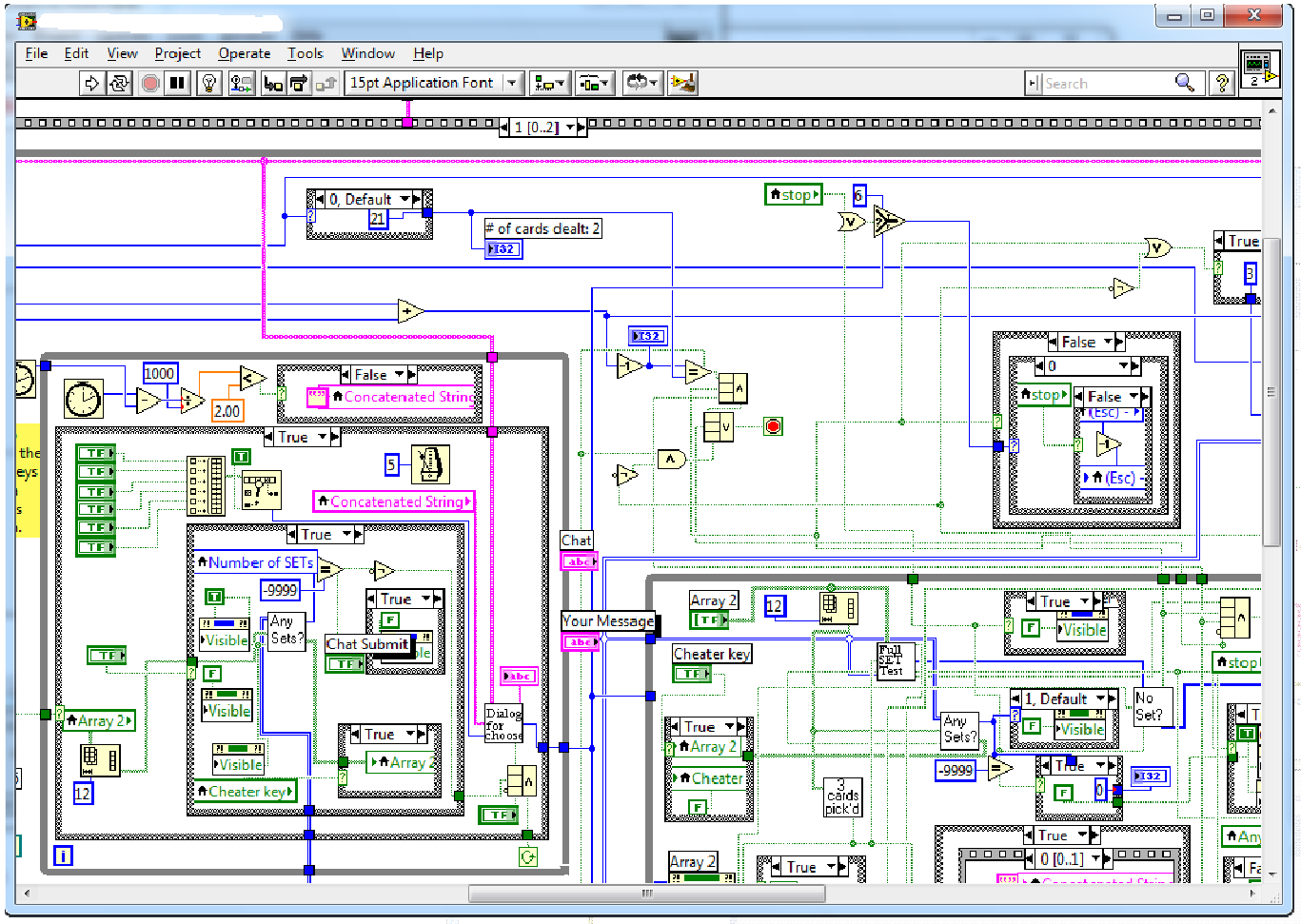
Welcome to

NI LabVIEW | Power User Seminar

Elijah Kerry – LabVIEW Product Manager

Certified LabVIEW Architect (CLA)



NATIONAL INSTRUMENTS

Examples of Software Engineering Debt

(just *some* of the most common LabVIEW development mistakes)

- ✓ No source code control (or Project)
- ✓ Flat file hierarchy
- ✓ ‘Stop’ isn’t tested regularly
- ✓ Wait until the ‘end’ of a project to build an application
- ✓ Few specifications / documentation / requirements
- ✓ No ‘buddying’ or code reviews
- ✓ Poor planning (Lack of consideration for SMoRES)
- ✓ No test plans
- ✓ Poor error handling
- ✓ No consistent style
- ✓ Tight coupling, poor cohesion

The Cost of a Software Defect

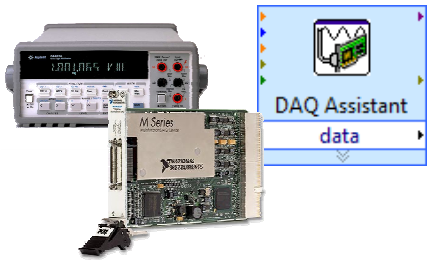
Development Phase	Cost Ratio
Requirements	1
Design	3-6x
Implementation	10x
Development Testing	15-40x
Acceptance Testing	30-70x
Post Release	40-1000x

Based on an analysis of 63 software development projects at companies including IBM, GTE and TRW

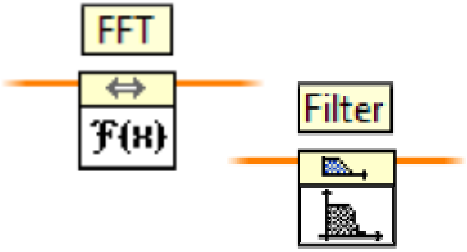
NATIONAL INSTRUMENTS

LabVIEW™

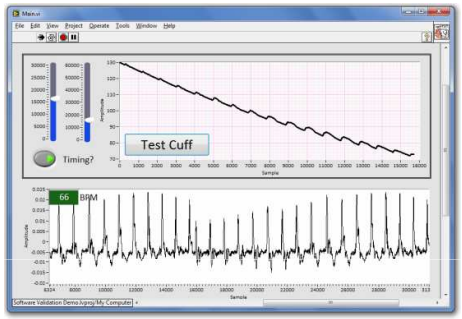
A Highly Productive Graphical Development Environment for Scientists and Engineers



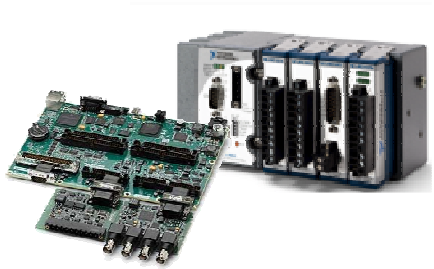
Hardware APIs



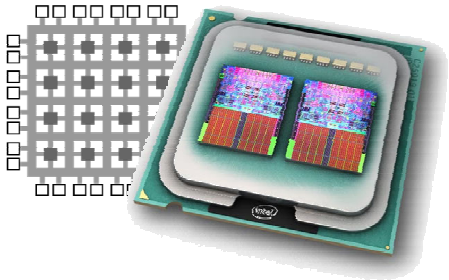
Analysis Libraries



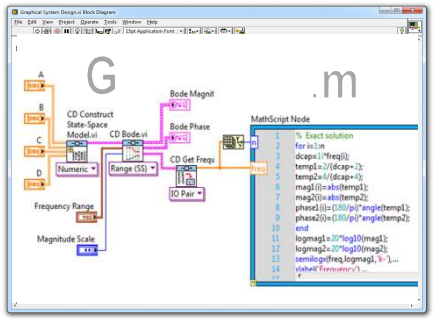
Custom User Interfaces



Deployment Targets



Technology Abstractions



Programming Approaches

NATIONAL INSTRUMENTS

Lawrence Livermore National Labs

Developed automated maintenance process for world's most energetic laser array at the National Ignition Facility using NI LabVIEW and PXI

- LabVIEW increased productivity by 3X over Java and C++
- Developed complex application consisting of over 1,000 VIs
- Applied software engineering practices to ensure quality

An overhead view of one of the main laser chambers

"The value in using the graphical dataflow language is the speed in which a team can deliver a robust solution while still using proper software engineering practices.

- Glenn Larkin, LLNL

NATIONAL INSTRUMENTS

LabVIEW in Large Applications

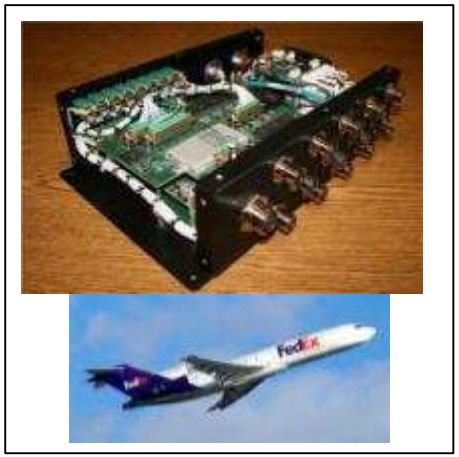
High-Volume Production Test

Structural Health at Olympics

Medical Devices

Nexans Spider Robot

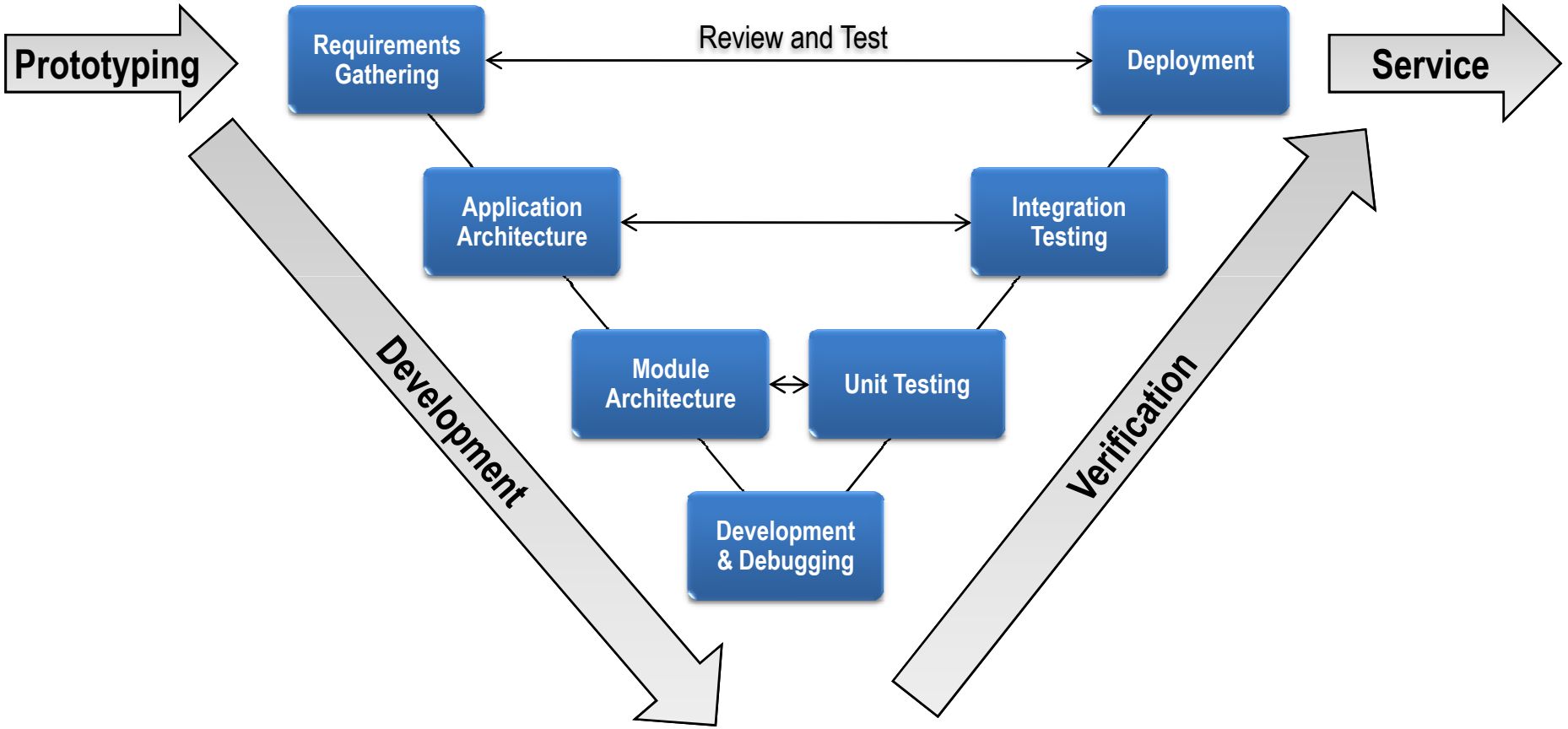
Large Physics Applications



In-Flight Fire Suppression

NATIONAL INSTRUMENTS

Software Engineering V-Model



NATIONAL INSTRUMENTS

Seminar Topics

- Configuration Management and Code Reuse
- Requirements Management and Traceability
- Peer Reviews and Code Analysis
- Testing and Verification of Software
- Object-Oriented Design Patterns
- Hardware Abstraction Layers
- Data Communication Approaches
- Error Handling
- Advanced User Interface Design

