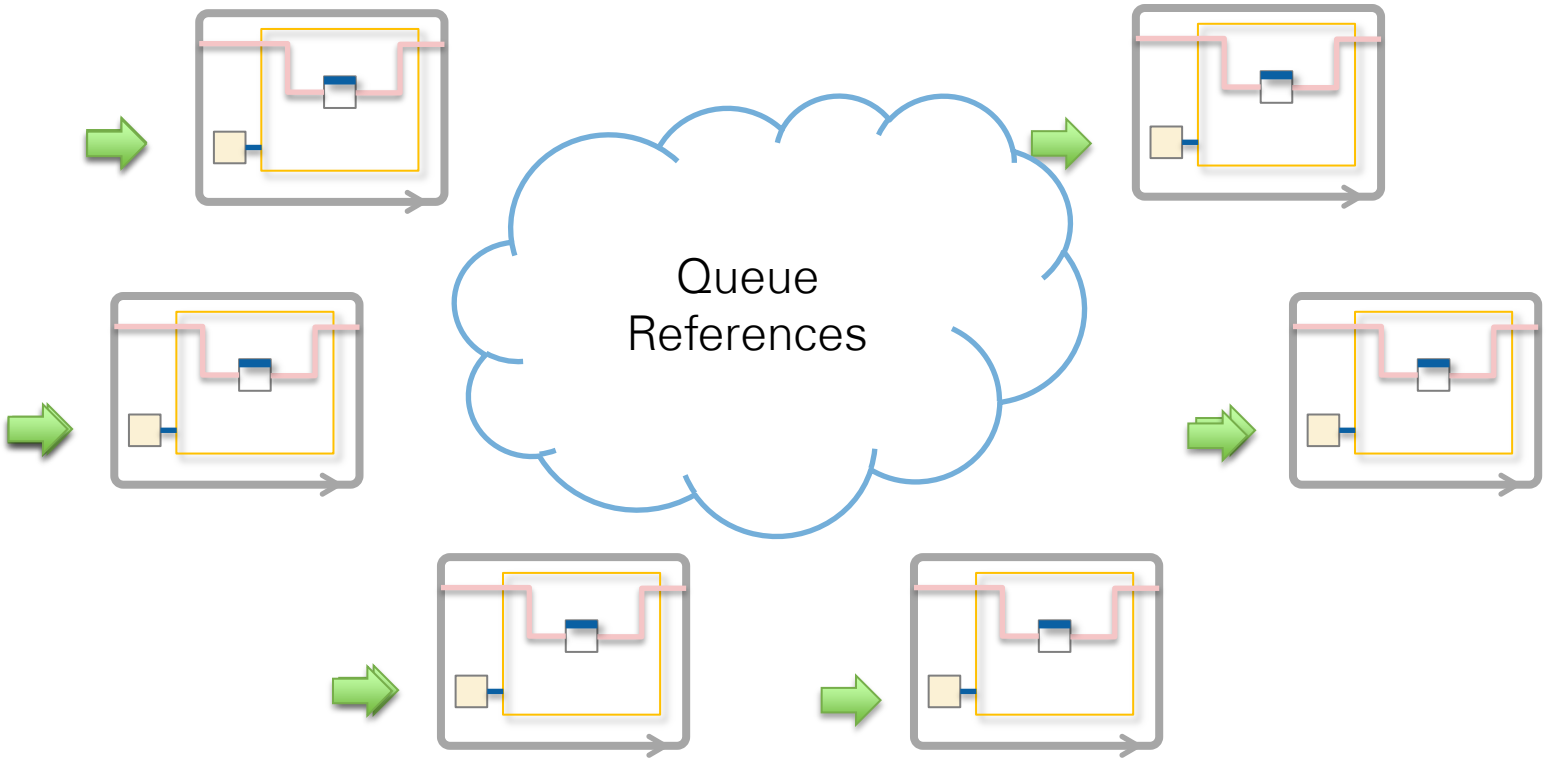
Messages Correspond to the Case Selector



These are the messages this consumer can handle

Scaling Multiple Consumer Systems

Scenario Two: Multiple Instances of Same Process

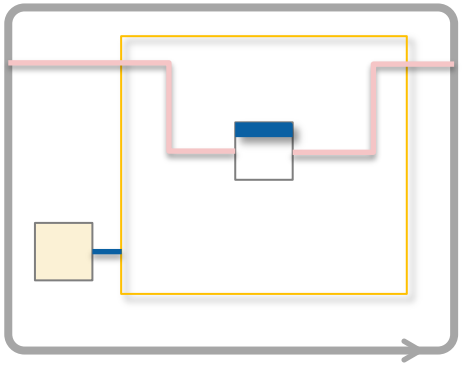


Traditional Solution: Dynamically load VI or duplicate code

Scaling Multiple Consumer Systems

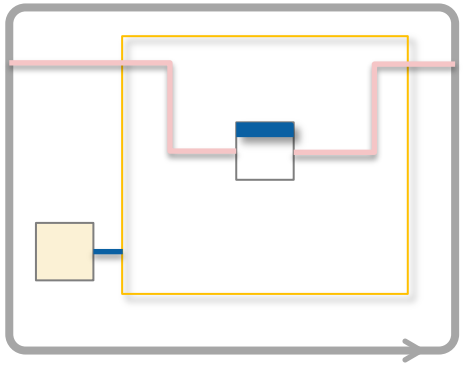
Scenario One: Customize existing behavior

Measurement A



Messages
- Configure
- Acquire
- Measure
- Close
- Exit

Measurement B



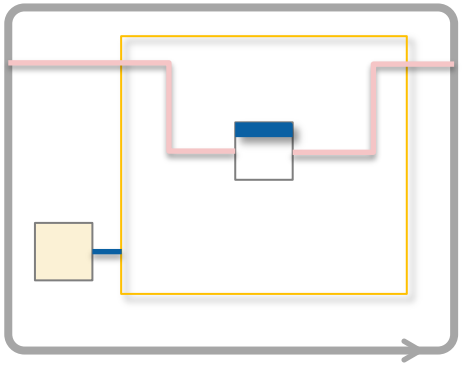
Messages
- Configure
- Acquire
- Measure
- Close
- Exit

Traditional Solution: Maintain two separate copies

Scaling Multiple Consumer Systems

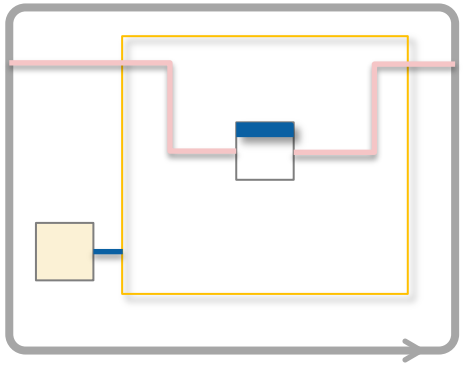
Scenario Two: Add Additional Capability

Measurement A



Messages
- Configure
- Acquire
- Measure
- Close
- Exit

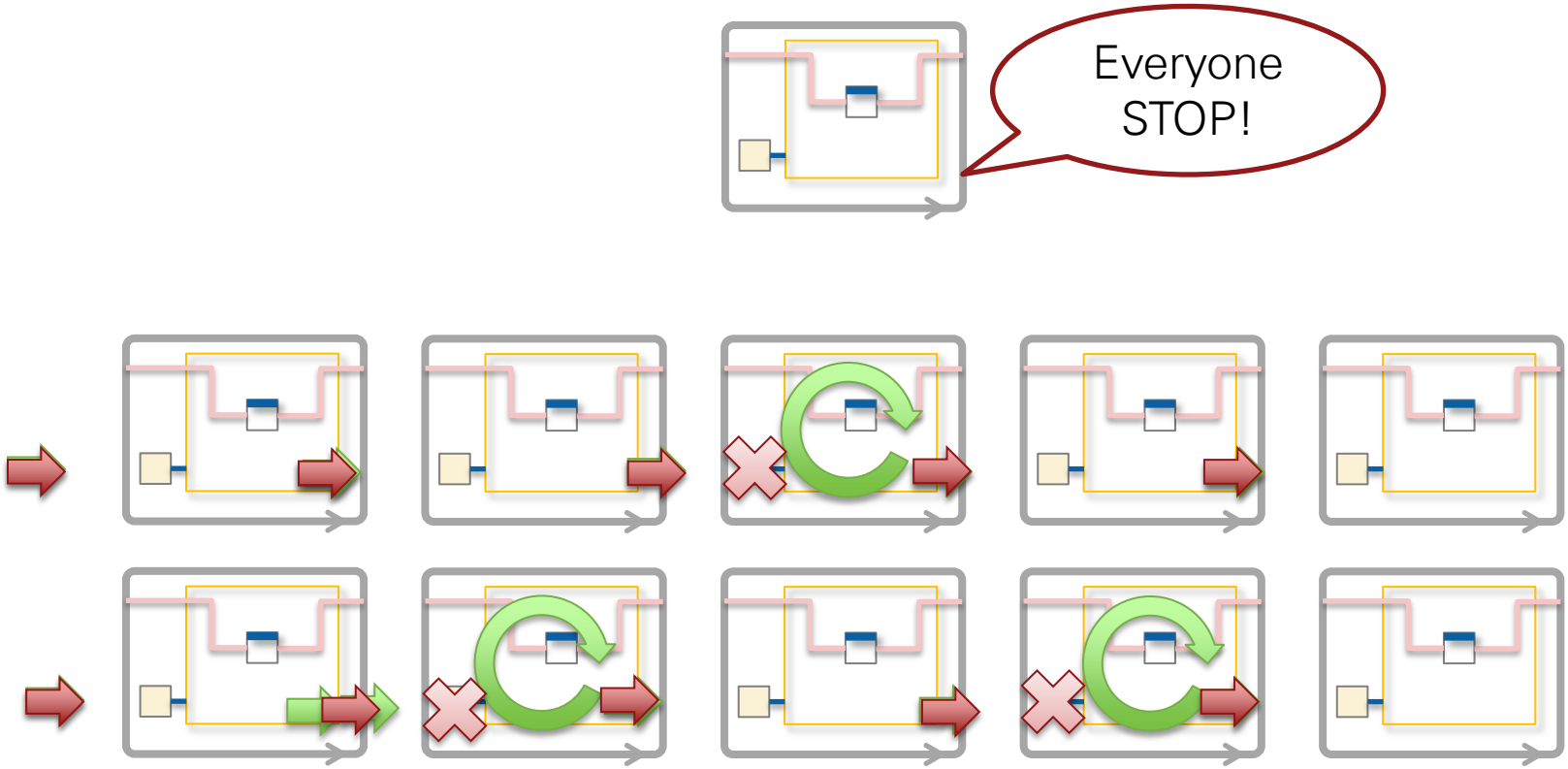
Measurement B



Messages
- Configure
- Acquire
- Log
- Measure
- Close
- Exit

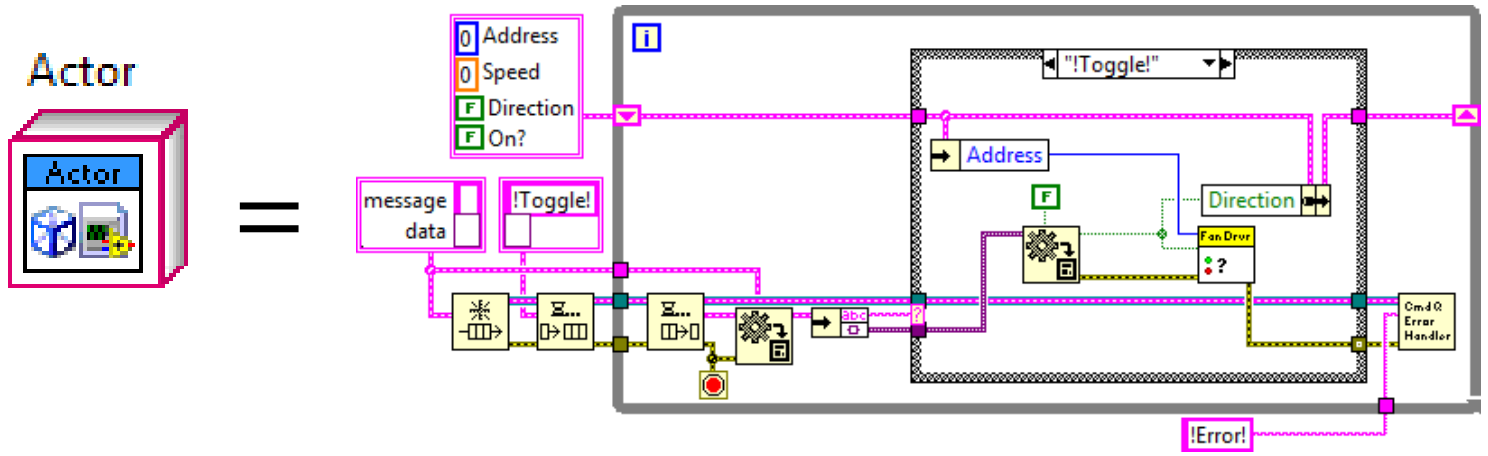
Traditional Solution: Maintain two separate copies