ni.com/largeapps

NATIONAL INSTRUMENTS

Application Introductions

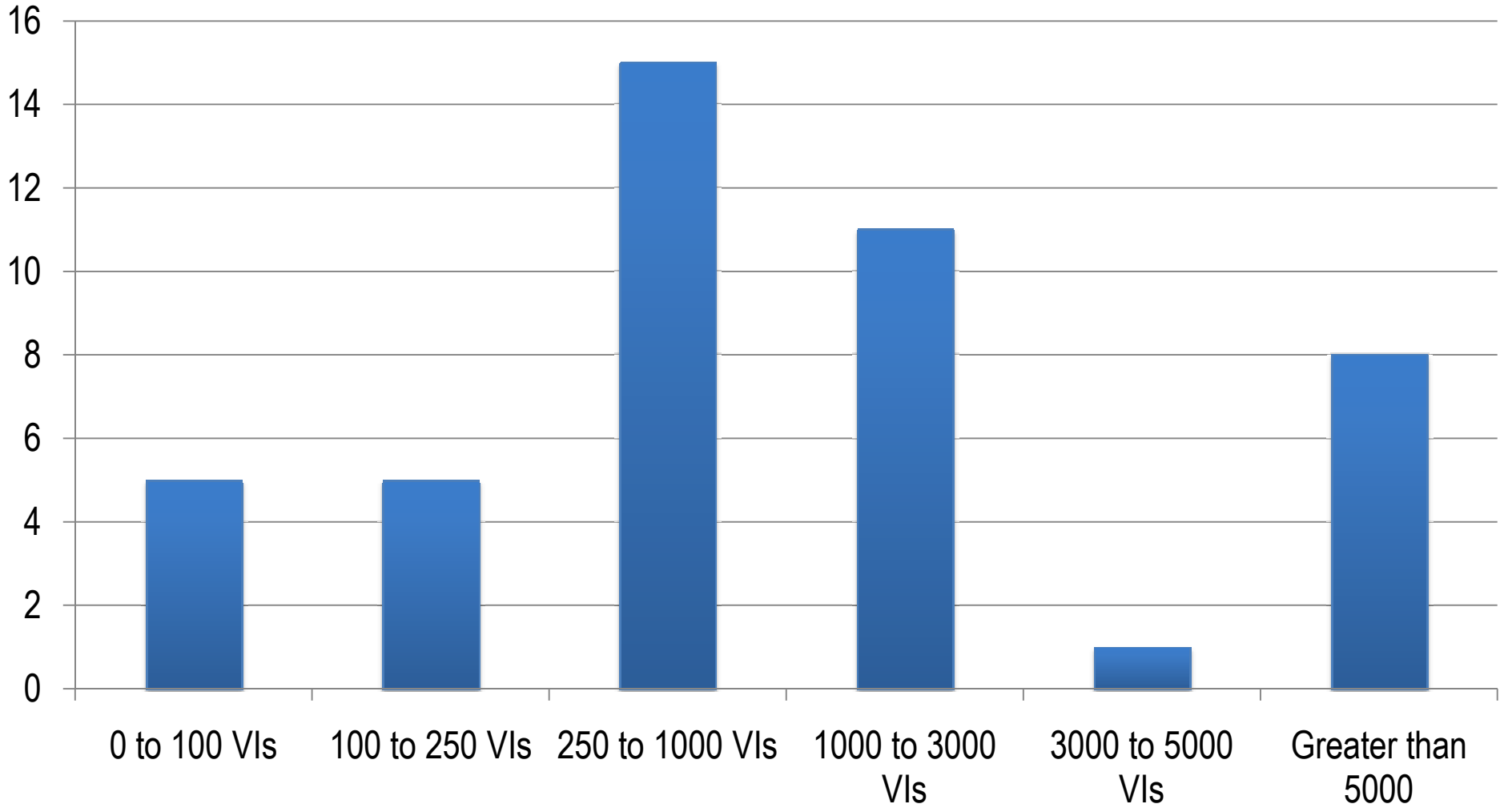
DEMO

Section One

Configuration Management and Code Reuse

NATIONAL INSTRUMENTS

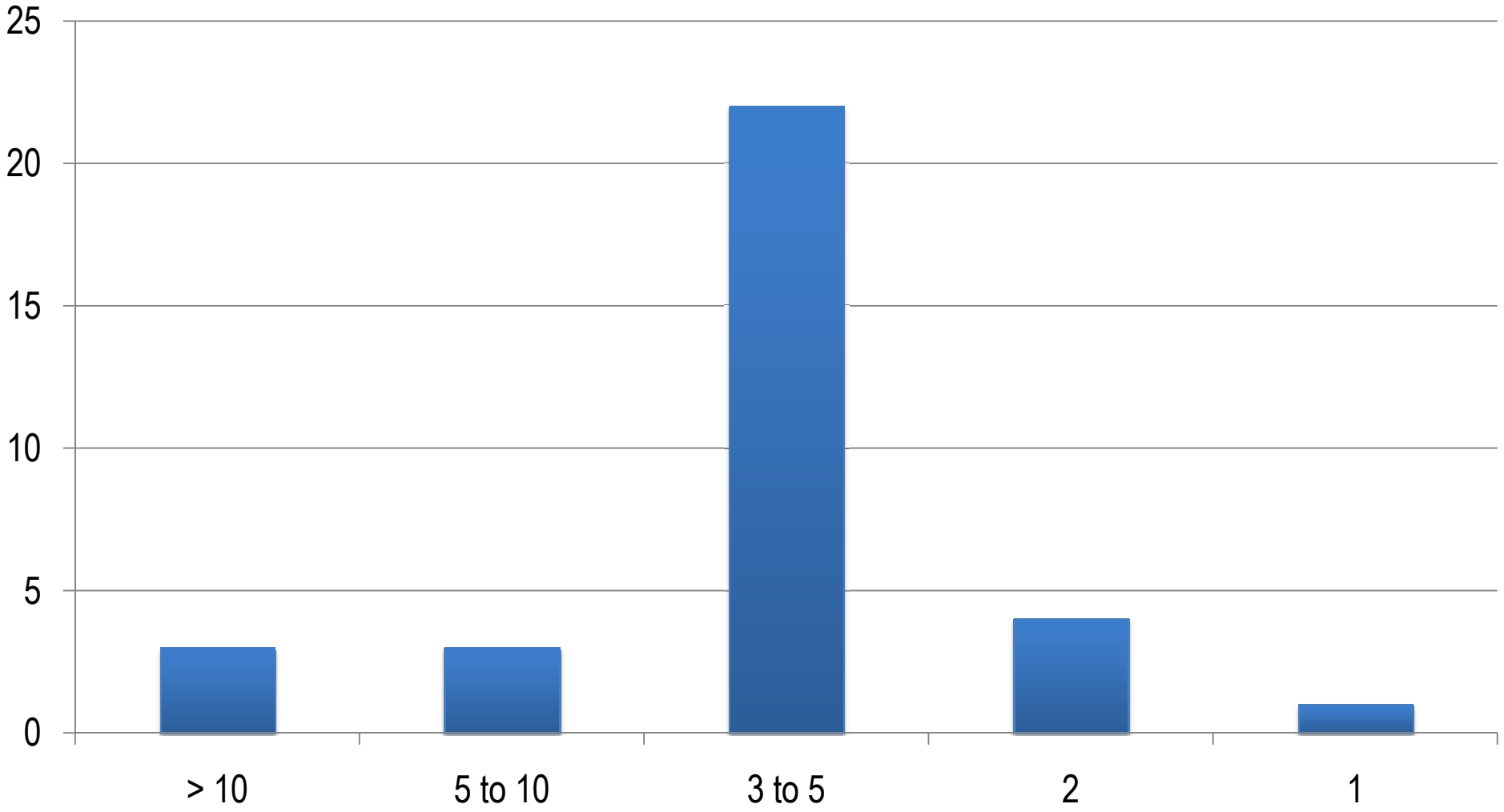
Size of LabVIEW Applications



***Source**: 2010 ni.com/largeapps survey*

NATIONAL INSTRUMENTS

Average Number of Developers Per Project



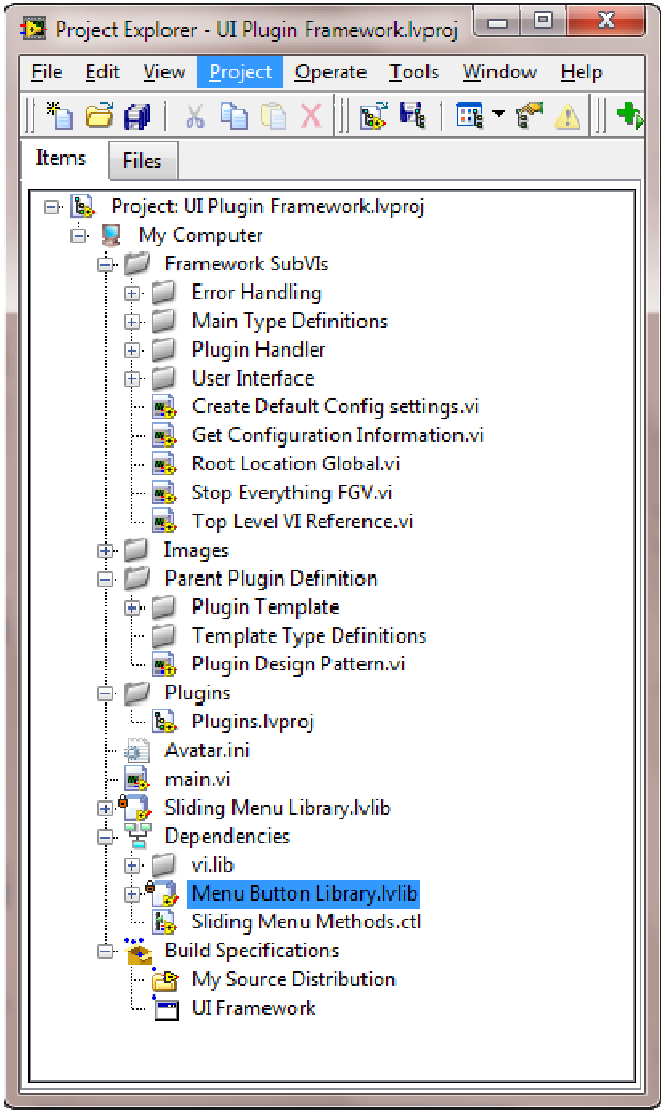
***Source**: NIWeek 2008 Software Engineering Survey*

NATIONAL INSTRUMENTS

The Project Explorer

Benefits

- Easily access and navigate files
- Preserve links when moving files
- Manage hardware targets
- Detect and find dependencies
- Prevent and auto-detect cross-linking
- Integration with Application Builder
- Access to source code control



NATIONAL INSTRUMENTS

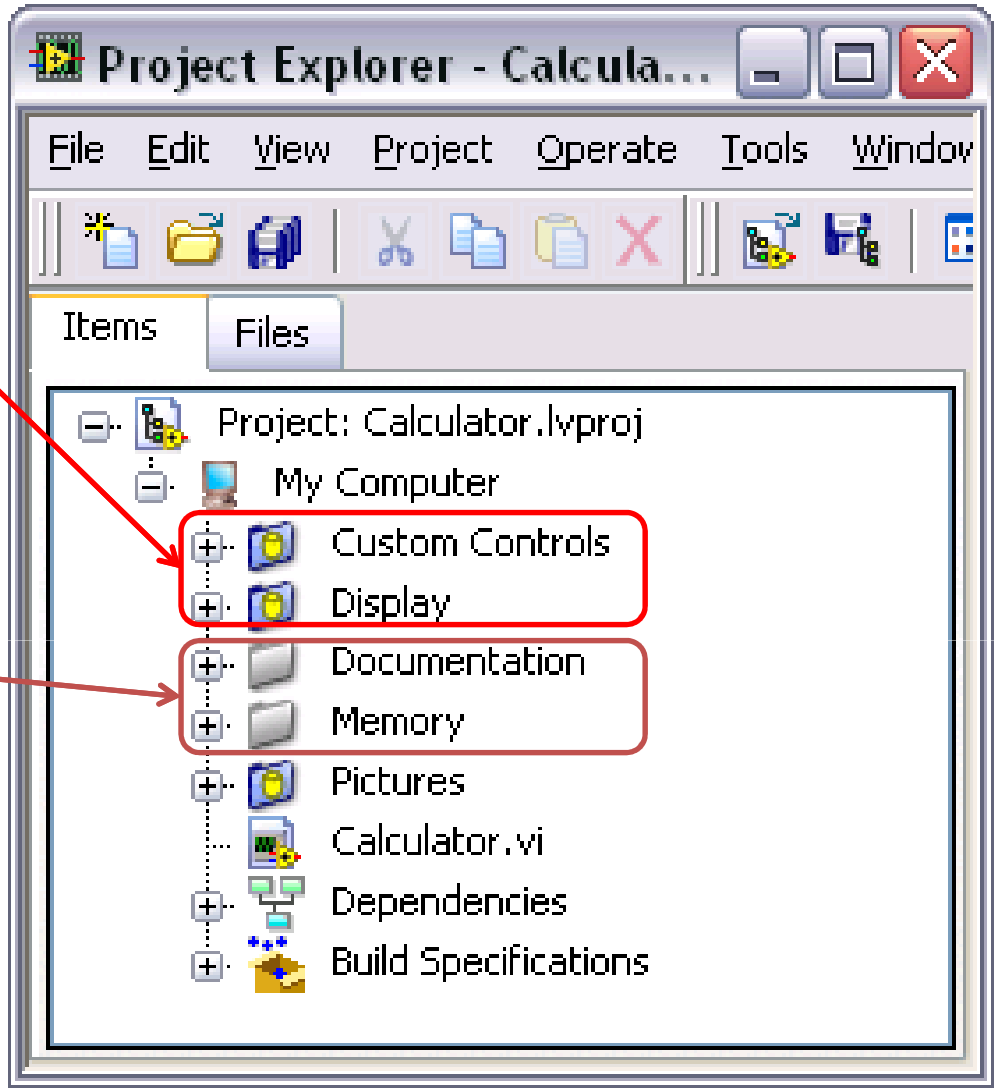
Managing Files

Autopopulating Folders

are synchronized to disk. They update in real time to reflect the contents of folders on disk.

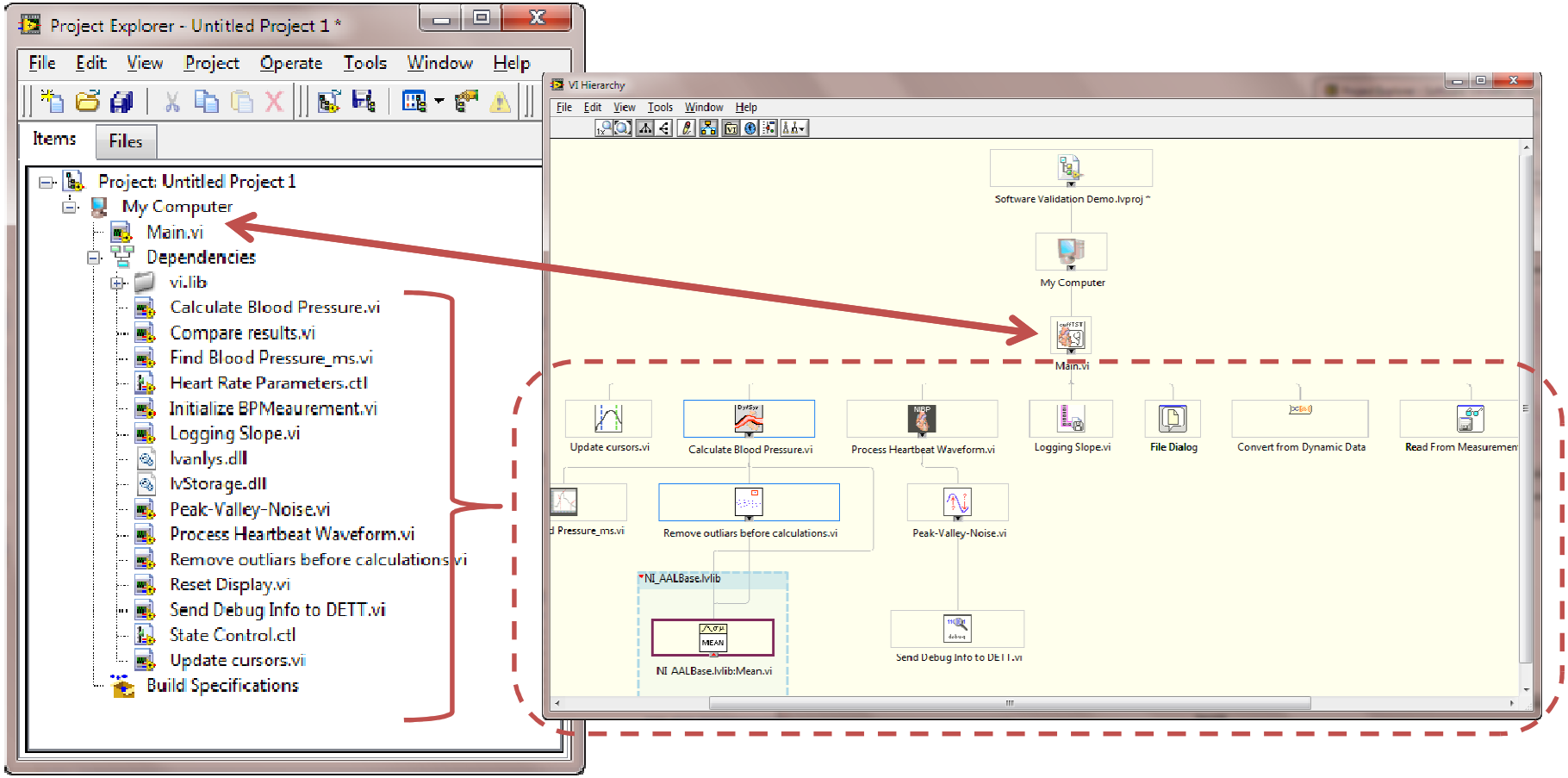
Virtual Folders

allow you to customize how and where files are displayed. They provide a ‘snapshot’ view if added from disk



NATIONAL INSTRUMENTS

Managing Dependencies



NATIONAL INSTRUMENTS

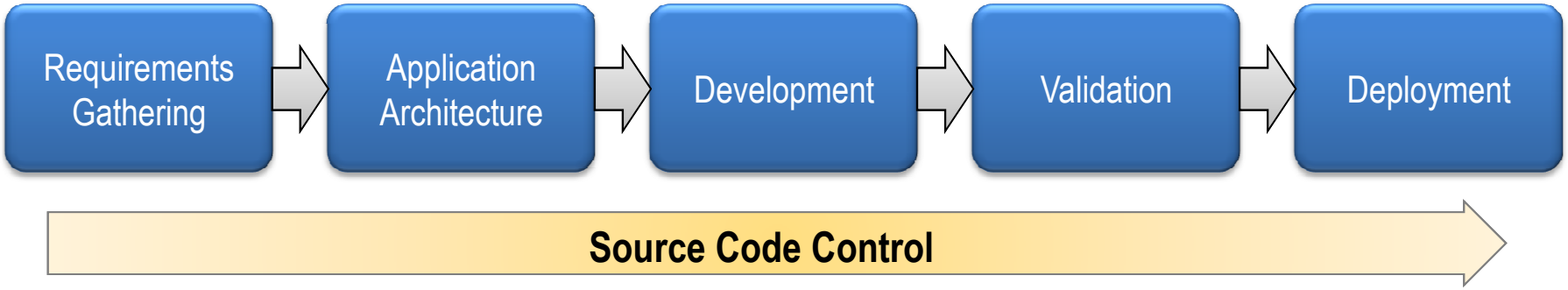
Moving an Application into the Project Explorer

DEMO

Project Explorer Recommendations

- Use virtual folders by default
- Use virtual folders to contain any .lvlib-like object
- Use auto-populating folders for components that you want to automatically bring into the project.
  (examples: log-files, pictures, plugins, test results)
- Check the dependencies section often
- Move project items from within the Files tab
- Break-up large applications into multiple projects

Software Configuration Management Tools

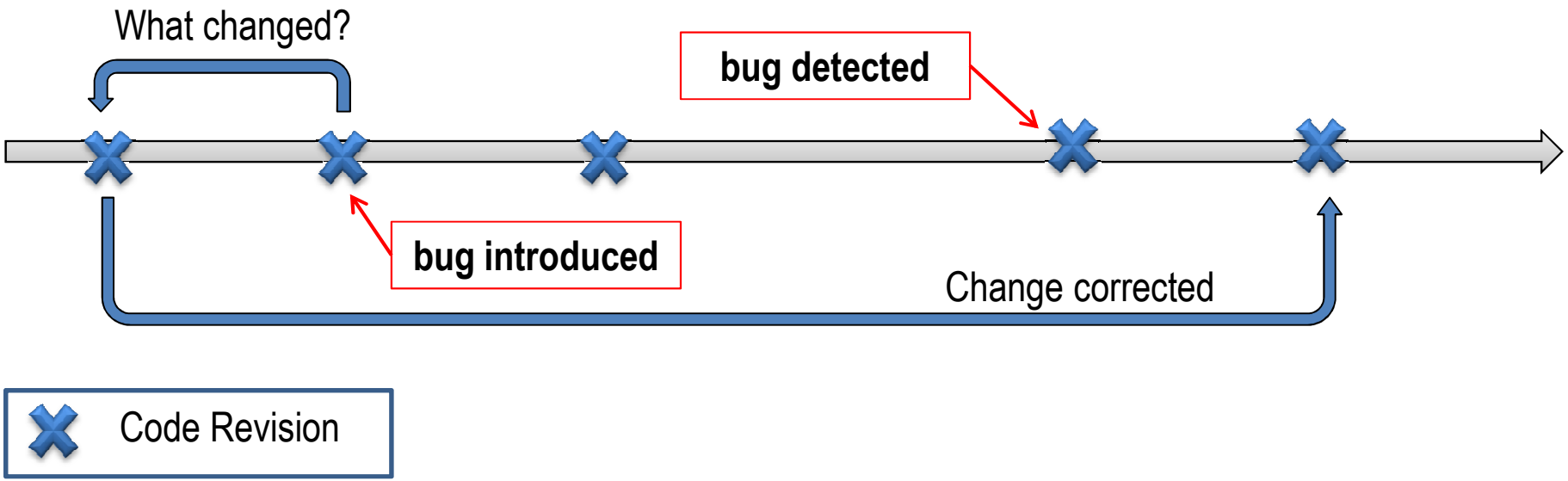


- ✓ Provides repository of code
- ✓ Store multiple versions and branches
- ✓ Easily track changes
- ✓ Manage a team of developers
- ✓ Reduces pain throughout process

NATIONAL INSTRUMENTS

Tracking Changes to Source Code

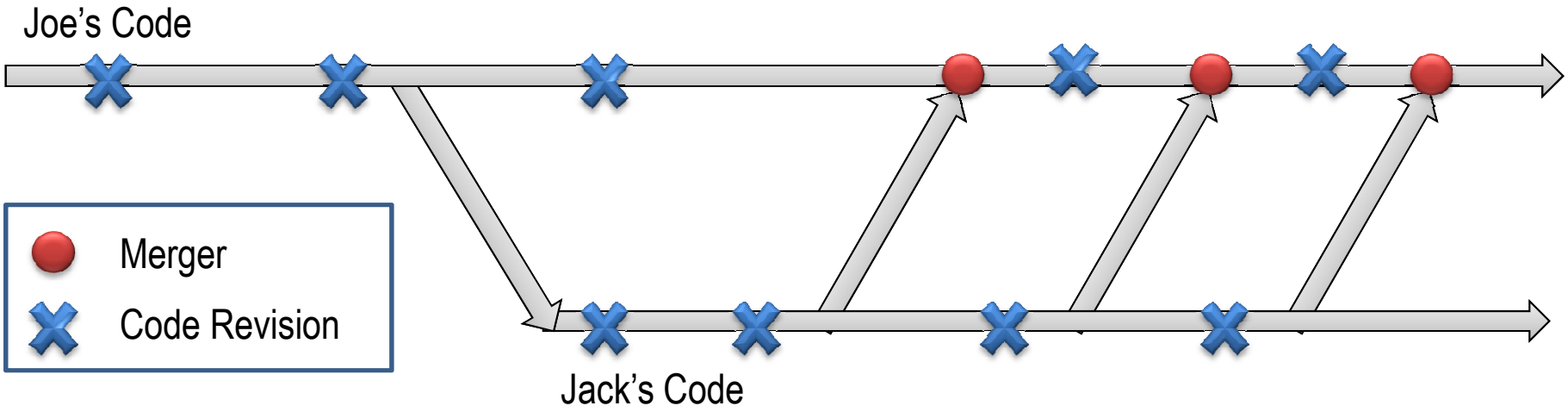
- A change to the code causes a problem that isn't detected for several revisions
- How can you track changes over time to identify when the behavior changed?



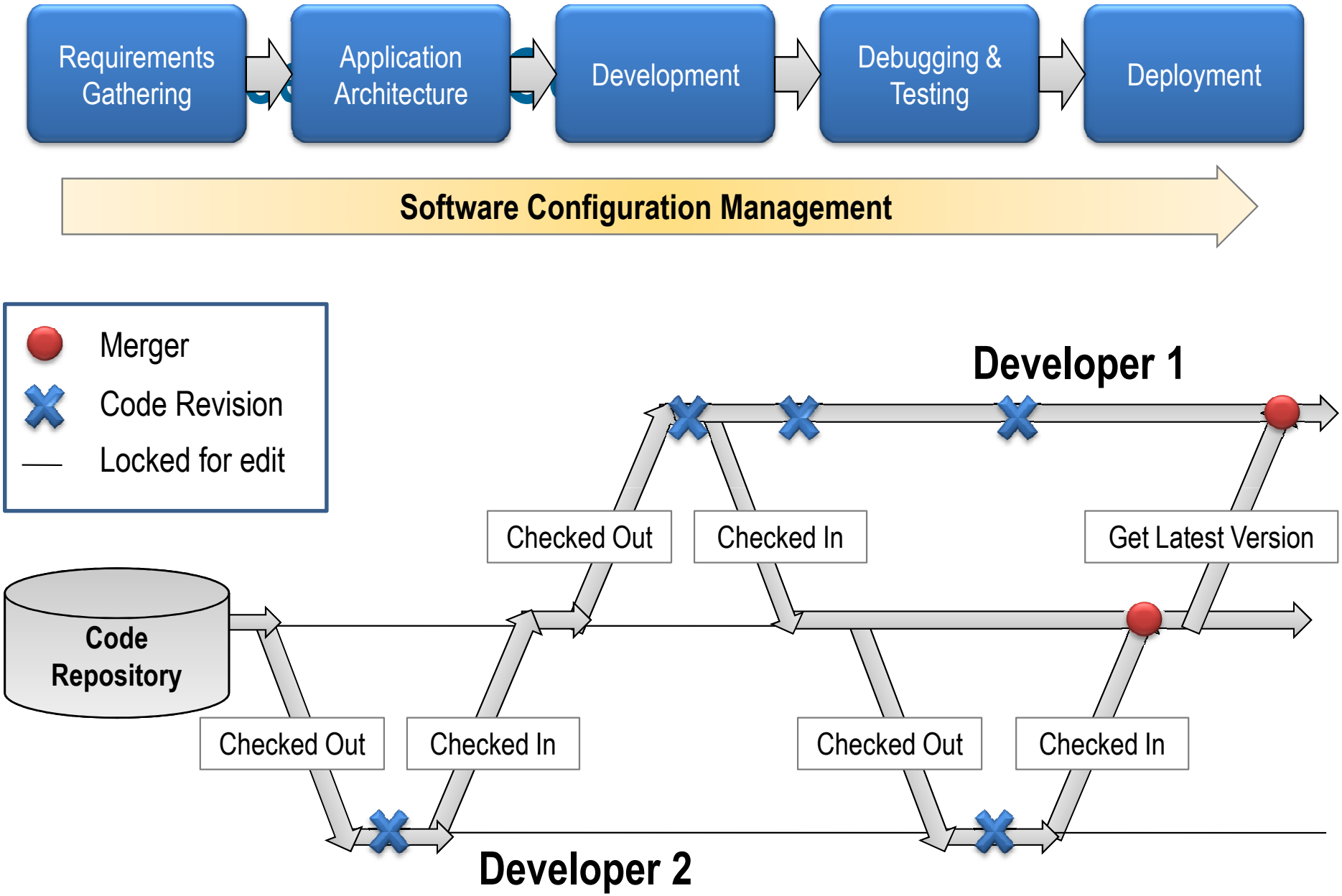
NATIONAL INSTRUMENTS

Group Development

- Two developers, Joe and Jack, are working together
- Jack copies (a.k.a. ‘branches’) Joe’s code
- Joe and Jack are making changes to the same application
- How do they track changes to shared dependencies and expedite the process of merging their work?