Queues Exist for the Lifetime of the Creator



When the creator of a queue leaves memory, the reference is destroyed.
Any items remaining in the buffer will not be read, including ‘Stop’

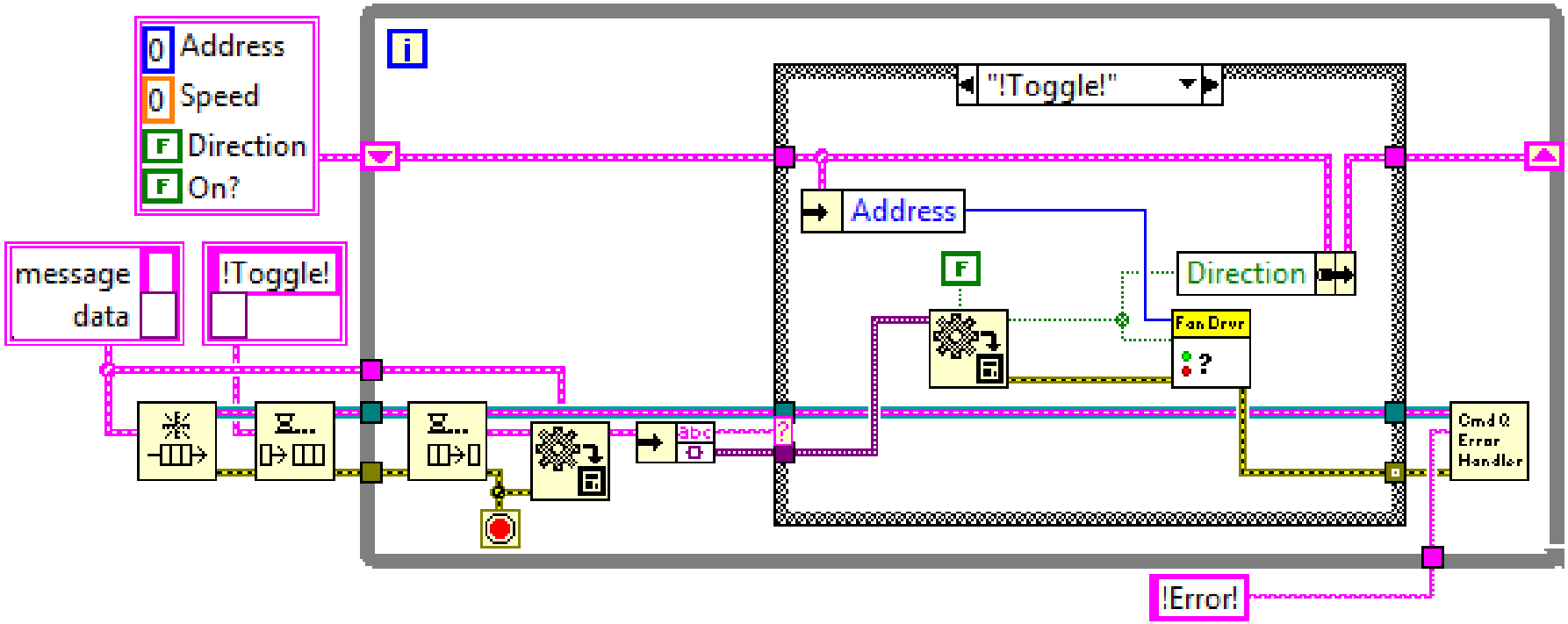
What is an Actor?



An actor is conceptually the same as a queued message handler (QMH) or queued state machine (QSM).

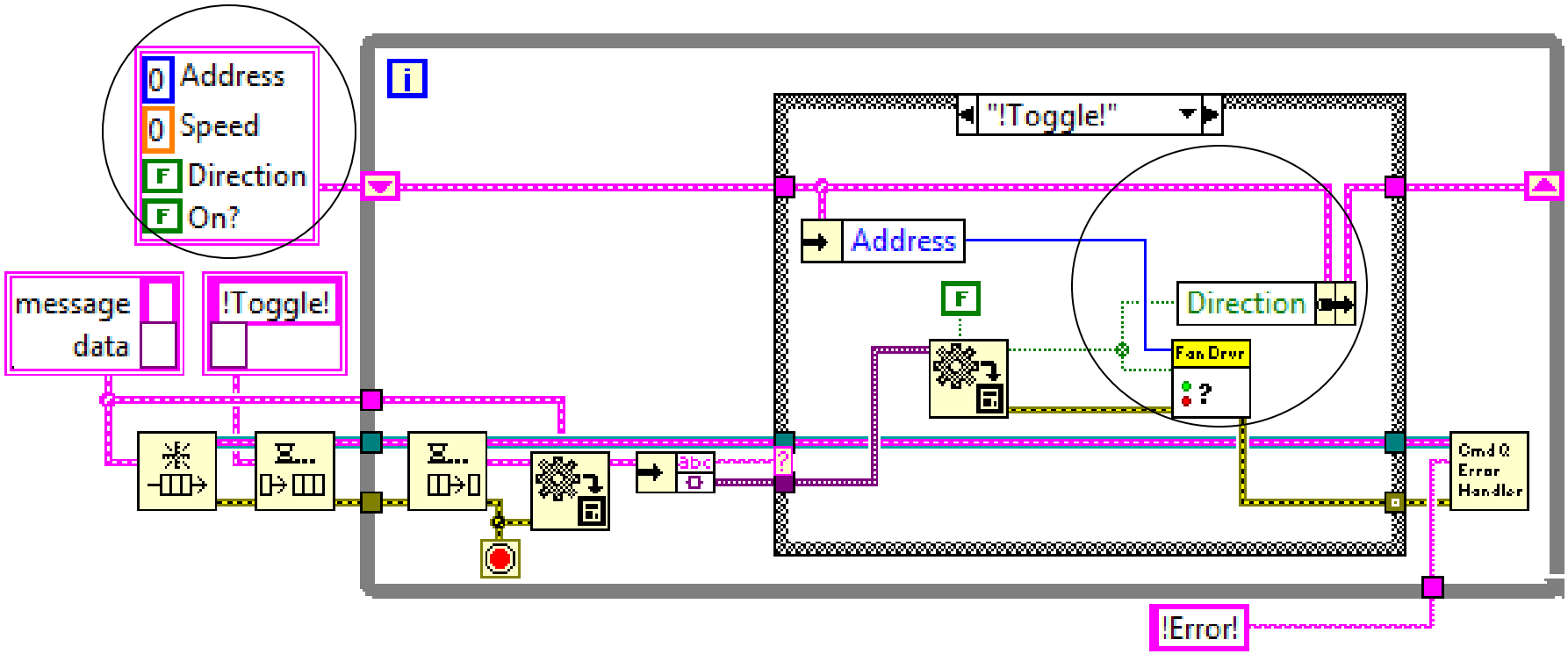
NATIONAL INSTRUMENTS

Queue-Driven State Machine



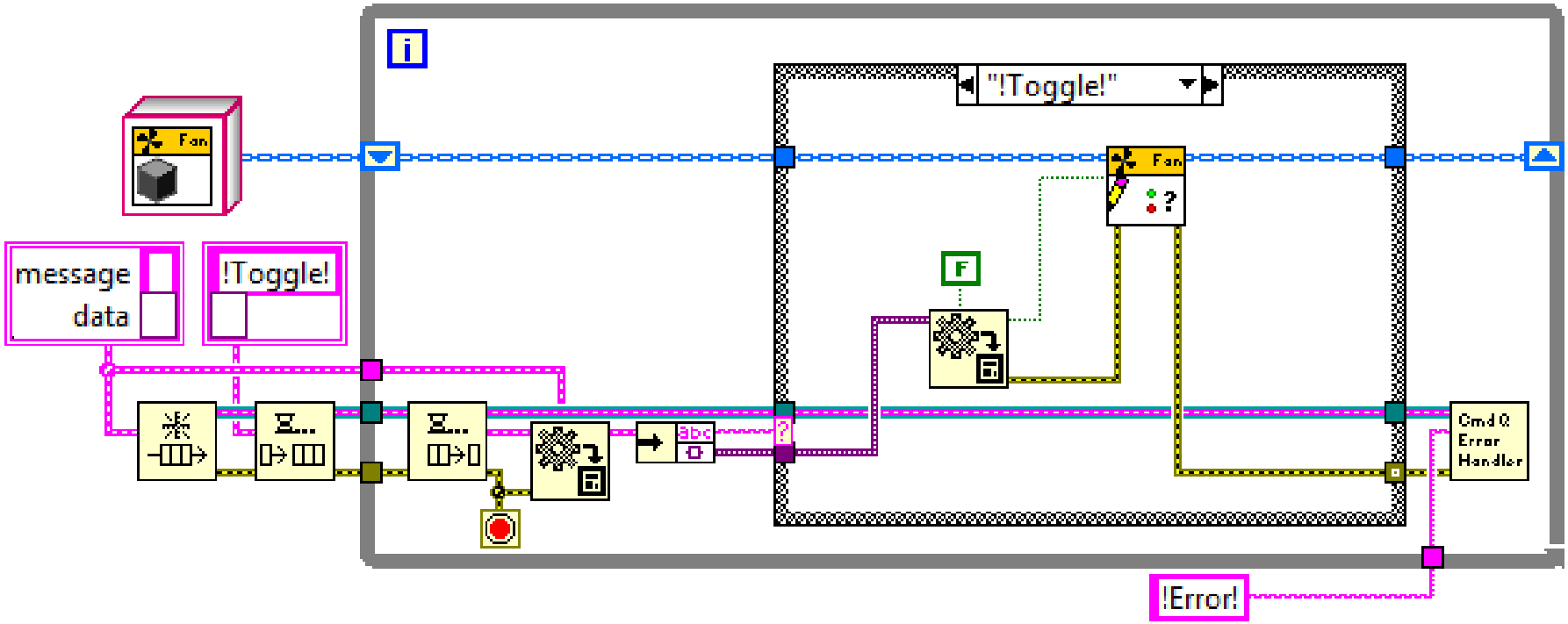
NATIONAL INSTRUMENTS

Cluster and Node...