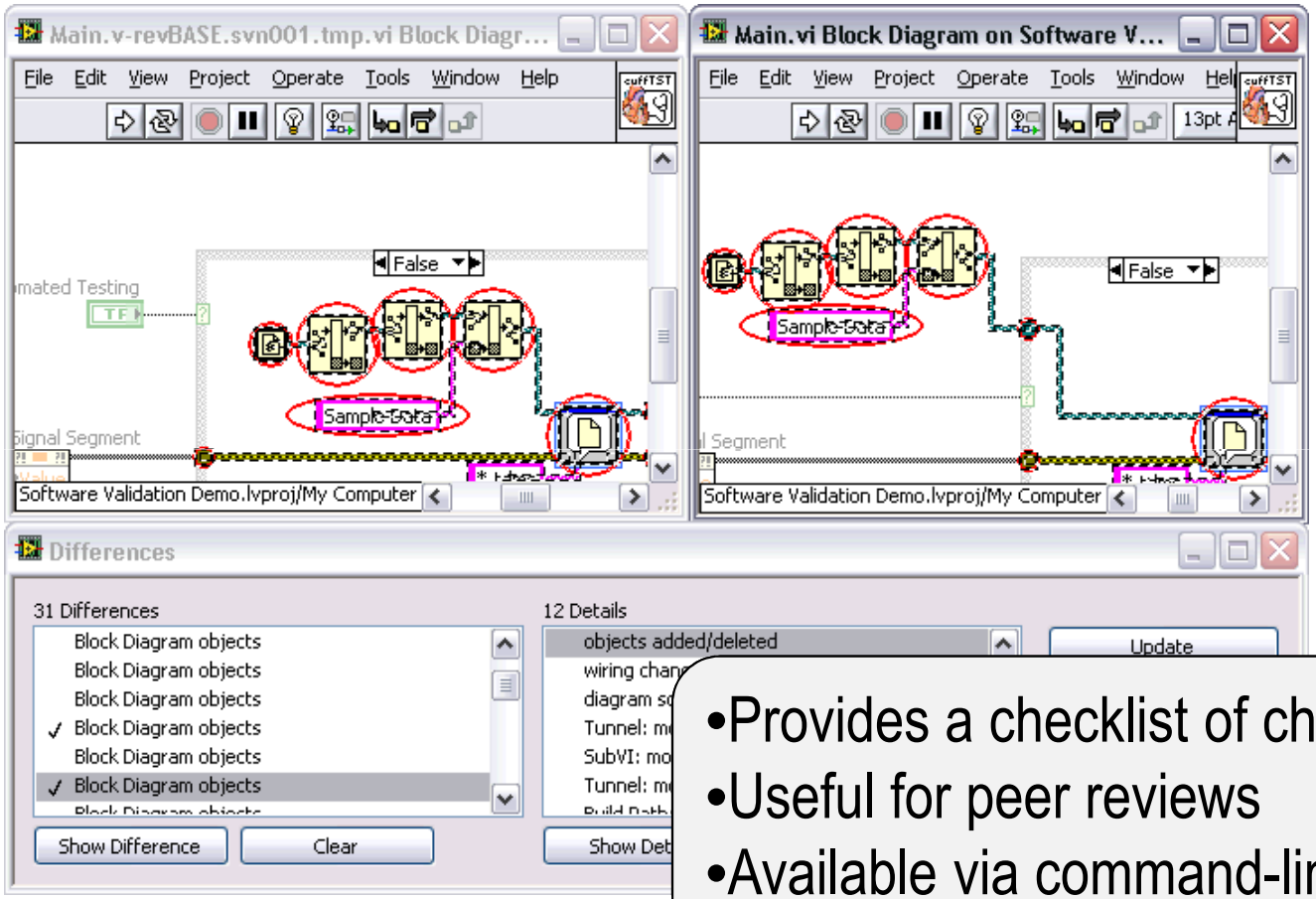


NATIONAL INSTRUMENTS



NATIONAL INSTRUMENTS

Graphical Differencing



- Provides a checklist of changes
- Useful for peer reviews
- Available via command-line

NATIONAL INSTRUMENTS

SCC Options for Integration within LabVIEW

Native LabVIEW Integration

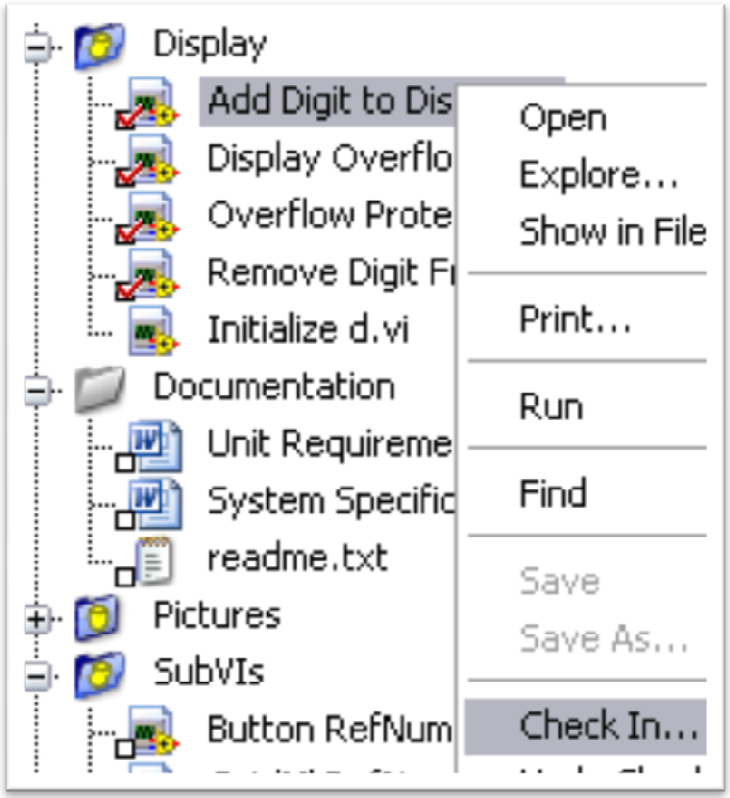
- Perforce

Integration Through Standard API

- Microsoft Visual SourceSafe
- Microsoft Team System
- Rational ClearCase
- PCVS (Serena) Version Manager
- MKS Source Integrity
- Seapine Surround SCM
- Borland StarTeam
- Telelogic Synergy
- ionForge Evolution

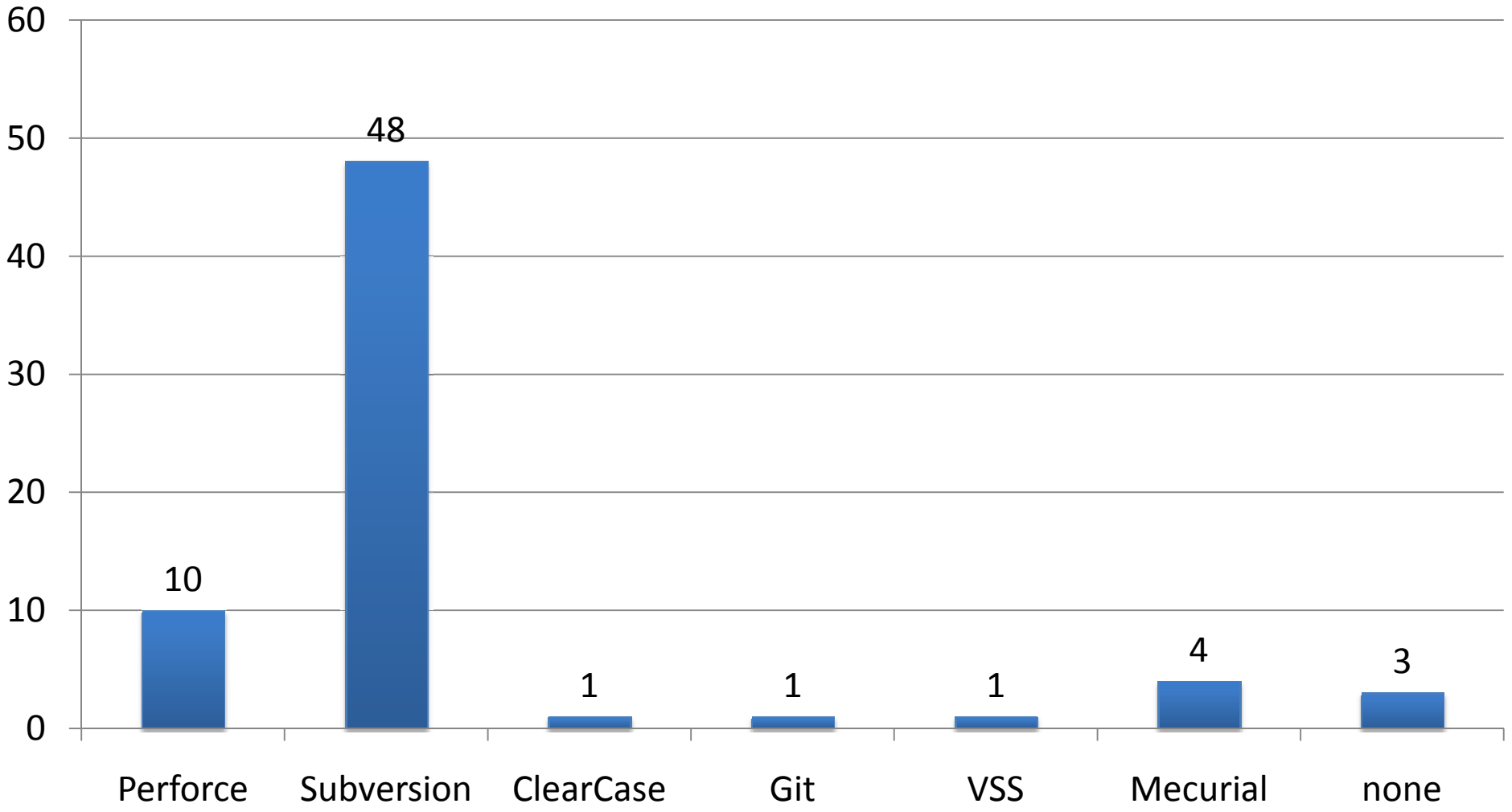
Support through additional add-ons

- Subversion
- Mecurial



NATIONAL INSTRUMENTS

Popularity of SCC Options Amongst LabVIEW Programmers



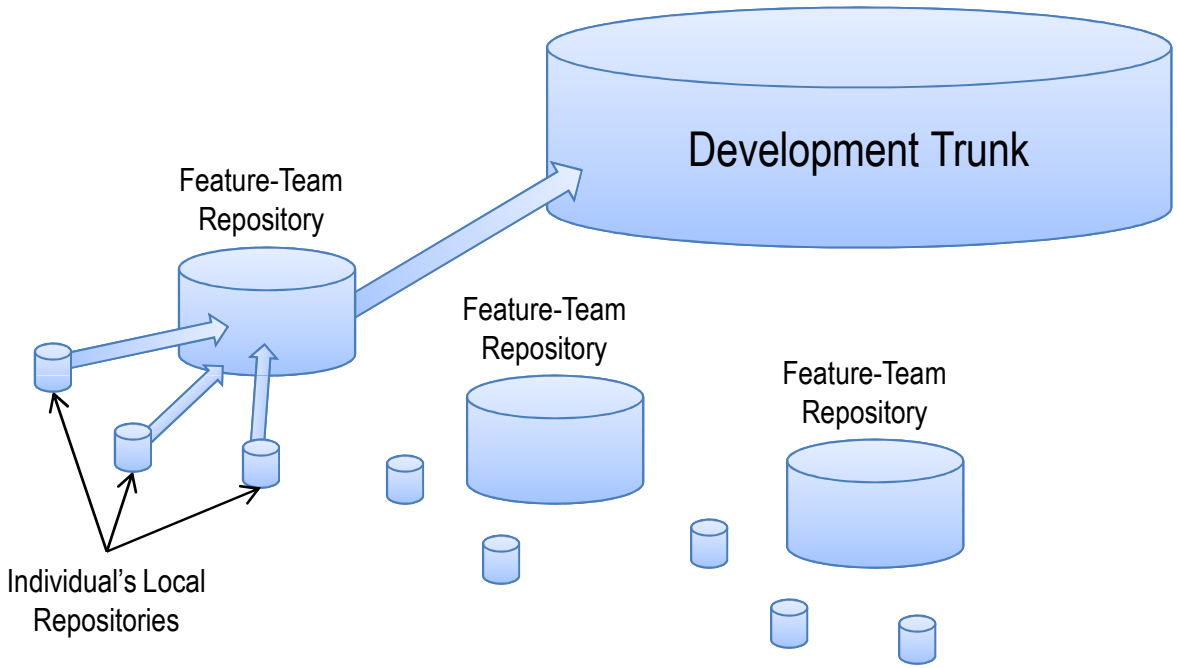
***Source**: 2010 ni.com/largeapps survey*