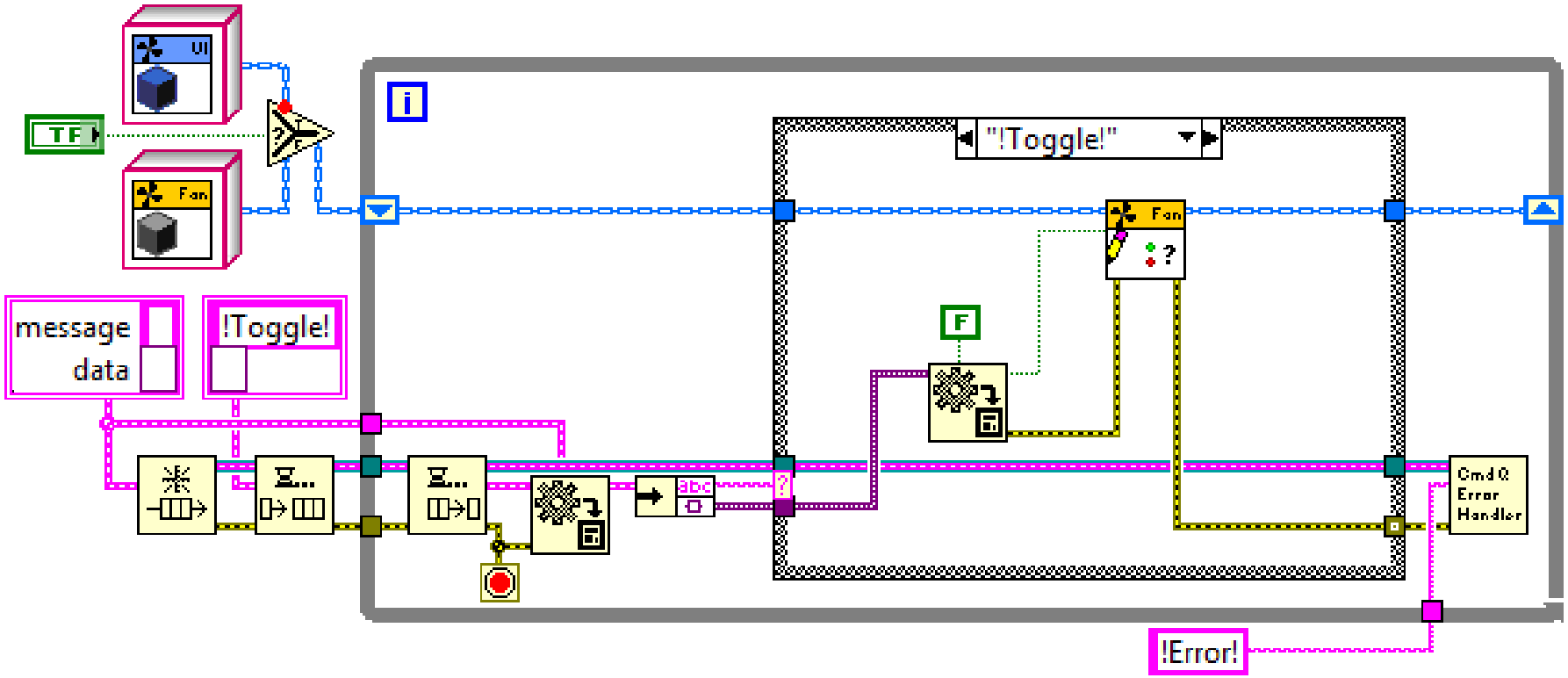
NATIONAL INSTRUMENTS

… become Class and Method



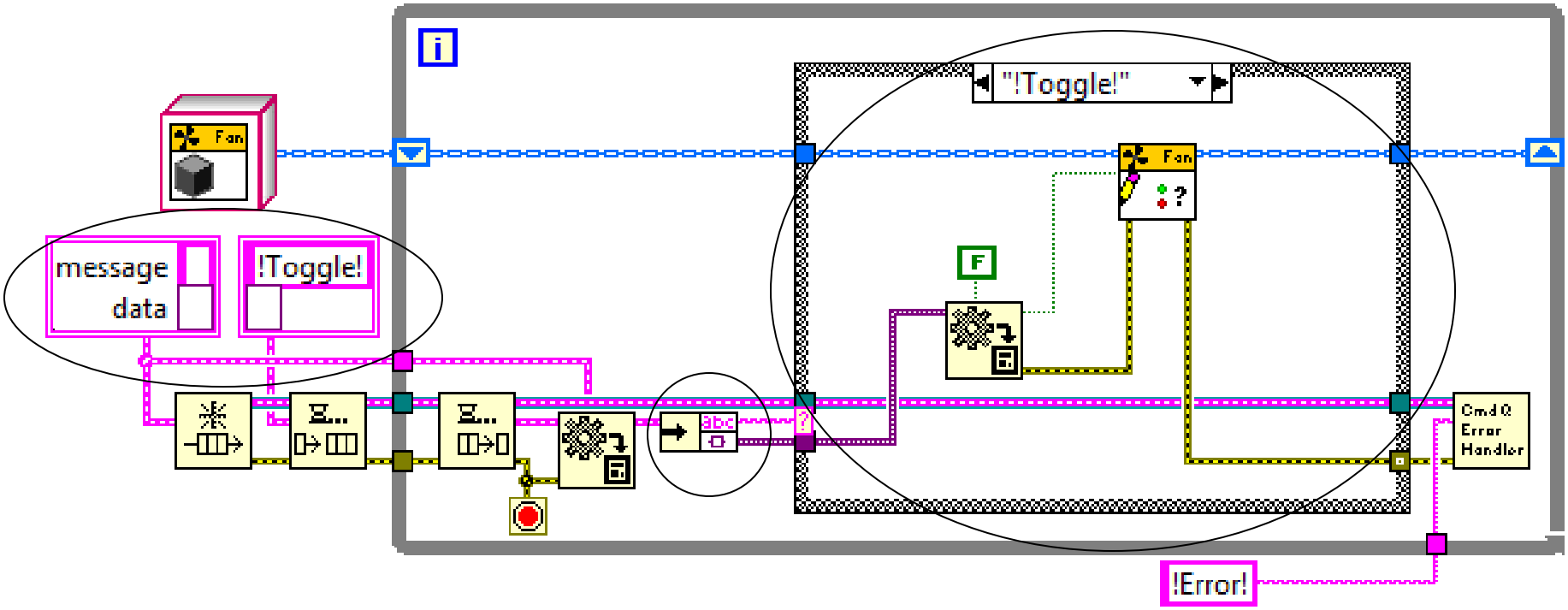
NATIONAL INSTRUMENTS

… become Class and Method



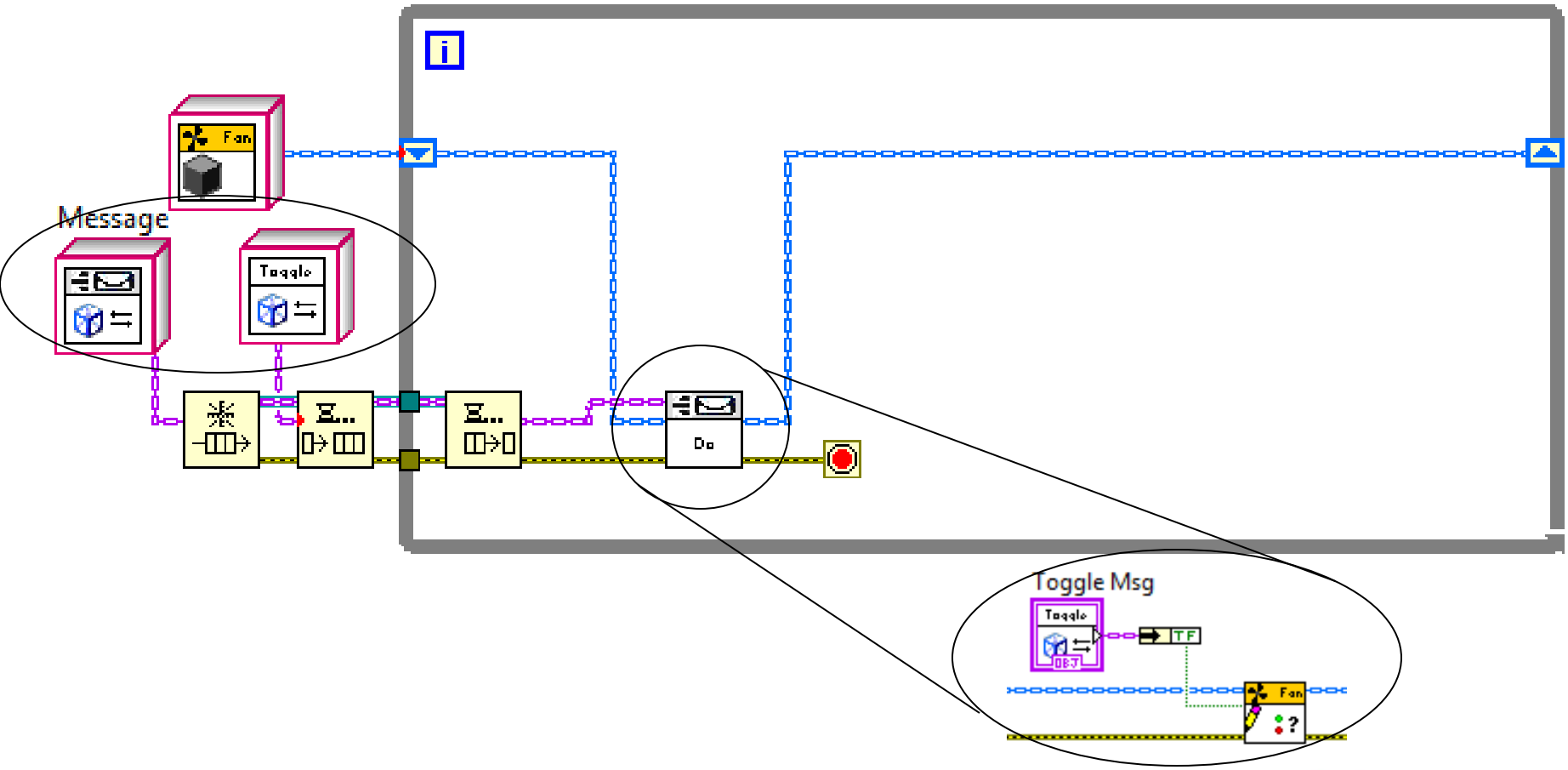
NATIONAL INSTRUMENTS

Message and Case Structure…



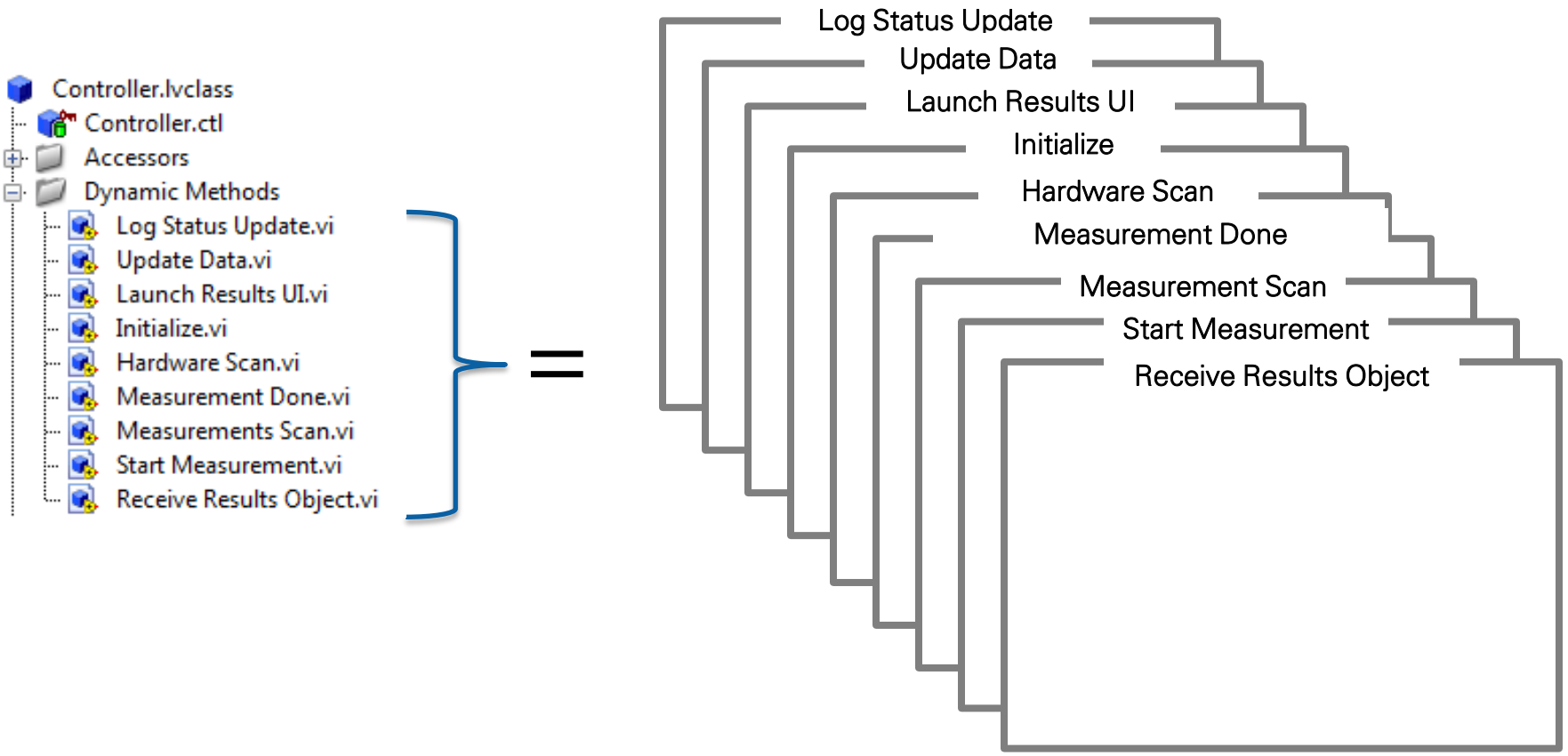
NATIONAL INSTRUMENTS

… become Class and Dynamic Dispatch



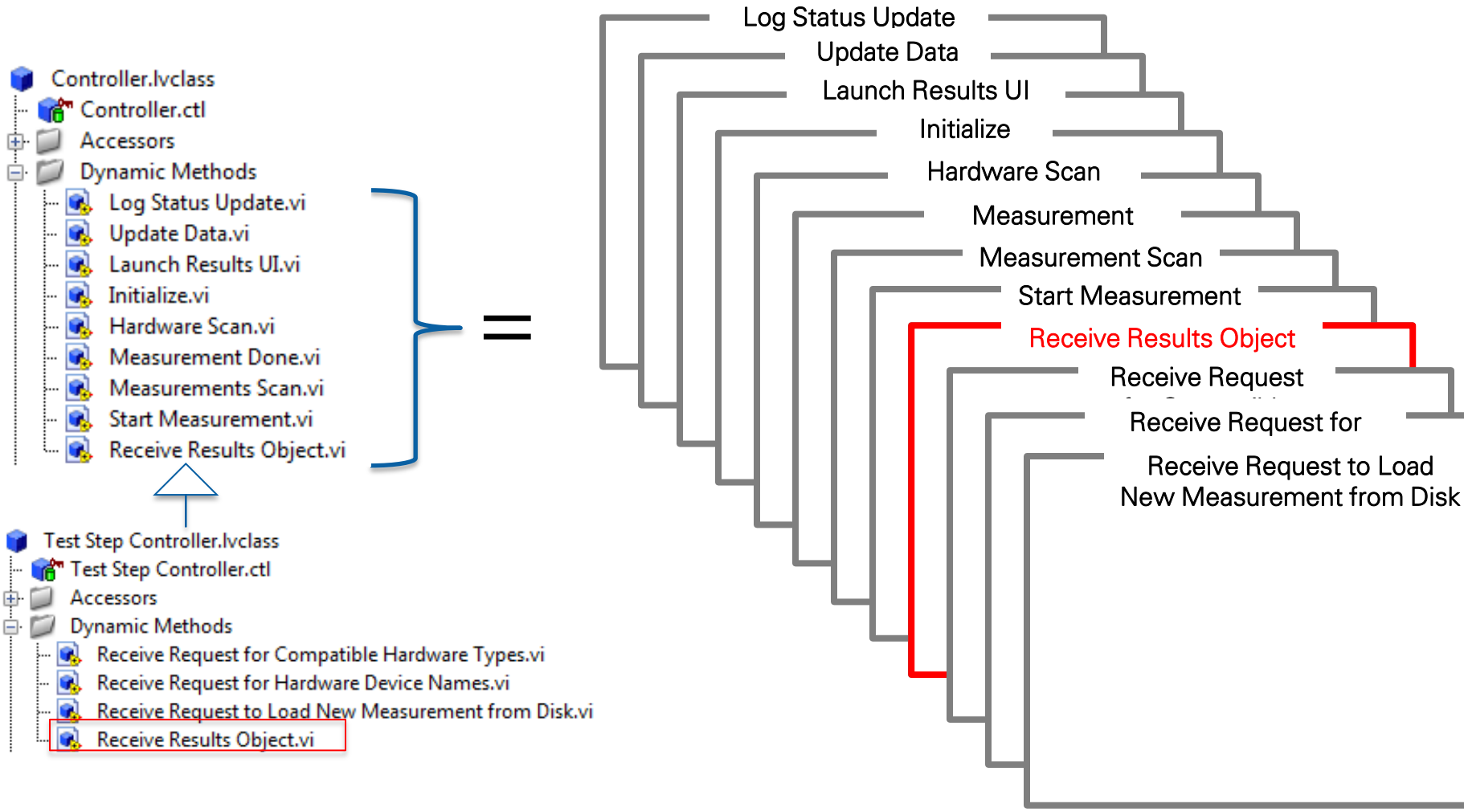
NATIONAL INSTRUMENTS

Methods of an Actor are run when a message is received



Methods of an Actor define the messages it can consume

What are the States of the Actor?



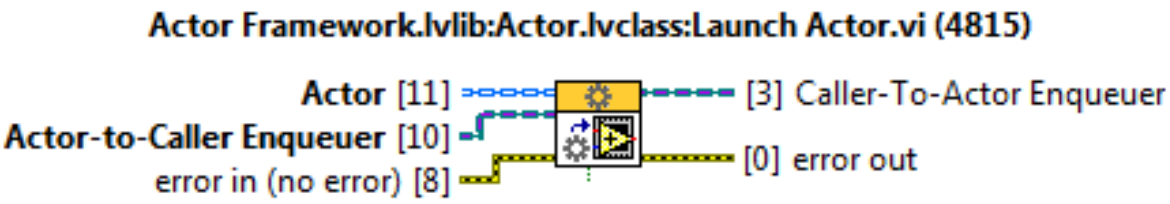
NATIONAL INSTRUMENTS

Launching Actors and Managing Queues

Actor.lvclass has two queues in the private data

- Queue for sending messages to the calling actor
- Queue for receiving messages

These are stored in the actor's private data when launched

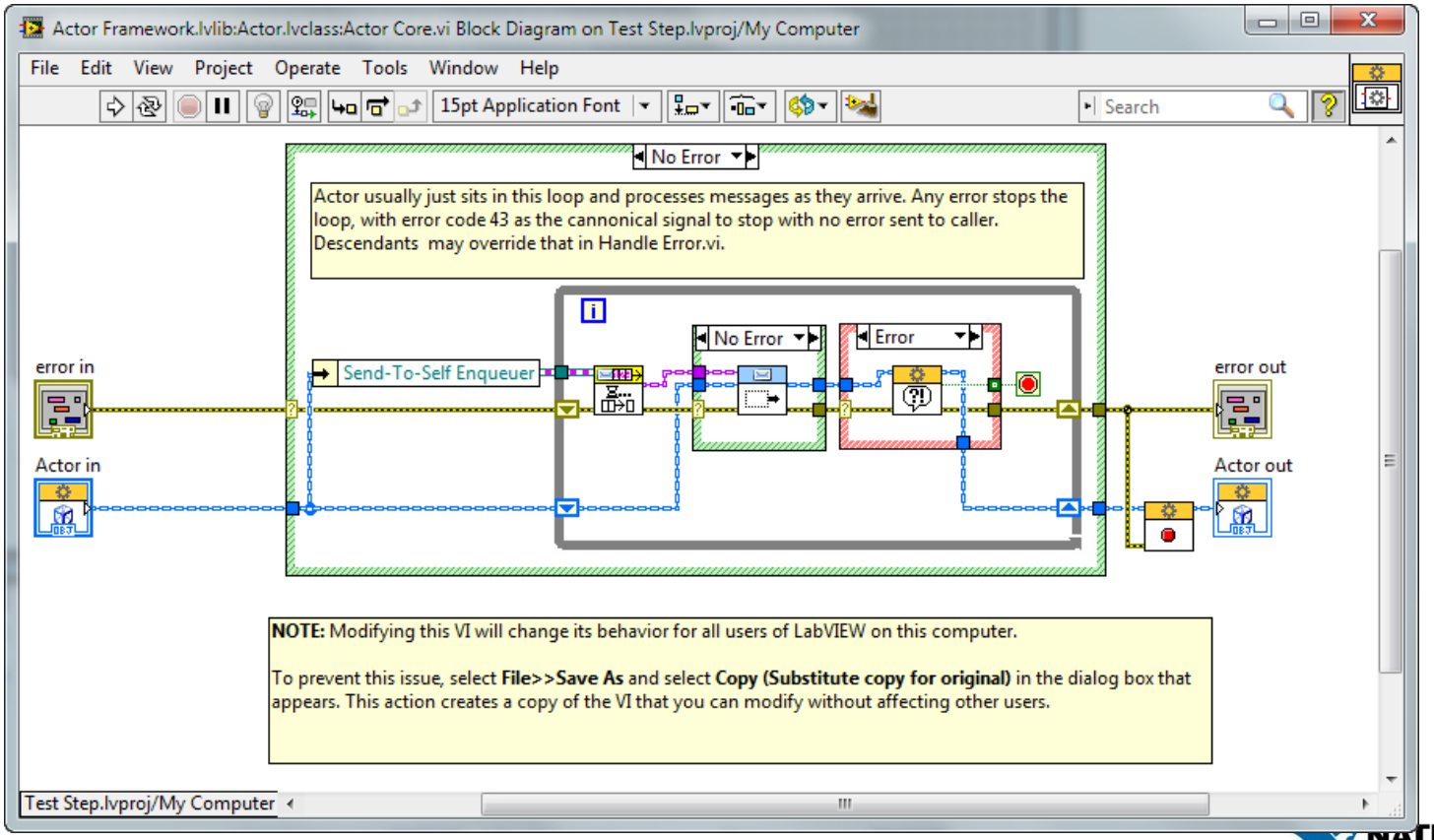


Actors that launch another actor should store the new actor's queue in its own private data

Understanding 'Actor Core.vi'

This is a queued message handler

Launching an Actor starts the 'actor core.vi' method.



NATIONAL INSTRUMENTS

When Do You Need to Override 'Actor Core.vi' ?