Configuration Management

DEMO

NI Configuration Management

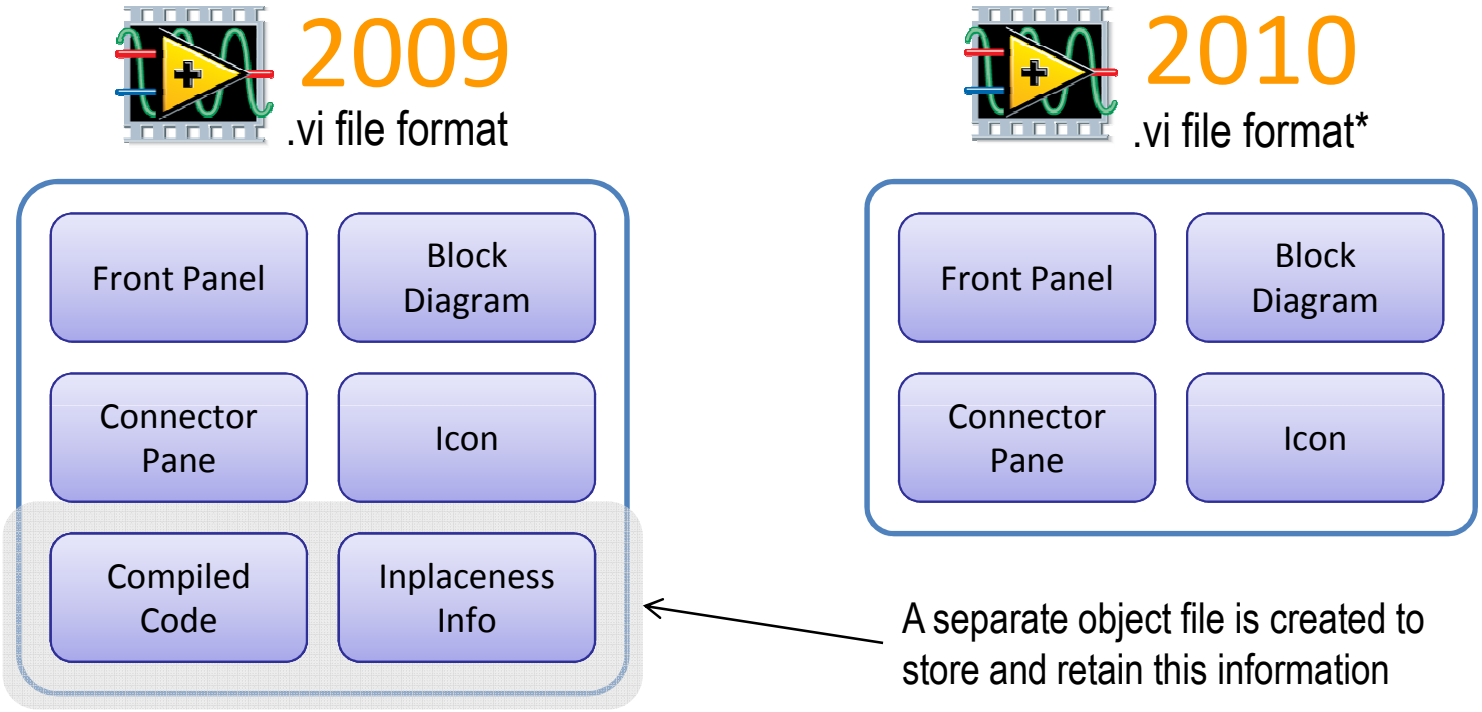
- Different trunk for each LabVIEW version
- Teams of 3 to 7 developers work in smaller repositories
- Individuals may have their own repositories
- New features and changes are regularly merged in



NATIONAL INSTRUMENTS

Separate Compiled Code From Source File

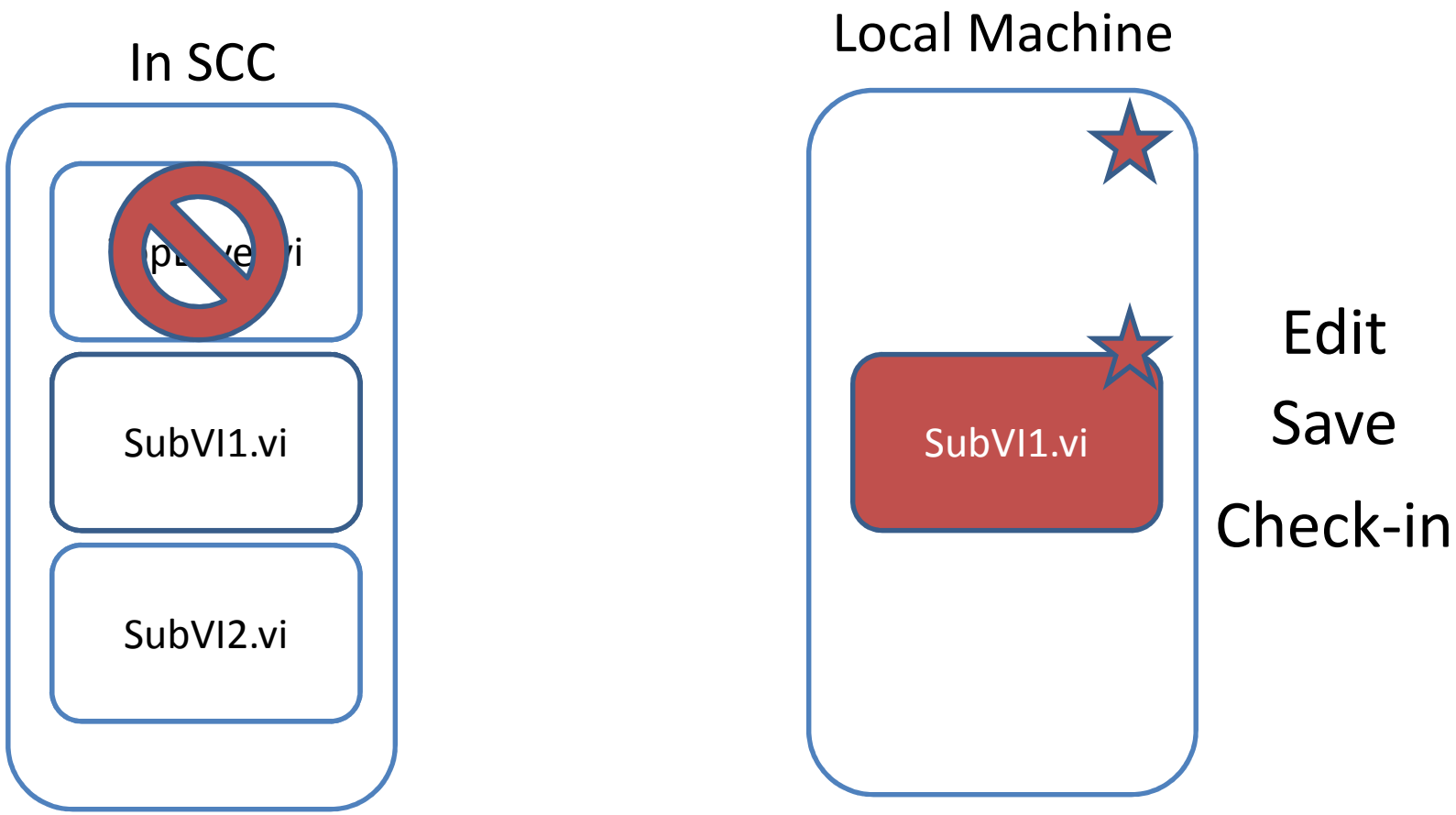
Improved Source Code Control Integration



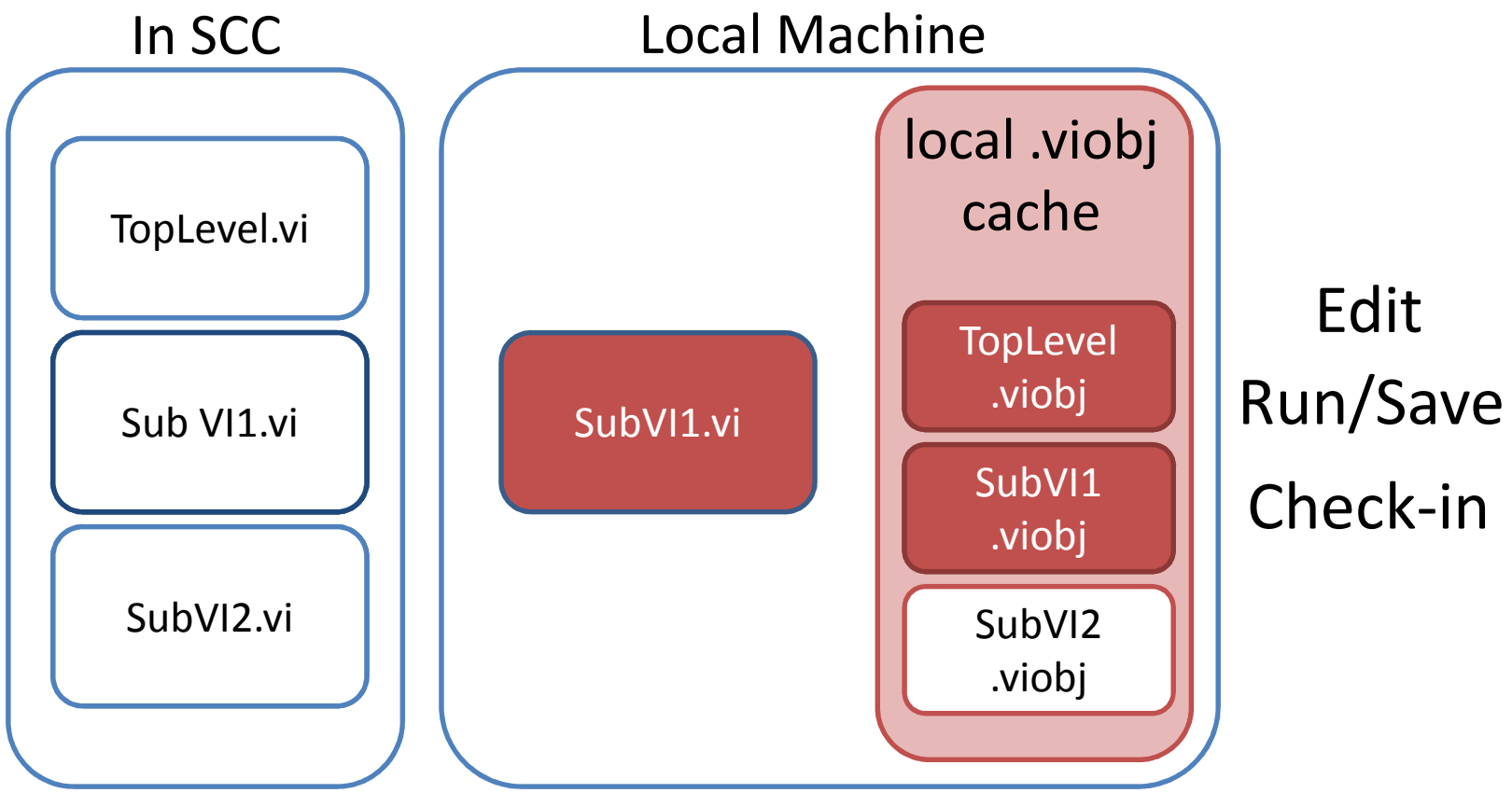
Eliminate the need to re-save and re-submit files to source code control unless the graphical source code has been changed by the developer