To maintain an additional loop within the Actor

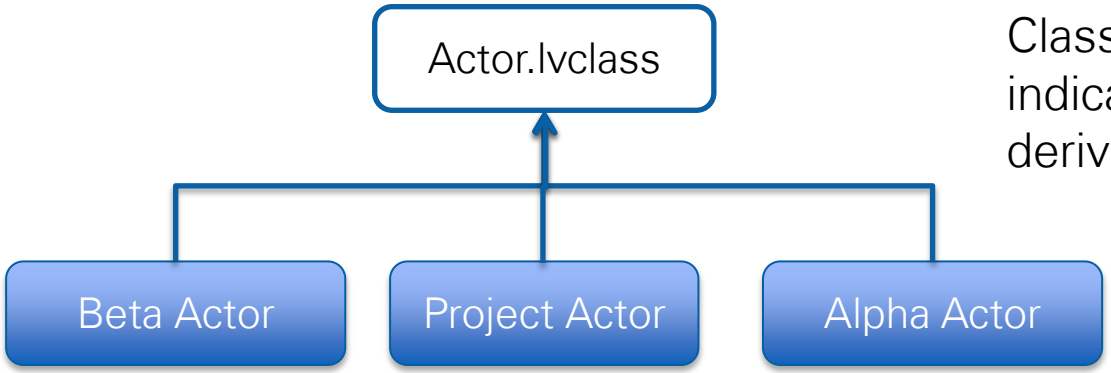
To guarantee certain actions are performed before any messages are handled

To create a unique user interface for a specific Actor

Actor Framework Template Demonstration

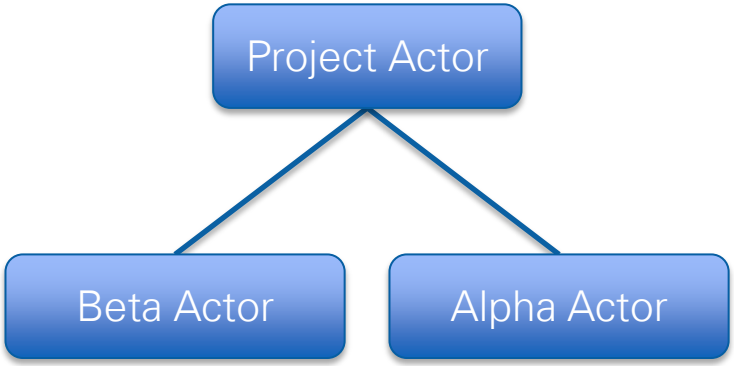
Actor Framework Template Relationships

Class Inheritance Hierarchy

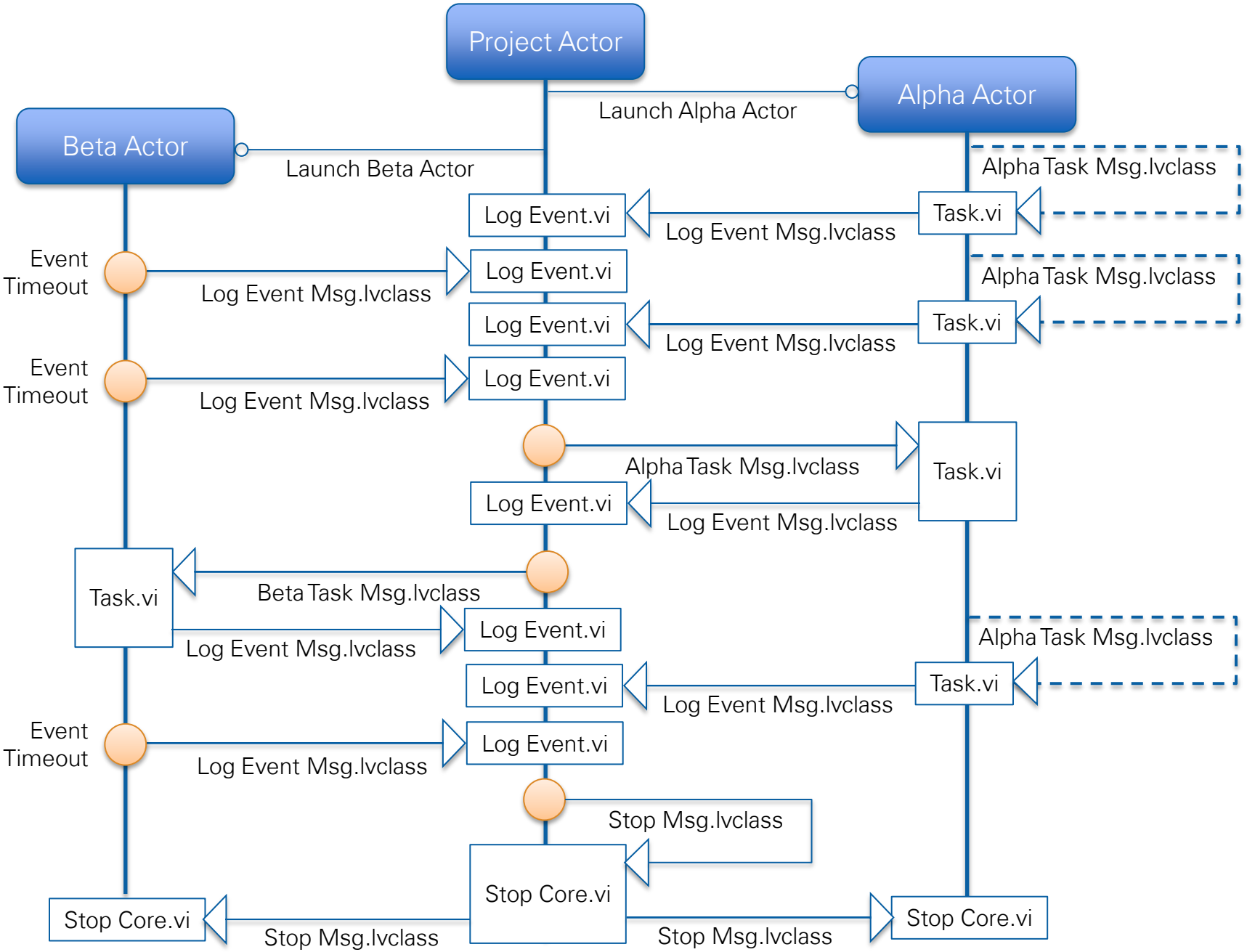


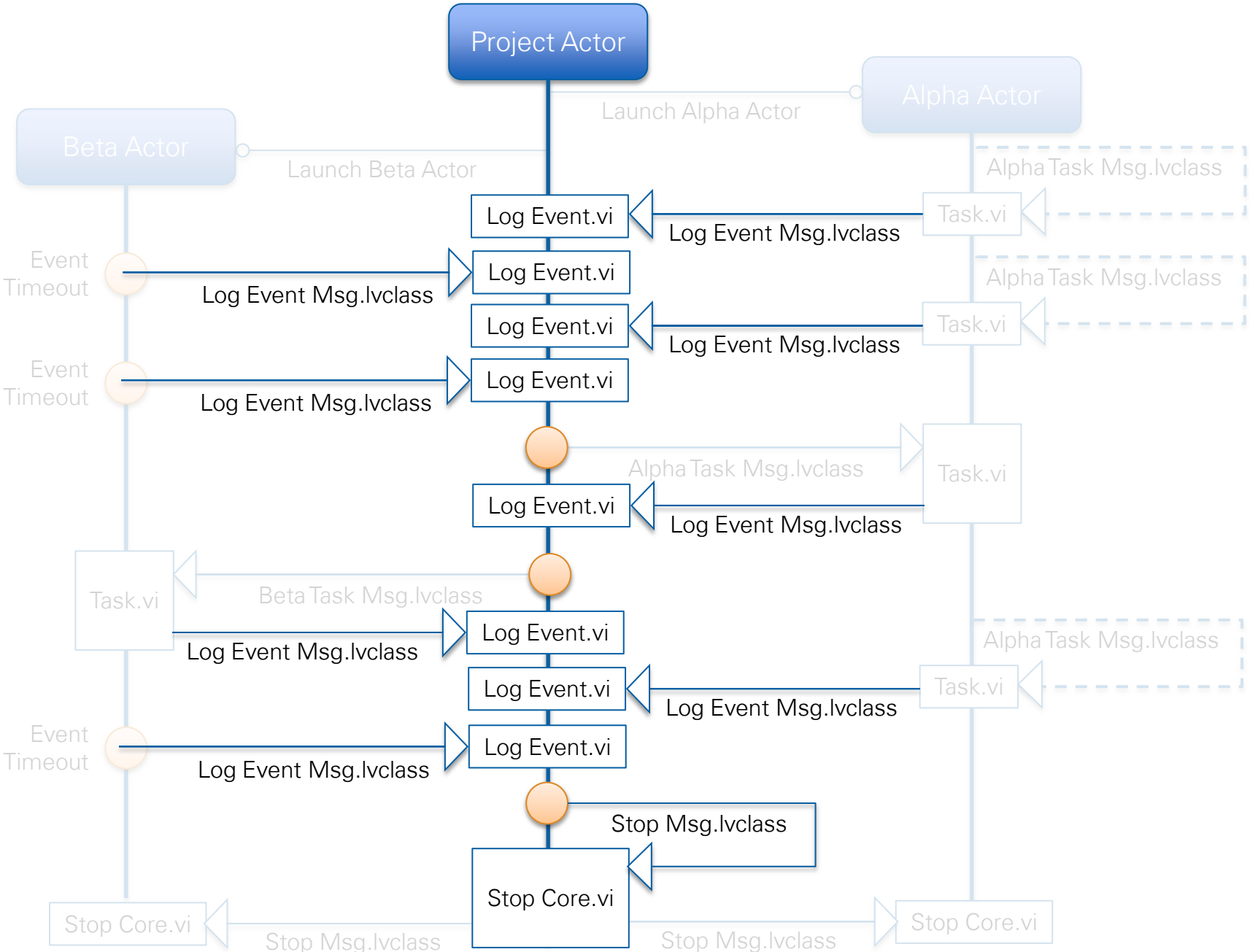
Class Inheritance Hierarchy indicates that all three are derived from Actor.lvclass

Task Tree



Task Tree Indicates that the Project Actor launches Alpha and Beta Actors

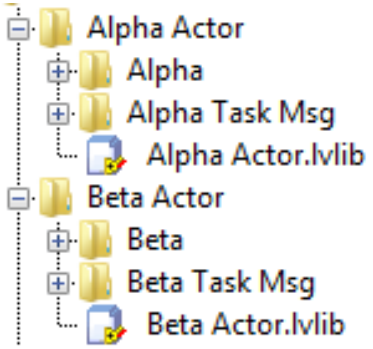

Project Actor
Alpha Actor
Beta Actor
Launch Alpha Actor
Launch Beta Actor
Alpha Task Msg.lvclass
Log Event.vi
Task.vi
Log Event Msg.lvclass
Event Timeout
Log Event.vi
Alpha Task Msg.lvclass
Log Event.vi
Task.vi
Log Event Msg.lvclass
Event Timeout
Log Event Msg.lvclass
Log Event.vi
Alpha Task Msg.lvclass
Task.vi
Log Event.vi
Log Event Msg.lvclass
Task.vi
Beta Task Msg.lvclass
Log Event.vi
Log Event Msg.lvclass
Alpha Task Msg.lvclass
Log Event.vi
Task.vi
Log Event Msg.lvclass
Event Timeout
Log Event.vi
Log Event Msg.lvclass
Stop Msg.lvclass
Stop Core.vi
Stop Core.vi
Stop Msg.lvclass
Stop Msg.lvclass
Stop Core.vi