**this feature is not on by default and needs to be enabled from the VI Properties dialog*

Source Code Control Scenario: Today

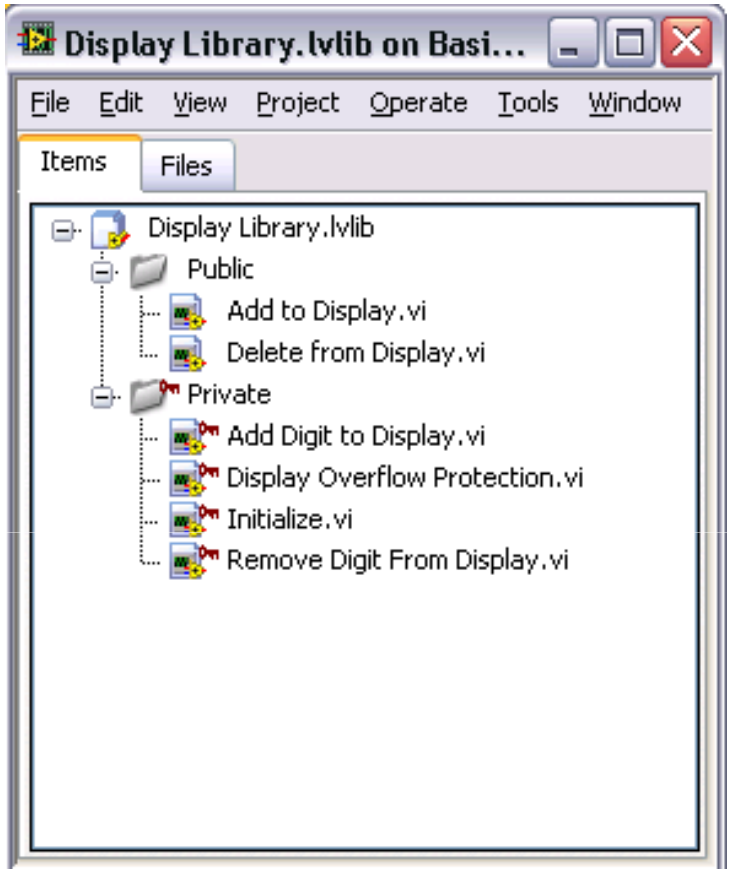


Source Code Control Scenario: 2010



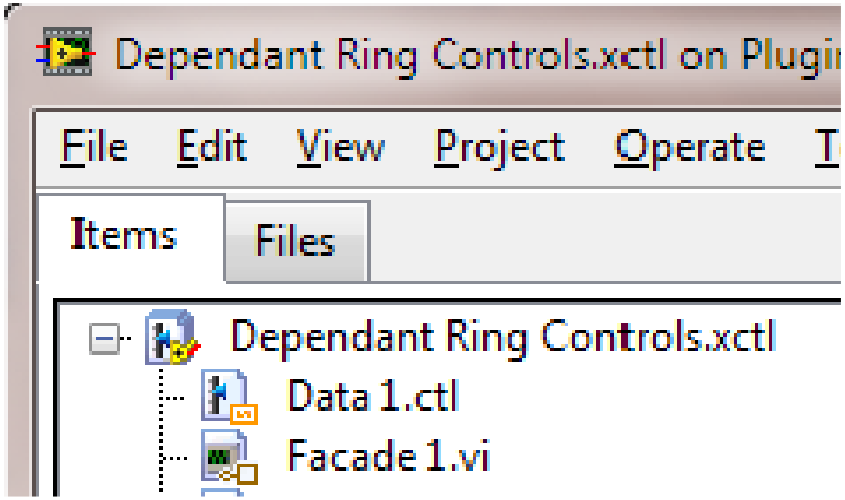
The Project Library

- Provide name-spacing to avoid cross-links
- Organize sections of an application into components
- Modify contents without modifying the Project file
- Pack them into a single, non-editable file

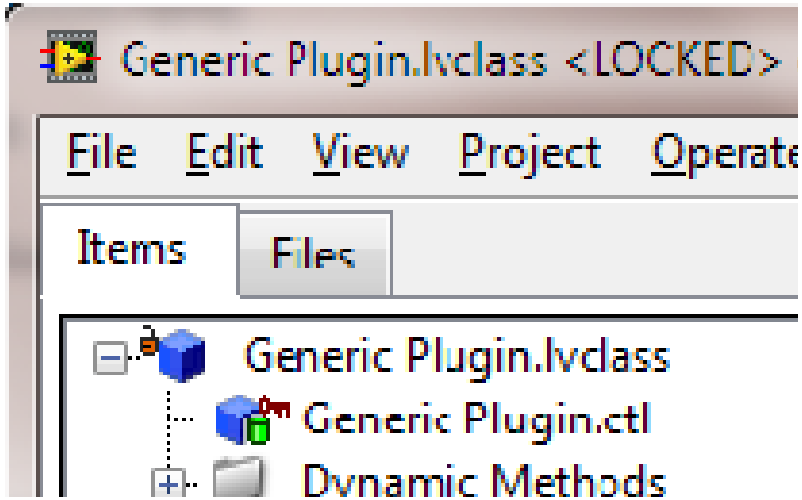


Project Libraries (.lvlib)

Xcontrols (.xctl)



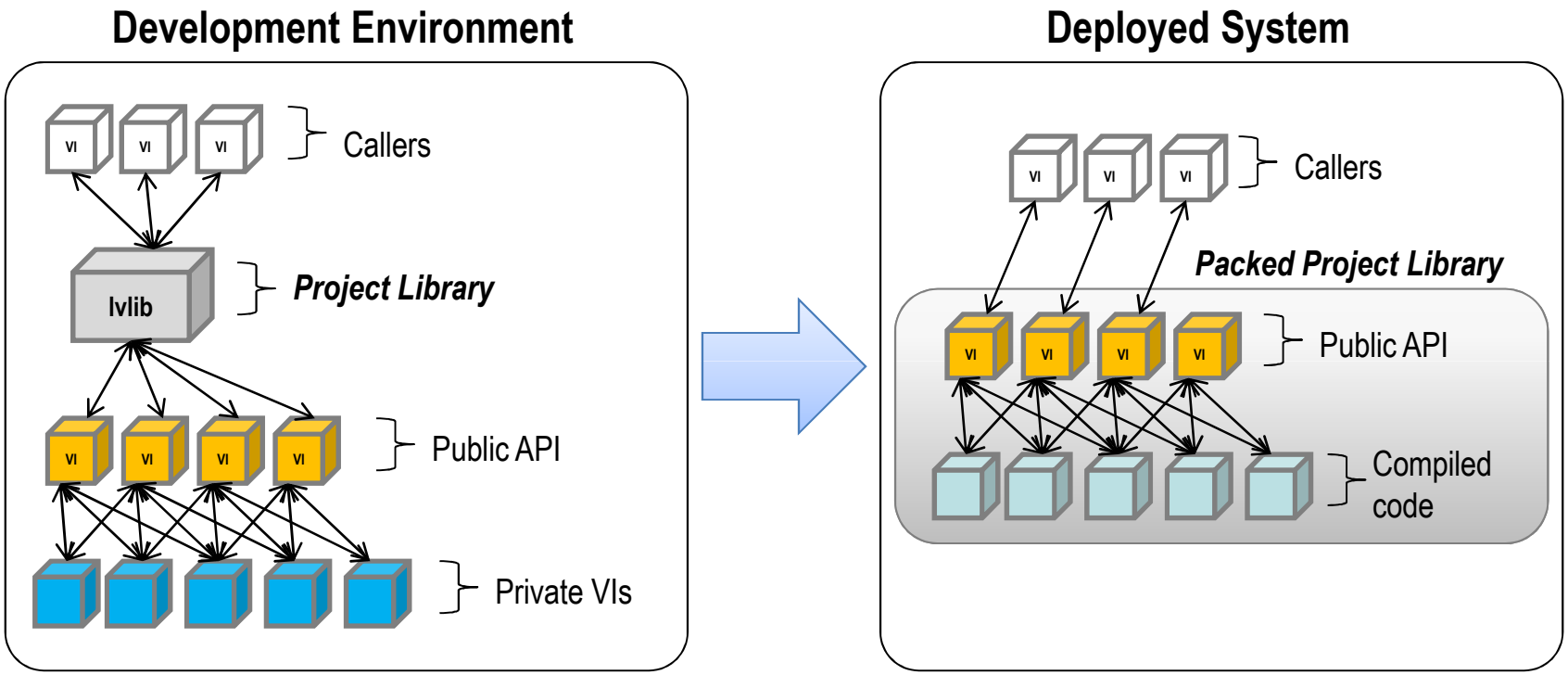
Classes (.lvclass)



NATIONAL INSTRUMENTS

Packed Project Libraries *New in LabVIEW 2010*

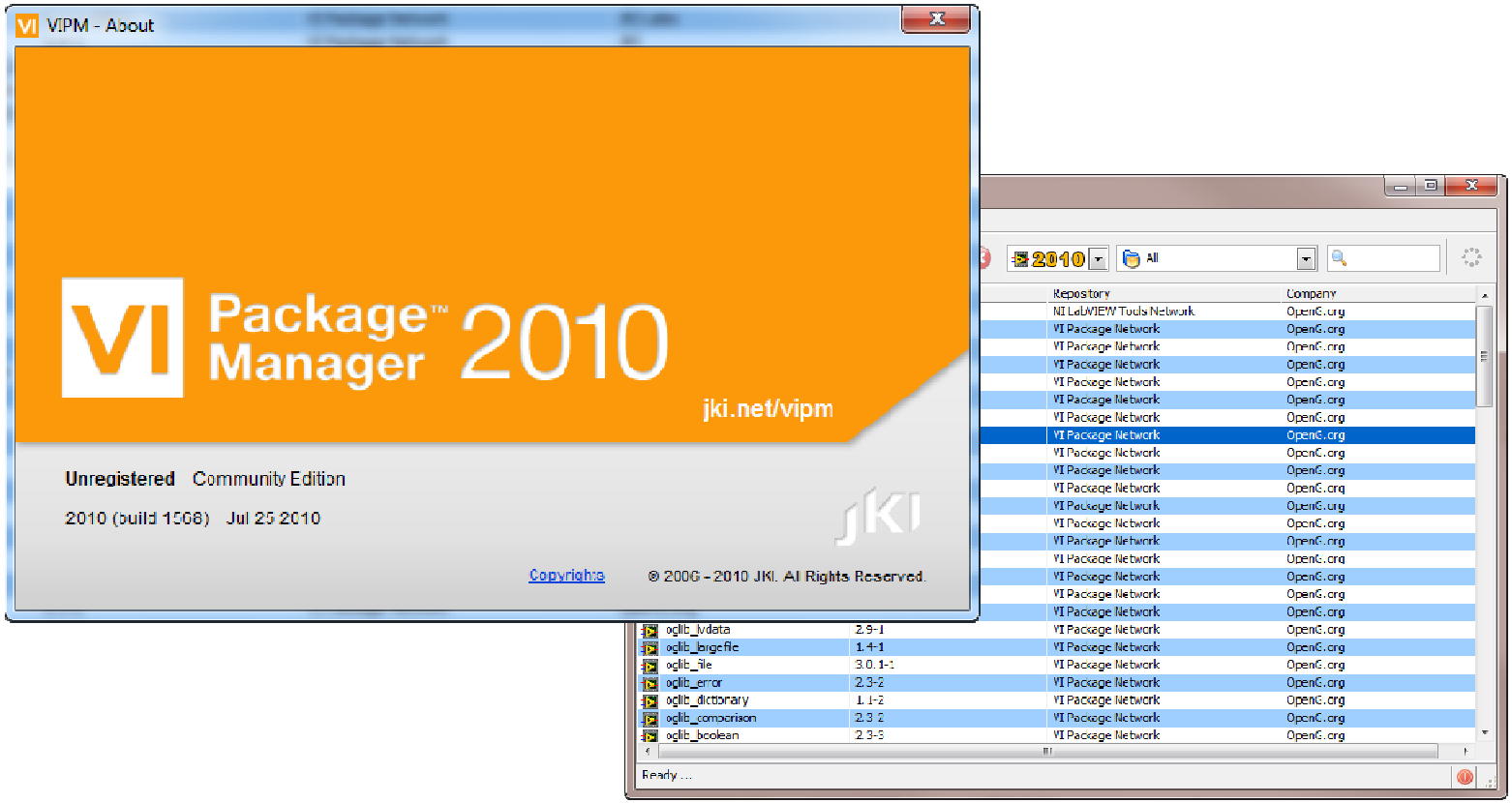
Distribute and Deploy LabVIEW Libraries as a Single File



- Decrease build times
- Deploy the VI hierarchy with a single file
- Simplified code deployment

NATIONAL INSTRUMENTS

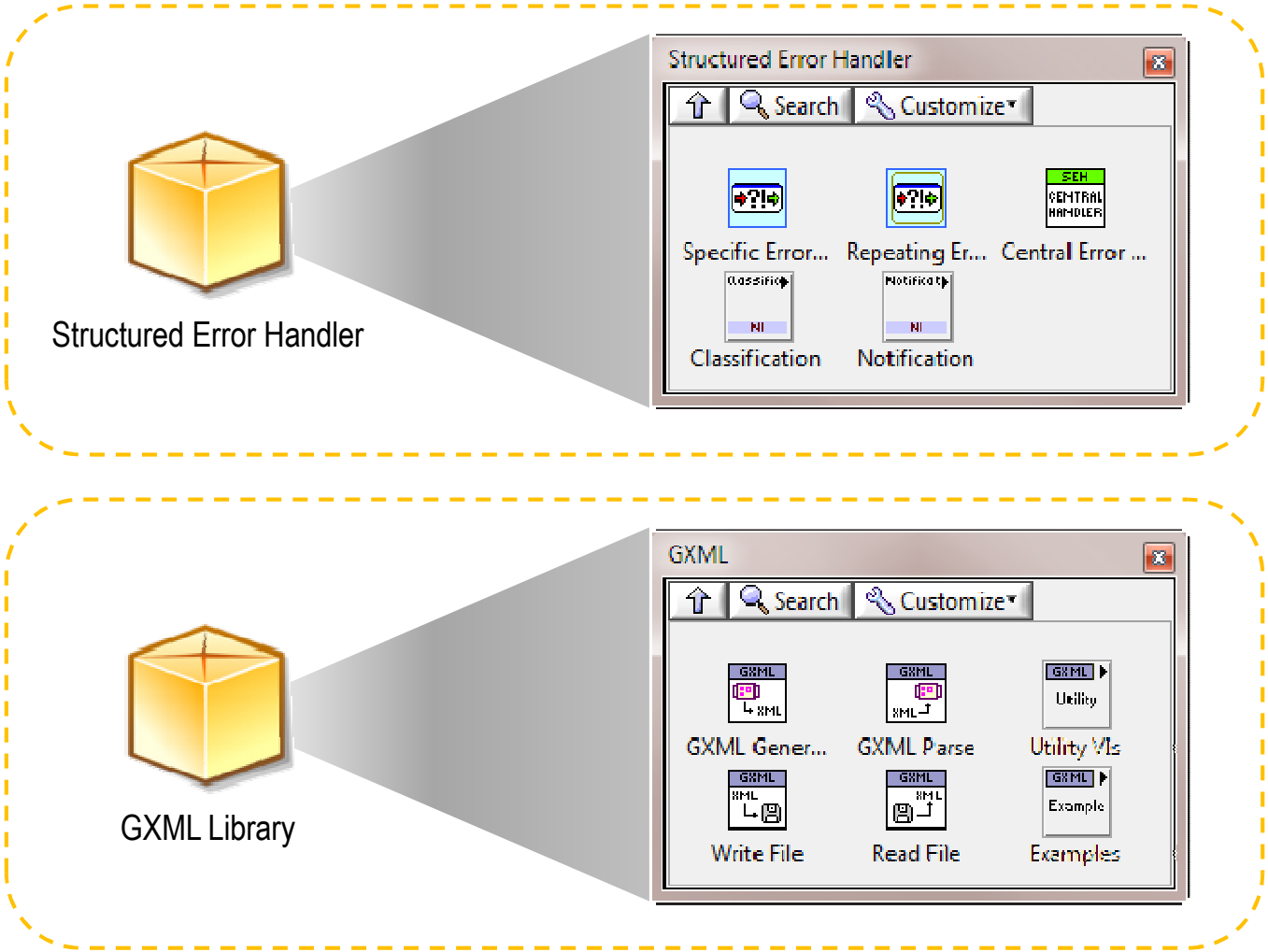
VI Package Manager



Build and manage packages of LabVIEW code

NATIONAL INSTRUMENTS

Install and Manage VI Packages

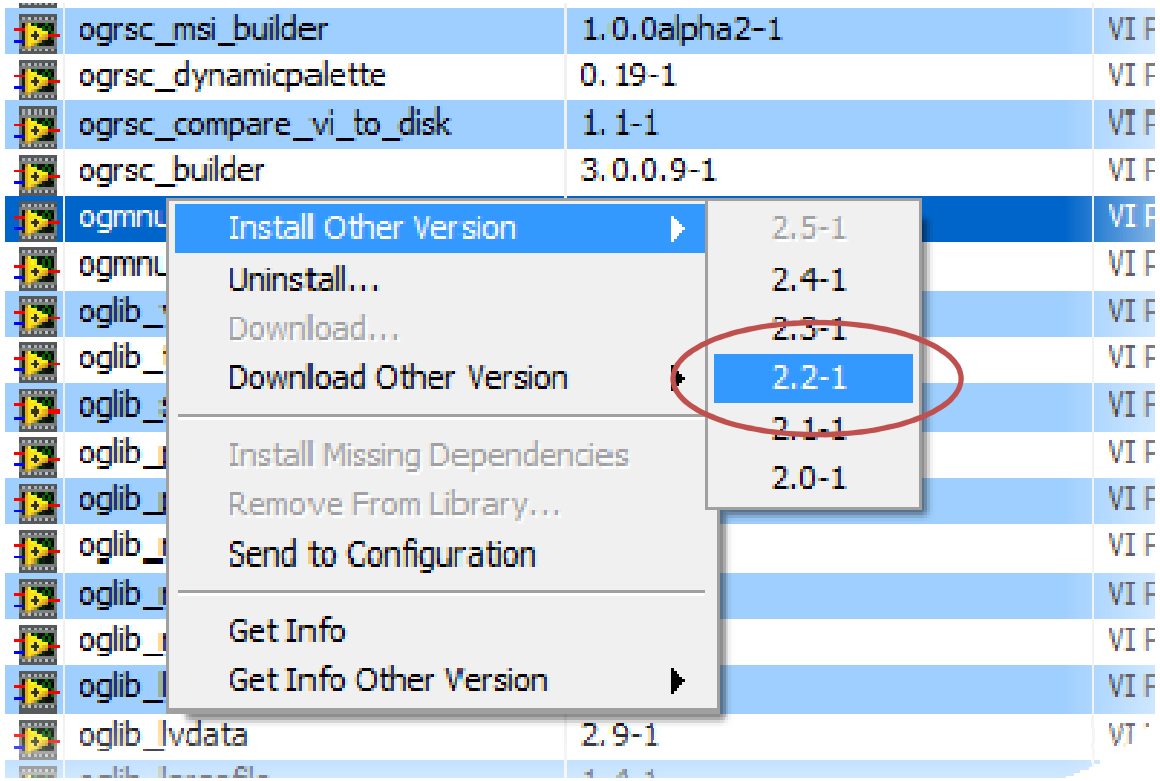


NATIONAL INSTRUMENTS

Introduction to the VI Package Manager

DEMO

Easily Upgrade and Downgrade Versions

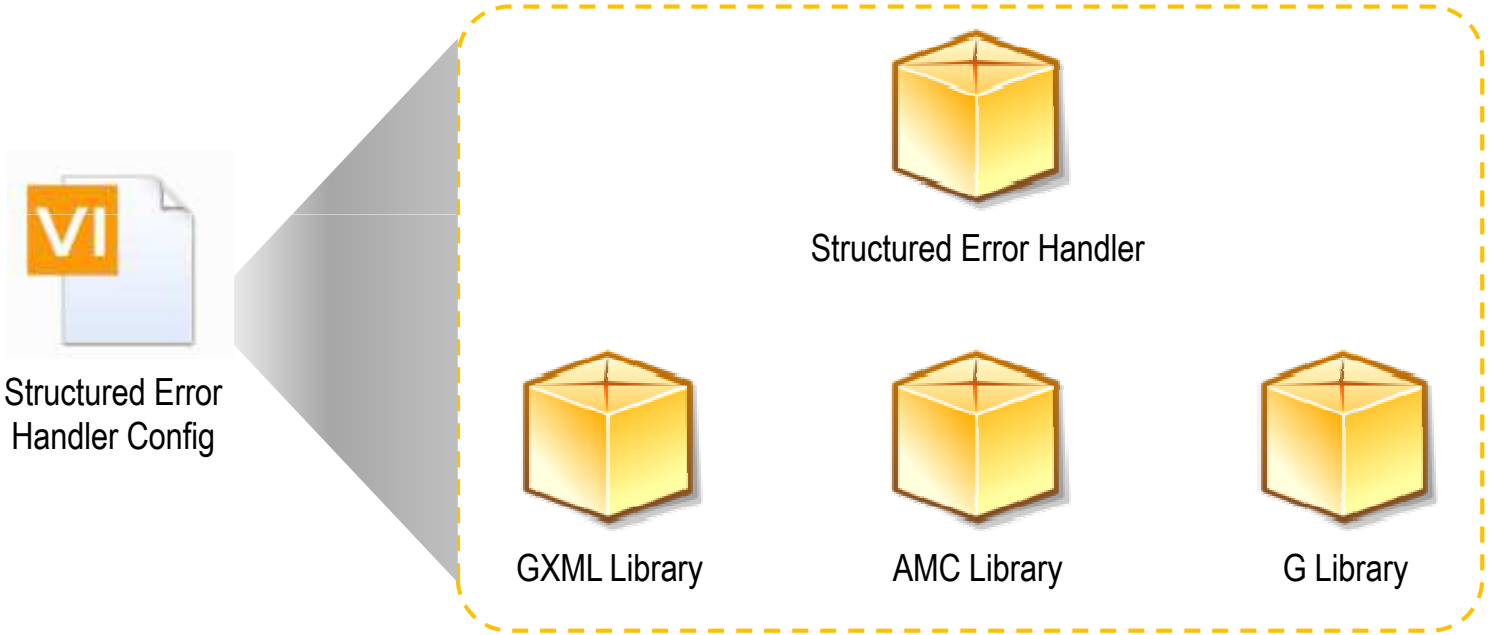


NATIONAL INSTRUMENTS

Create VI Configuration Files

A single file that contains multiple packages.

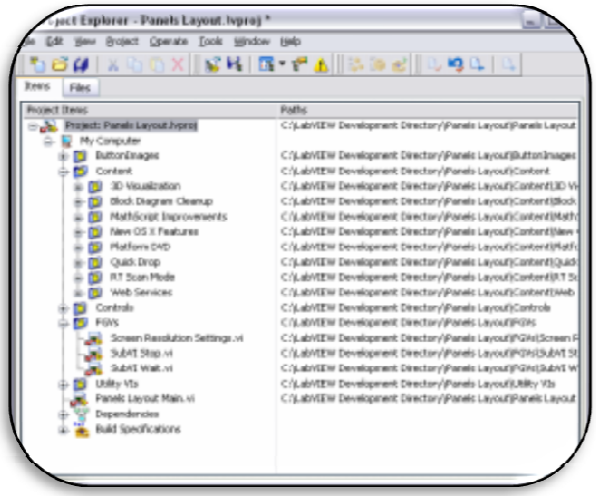
Easily share and distribute code that depends upon multiple libraries.



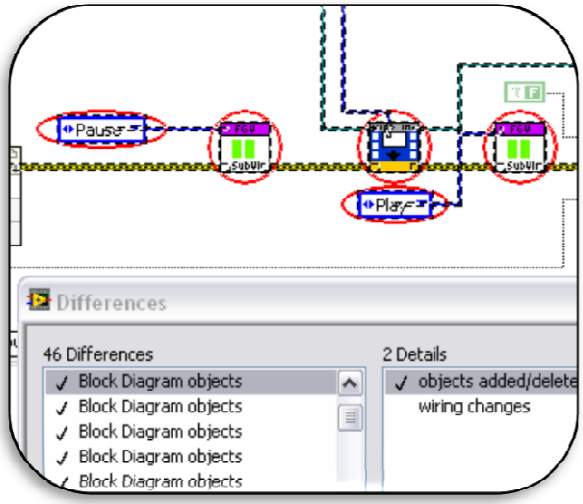
Create a Configuration File by Scanning a Project

DEMO

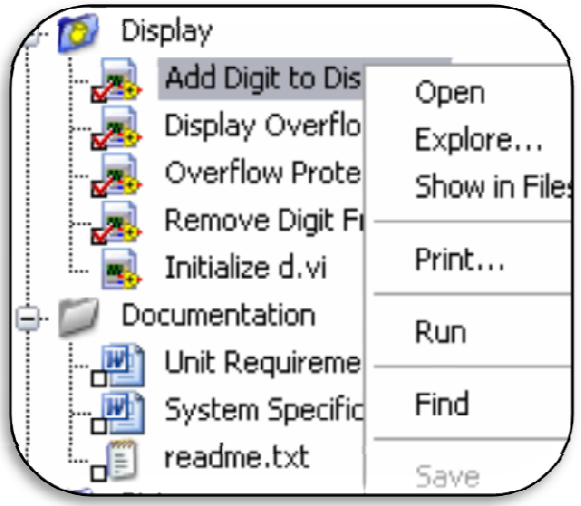
System Level View



Track Changes

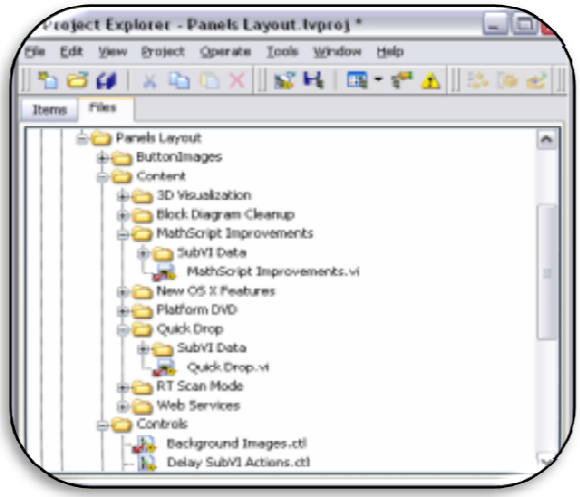


Integrate with SCC

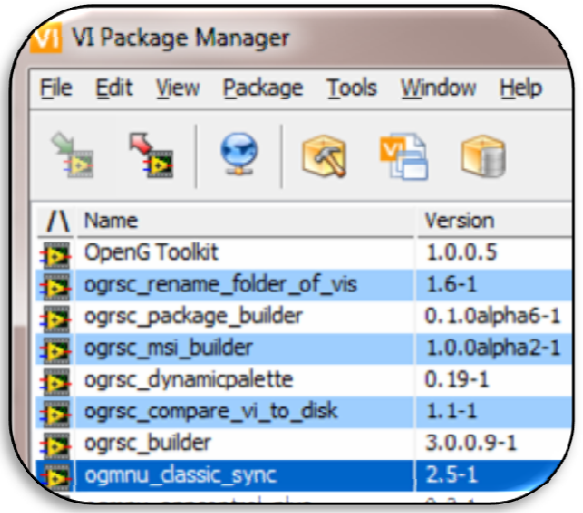


Software Configuration Management for LabVIEW

Merge Graphical Code



Manage Files and Links



Manage Reuse Libraries

Section Two

Software Development Best Practices

NATIONAL INSTRUMENTS

Software Development Risk Continues to Rise

GE Energy acknowledges blackout bug

Anick Jesdanun, The Associated Press 2004-02-12

A programming error has been identified as the cause of alarm failures that might have contributed to the scope of last summer's Northeast blackout, industry officials said Thursday.

Ralph DiNicola, spokesman for FirstEnergy Corp., said the utility has since applied fixes developed by the system's vendor, General Electric Co., and has accelerated plans to replace GE's

A U.S.-Canad steps that co

ontrol

Software Errors Cost U.S. Economy $59.5 Billion Annually

NIST Assesses Technical Needs of Industry to Improve Software-Testing

FOR IMMEDIATE RELEASE

rubble across the mangrove swamps of French Guiana, was a small computer program trying to stuff a 64-bit number into a 16-bit space.

One bug, one crash. Of all the careless lines of code recorded in the annals of computer science, this one may stand as the most devastatingly efficient. From interviews with rocketry experts and an analysis prepared for the space agency, a clear path from an arithmetic error to total destruction emerges.