Project Actor
Alpha Actor
Launch Alpha Actor
Beta Actor
Launch Beta Actor
Alpha Task Msg.lvclass
Log Event.vi
Task.vi
Log Event Msg.lvclass
Event Timeout
Log Event.vi
Alpha Task Msg.lvclass
Log Event.vi
Task.vi
Log Event Msg.lvclass
Event Timeout
Log Event Msg.lvclass
Log Event.vi
Alpha Task Msg.lvclass
Task.vi
Log Event.vi
Log Event Msg.lvclass
Task.vi
Beta Task Msg.lvclass
Log Event.vi
Log Event Msg.lvclass
Alpha Task Msg.lvclass
Log Event.vi
Task.vi
Log Event Msg.lvclass
Event Timeout
Log Event Msg.lvclass
Log Event.vi
Stop Msg.lvclass
Stop Core.vi
Stop Core.vi
Stop Msg.lvclass
Stop Msg.lvclass
Stop Core.vi

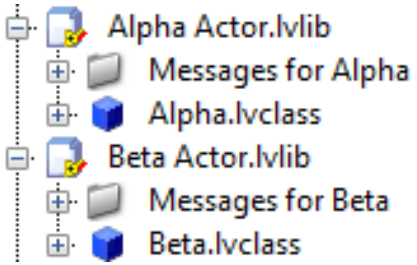
Relationship between Messages and Actors

- Messages call a method of an Actor
- Messages encapsulate data for that method in the private data control
- Messages are therefore coupled to that Actor, but not the children of that Actor
- Children may have unique messages, but they are then coupled to those messages

File View



Items Tree

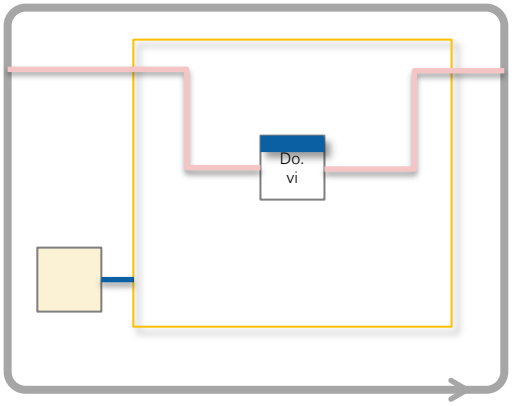


When Do you Need to Override Actor Core.vi ?

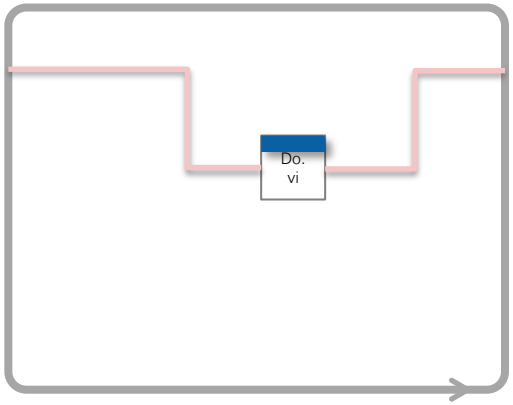
To maintain an additional loop within the Actor

To guarantee certain actions are performed before any messages are handled

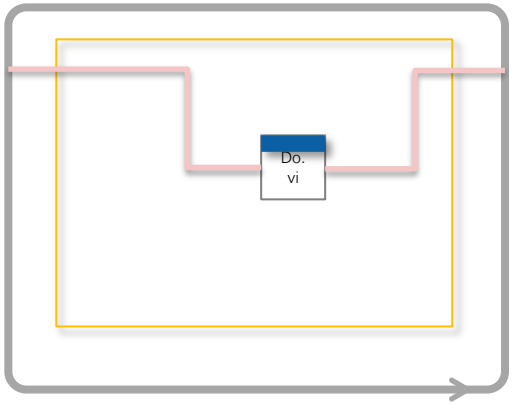
To create a unique user interface for a specific Actor



Message Handler

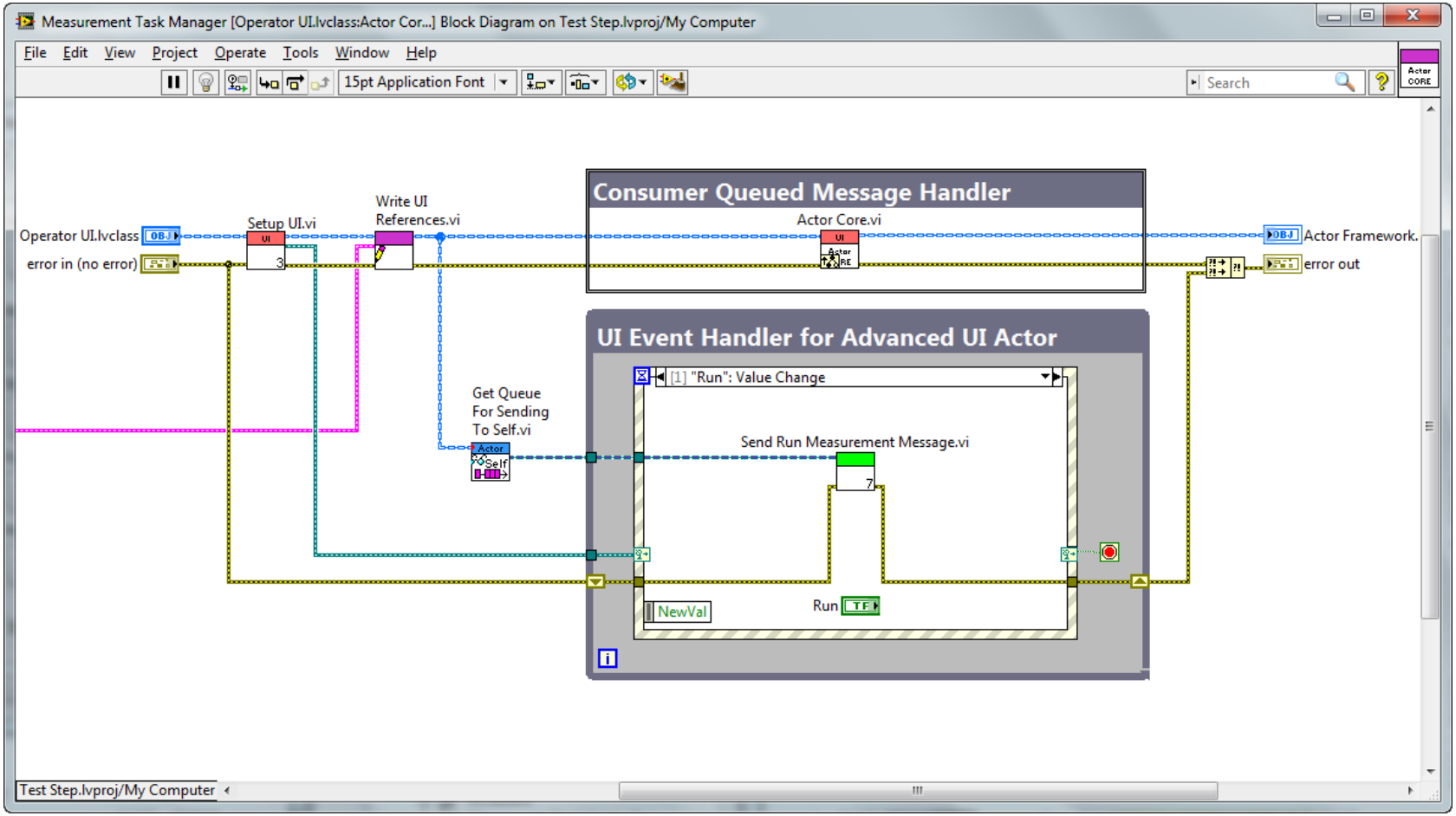


Free Running



Event Driven

Actor core.vi of Operator UI.lvclass
Operator User Interface Block Diagram

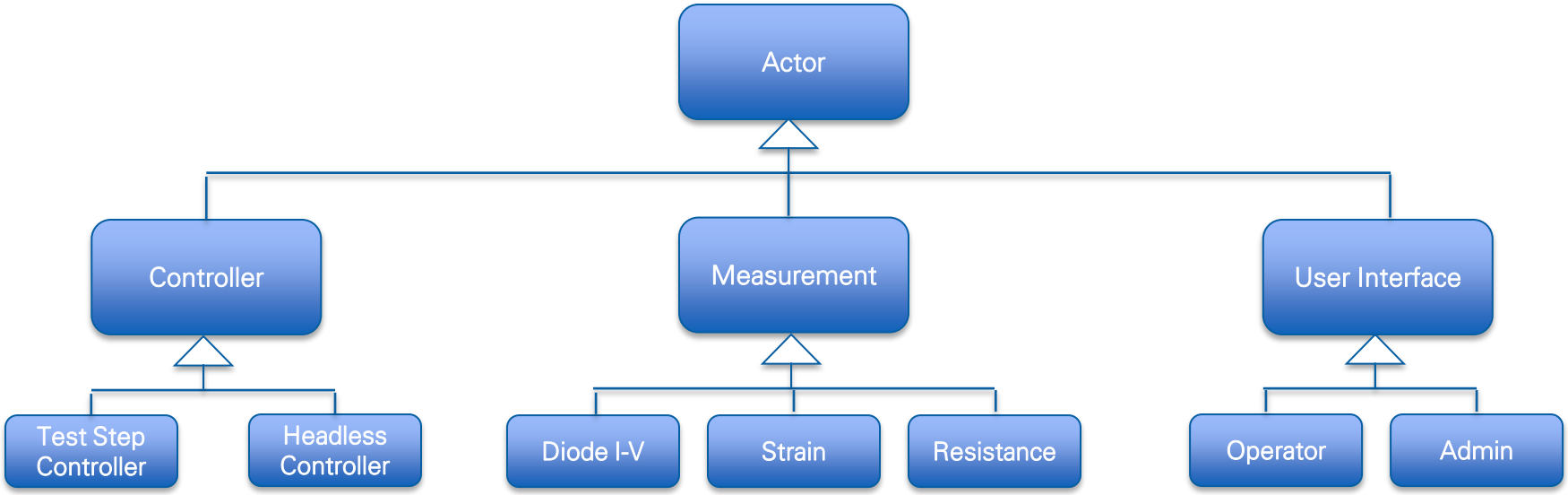


NATIONAL INSTRUMENTS

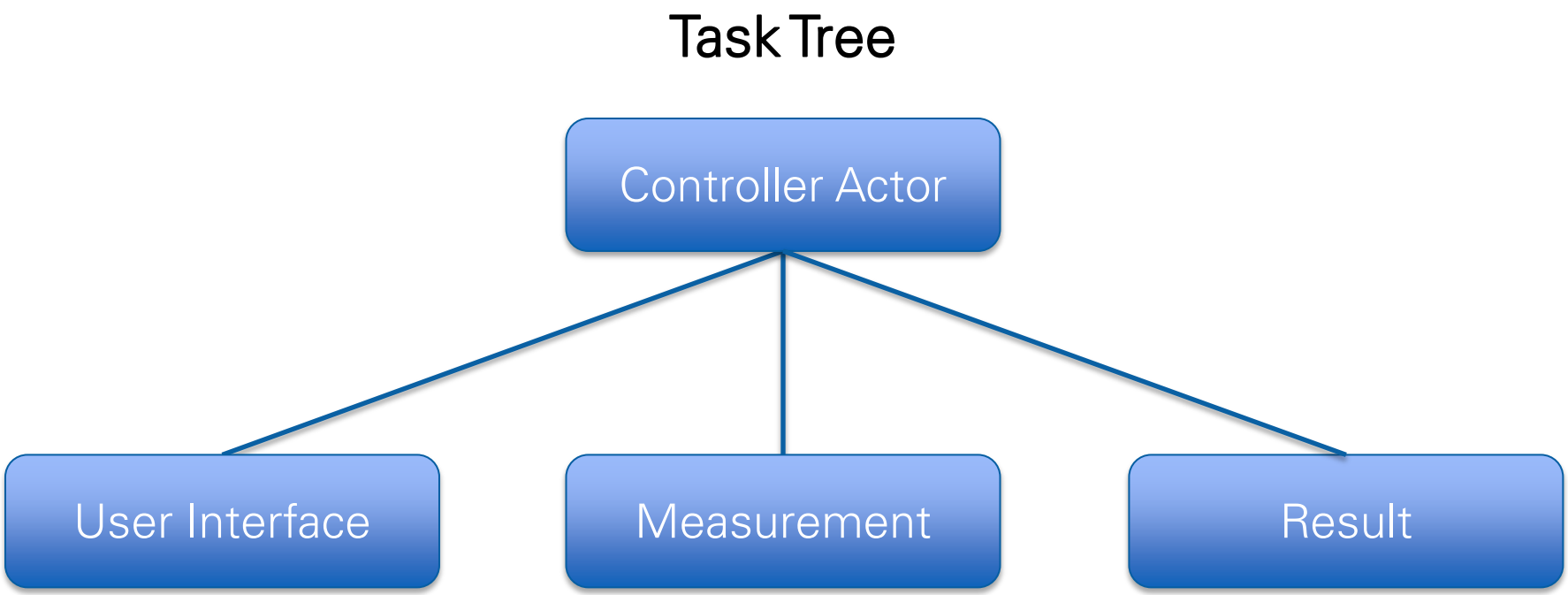
Example Actor Framework Application

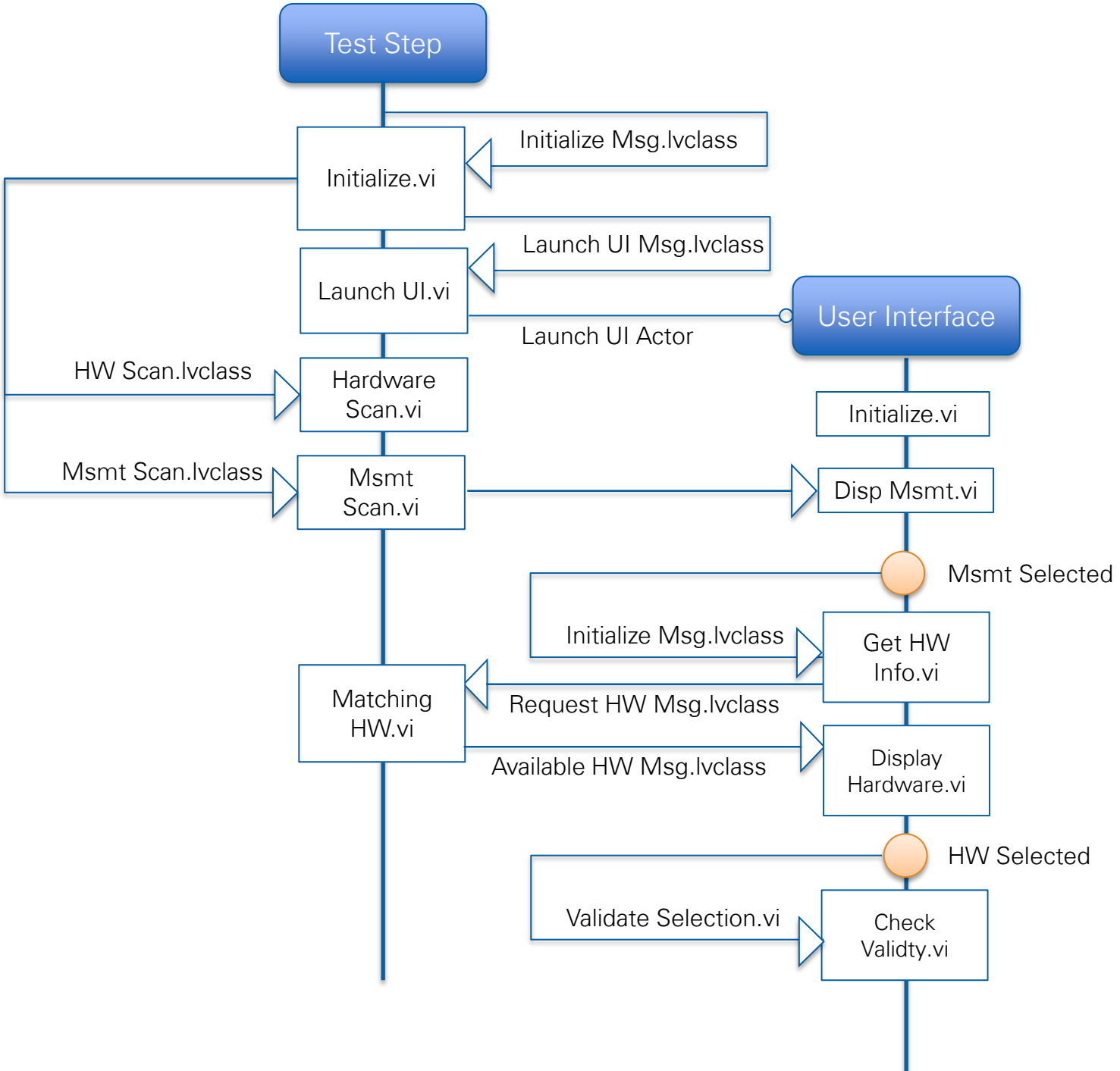
Measurement System Sample Project

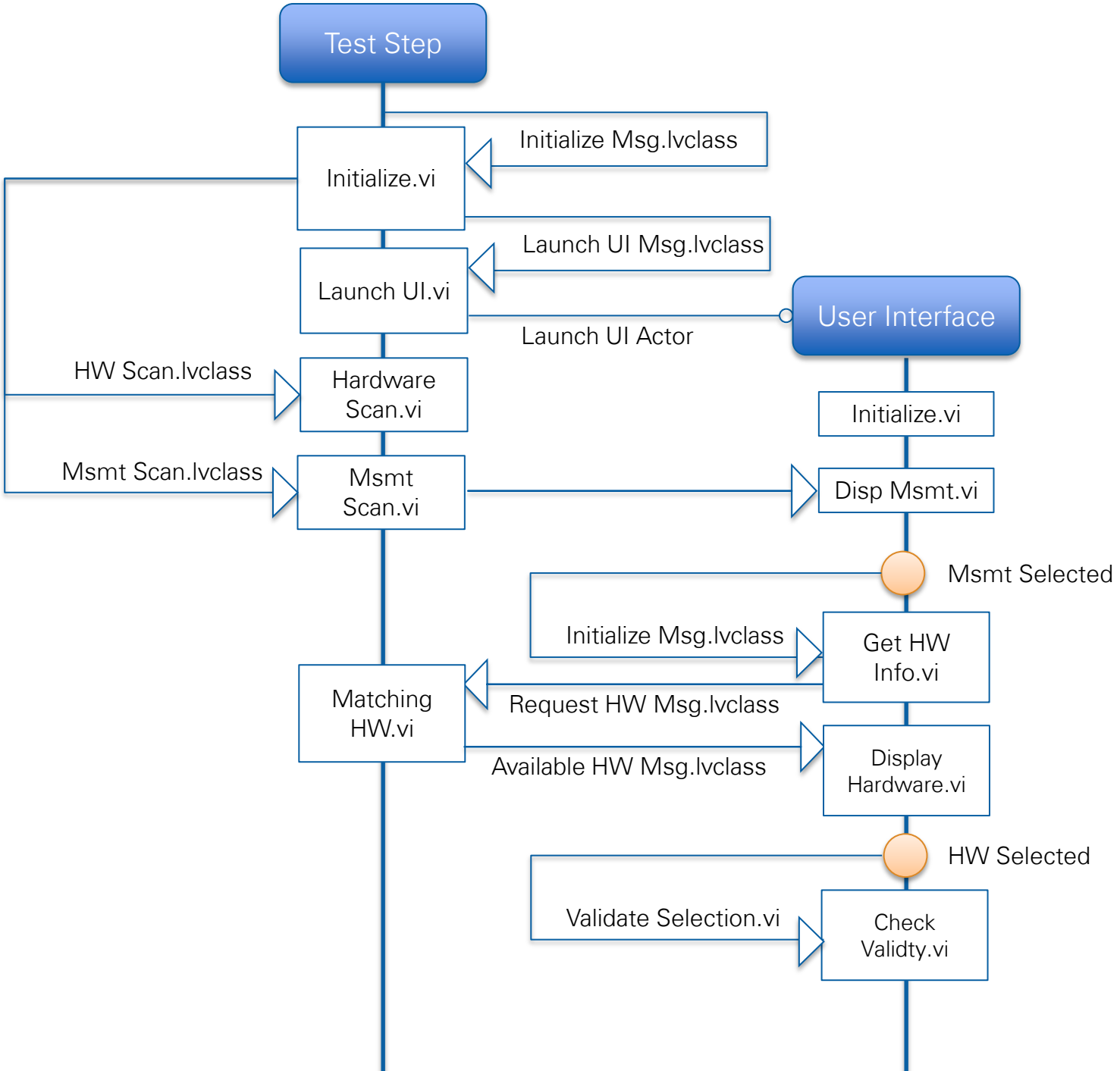
Class hierarchy view

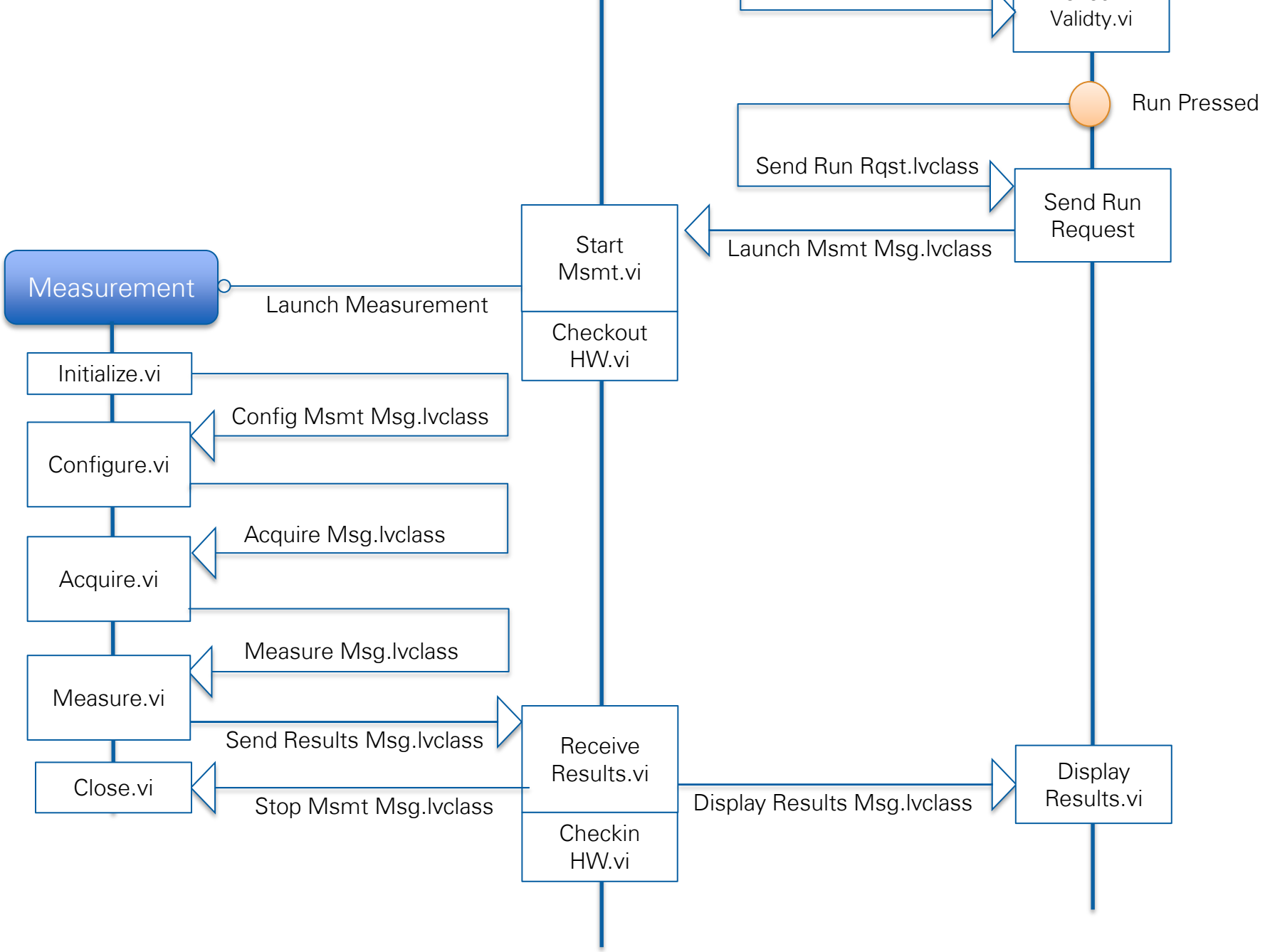


Test Step Task Tree

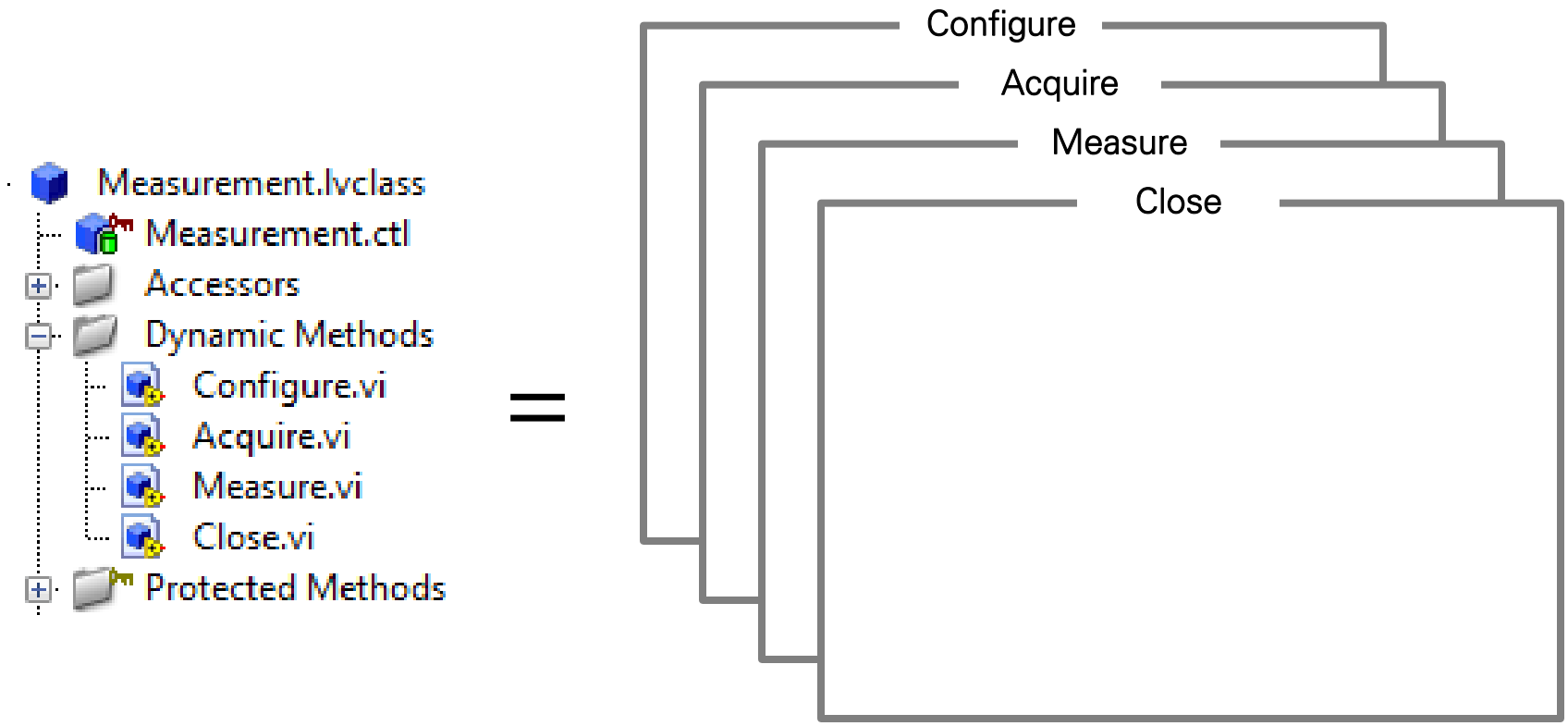



Test Step
Initialize.vi
Initialize Msg.lvclass
Launch UI.vi
Launch UI Msg.lvclass
User Interface
Launch UI Actor
HW Scan.lvclass
Hardware Scan.vi
Initialize.vi
Msmt Scan.lvclass
Msmt Scan.vi
Disp Msmt.vi
Msmt Selected
Initialize Msg.lvclass
Get HW Info.vi
Matching HW.vi
Request HW Msg.lvclass
Available HW Msg.lvclass
Display Hardware.vi
HW Selected
Validate Selection.vi
Check Validty.vi


Test Step
Initialize.vi
Initialize Msg.lvclass
Launch UI.vi
Launch UI Msg.lvclass
User Interface
Launch UI Actor
HW Scan.lvclass
Hardware Scan.vi
Initialize.vi
Msmt Scan.lvclass
Msmt Scan.vi
Disp Msmt.vi
Msmt Selected
Initialize Msg.lvclass
Get HW Info.vi
Matching HW.vi
Request HW Msg.lvclass
Available HW Msg.lvclass
Display Hardware.vi
HW Selected
Validate Selection.vi
Check Validty.vi


Validty.vi
Run Pressed
Send Run Rqst.lvclass
Send Run Request
Launch Msmt Msg.lvclass
Start Msmt.vi
Checkout HW.vi
Measurement
Launch Measurement
Initialize.vi
Config Msmt Msg.lvclass
Configure.vi
Acquire Msg.lvclass
Acquire.vi
Measure Msg.lvclass
Measure.vi
Send Results Msg.lvclass
Close.vi
Stop Msmt Msg.lvclass
Receive Results.vi
Checkin HW.vi
Display Results Msg.lvclass
Display Results.vi

Measurement QSM



We are going to override Acquire and Measure

Checklist for Creating a Plugin

Project Name: Advanced Sample Project Session

Measurement: NIWeek Demo

Methods:

- Acquire
- Measure

Hardware

- Digital Multimeter (DMM)
- Power Supply (PSU)

Adding a Template

Creating a New Template

Required

1. Create the source code for template
2. Create the XML document for template

Optional: Scripting on Copied Code

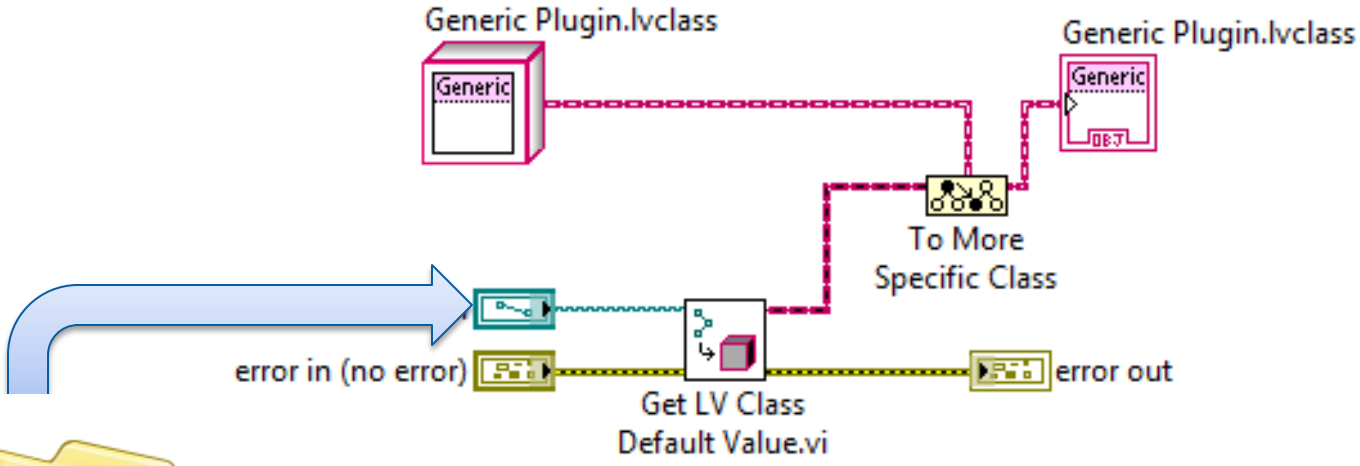
1. Create a custom scripting method to modify template

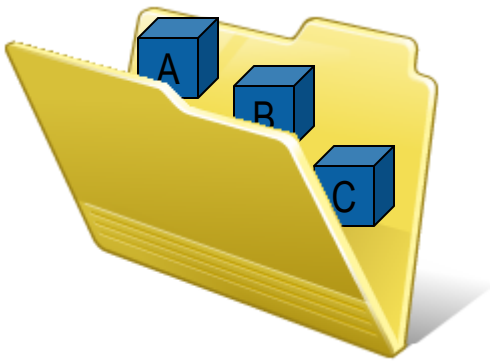
Optional: Custom Create Project Dialog

1. Override the MetaData object (resources folder)
2. Create a new ‘Spec Page’ *(must have required inputs)*

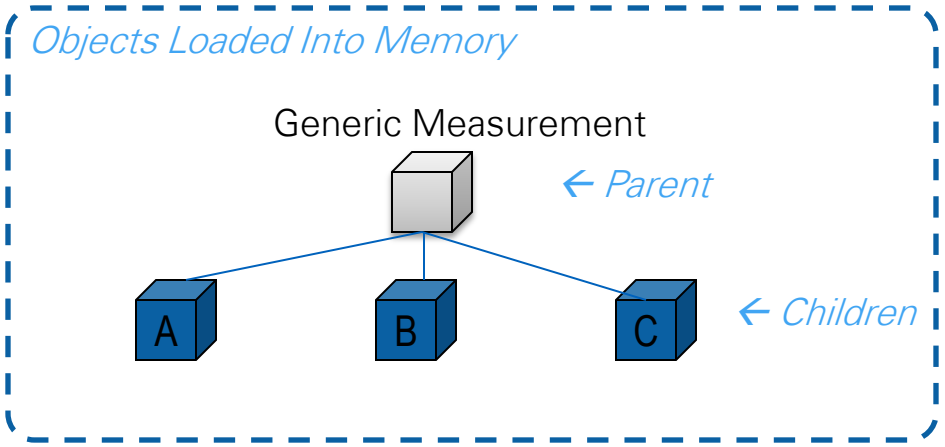
Information Available Here

The Basics of an Object Factory





Location on Disk
Where Measurements are Stored



NATIONAL INSTRUMENTS

More Information on Architectures and Process

Dedicated to LabVIEW Development and Software Engineering Practices

Technical White Paper Series

ni.com/largeapps

Online Community Dedicated to Development Best Practices

ni.com/community/largeapps

Follow My Blog on Software Engineering with LabVIEW

ekerry.wordpress.com

Follow me on Twitter

elijah286