To play the tape backward:

At 39 seconds per launch as the rocket reached an altitude of two and a half miles, a self-destr

NATIONAL INSTRUMENTS

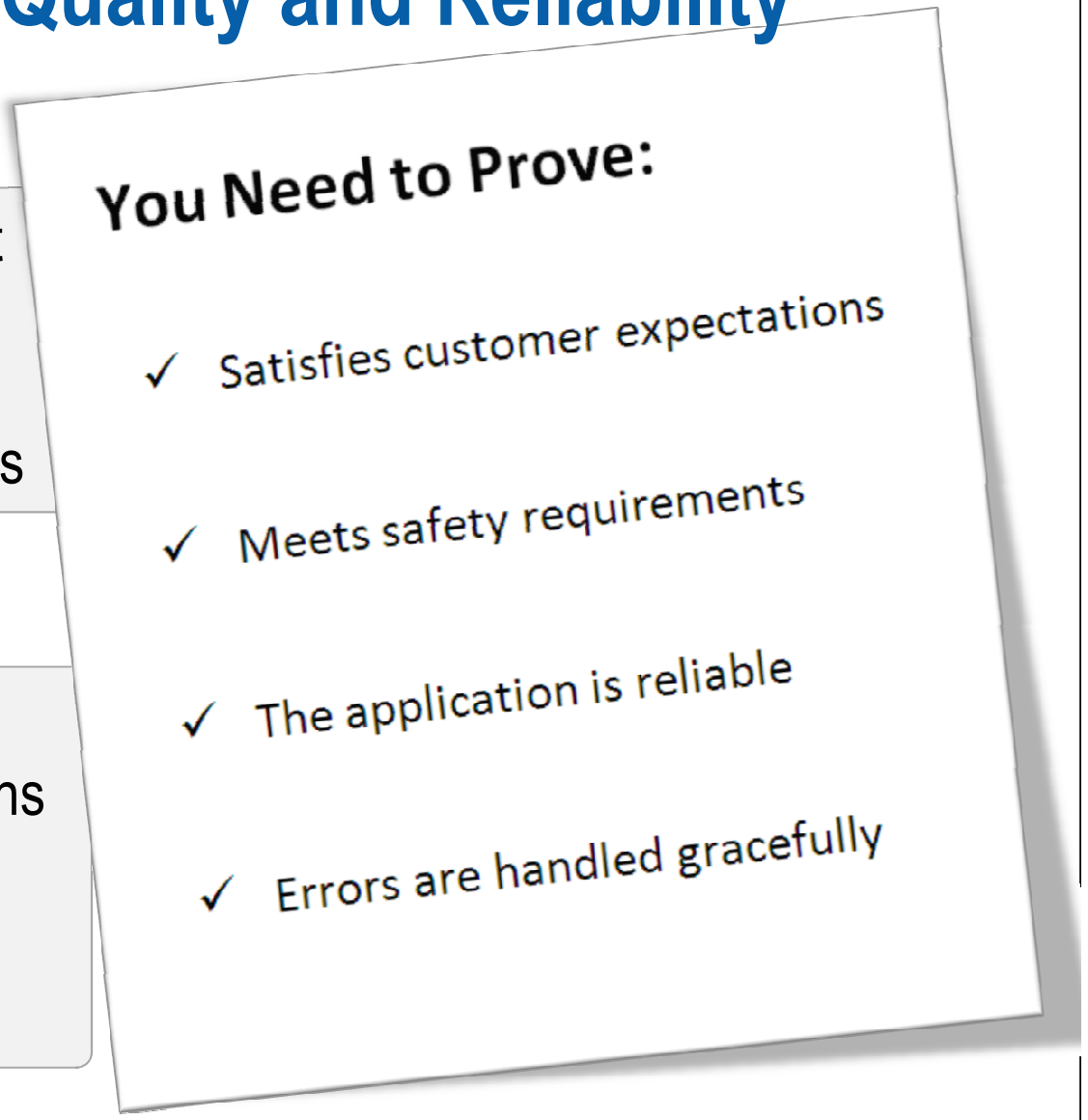
Ensuring Software Quality and Reliability

Goals

1. Deliver a working product
2. Prove it works right
3. Mitigate risk of failure
4. Avoid last-minute changes

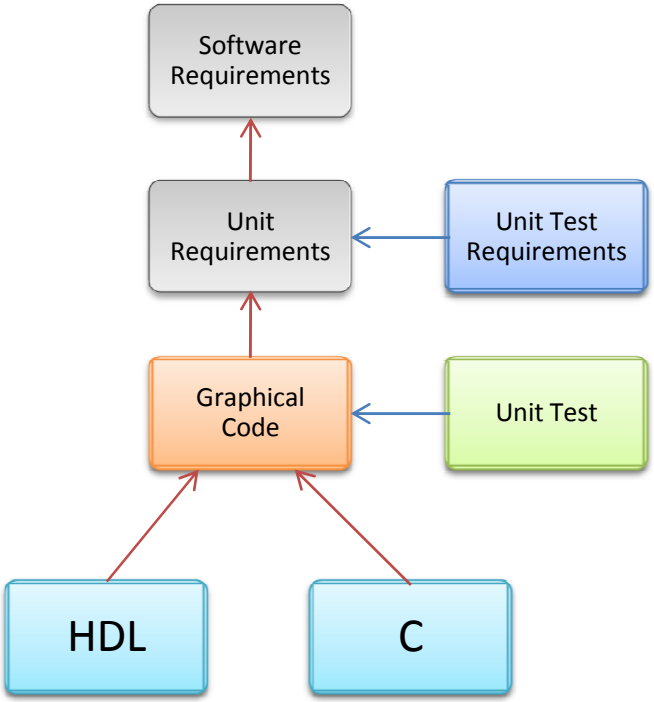
Why?

1. More complex software
2. Mission-critical applications
3. Team size is growing
4. Increased scrutiny
5. Decreased time



NATIONAL INSTRUMENTS

Certifying an Application



Have the software requirements been implemented?
We did what we were supposed to.

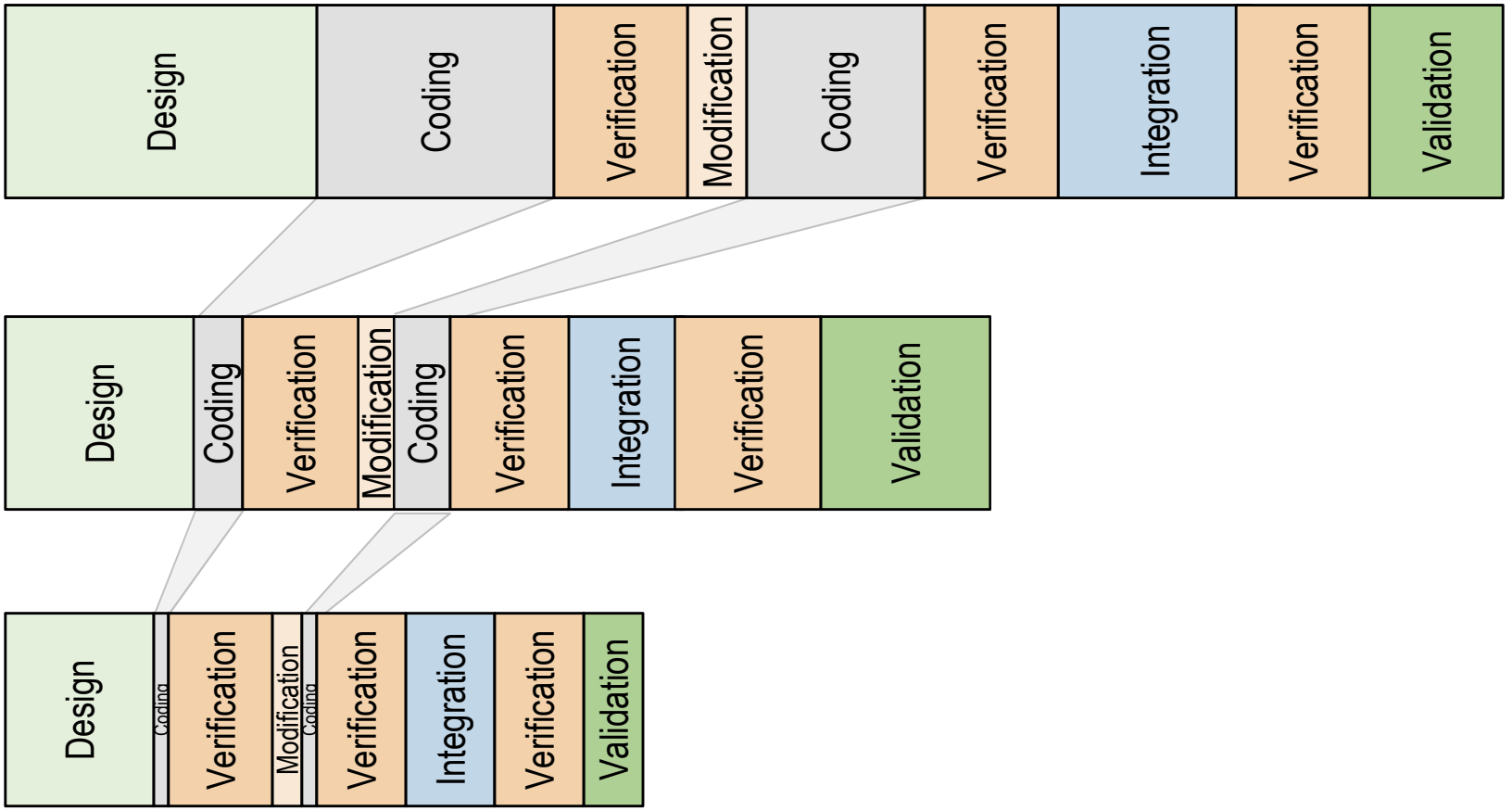
Have the test cases been implemented?
We tested what we were supposed to.

Has the code passed the functional tests?
The implementation is correct.

Have the tests exercised the required percentage of code? *Code has been sufficiently covered in testing.*

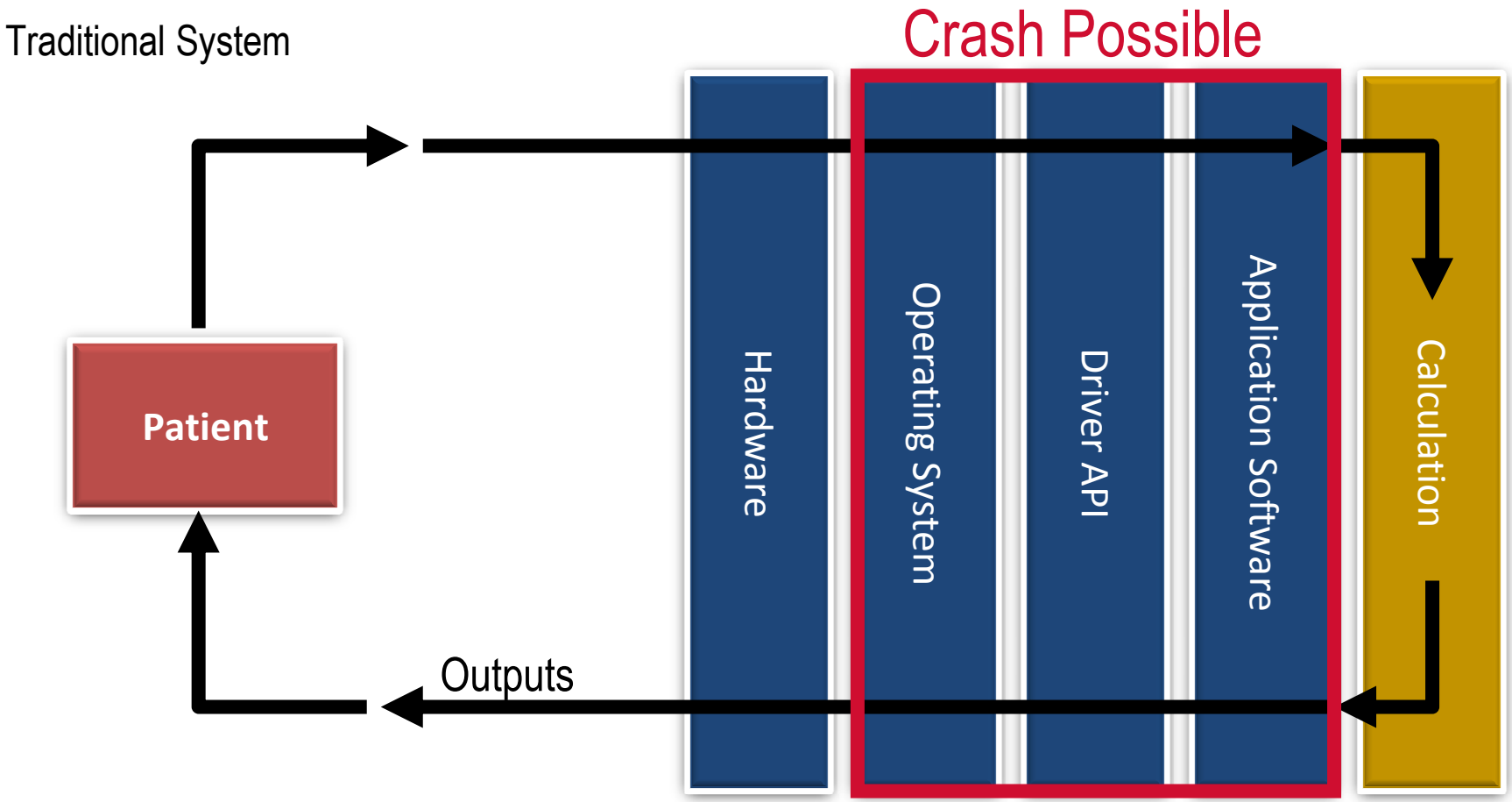
Has the code passed static analysis standards?
The implementation is standard.

More Overhead - Less Time



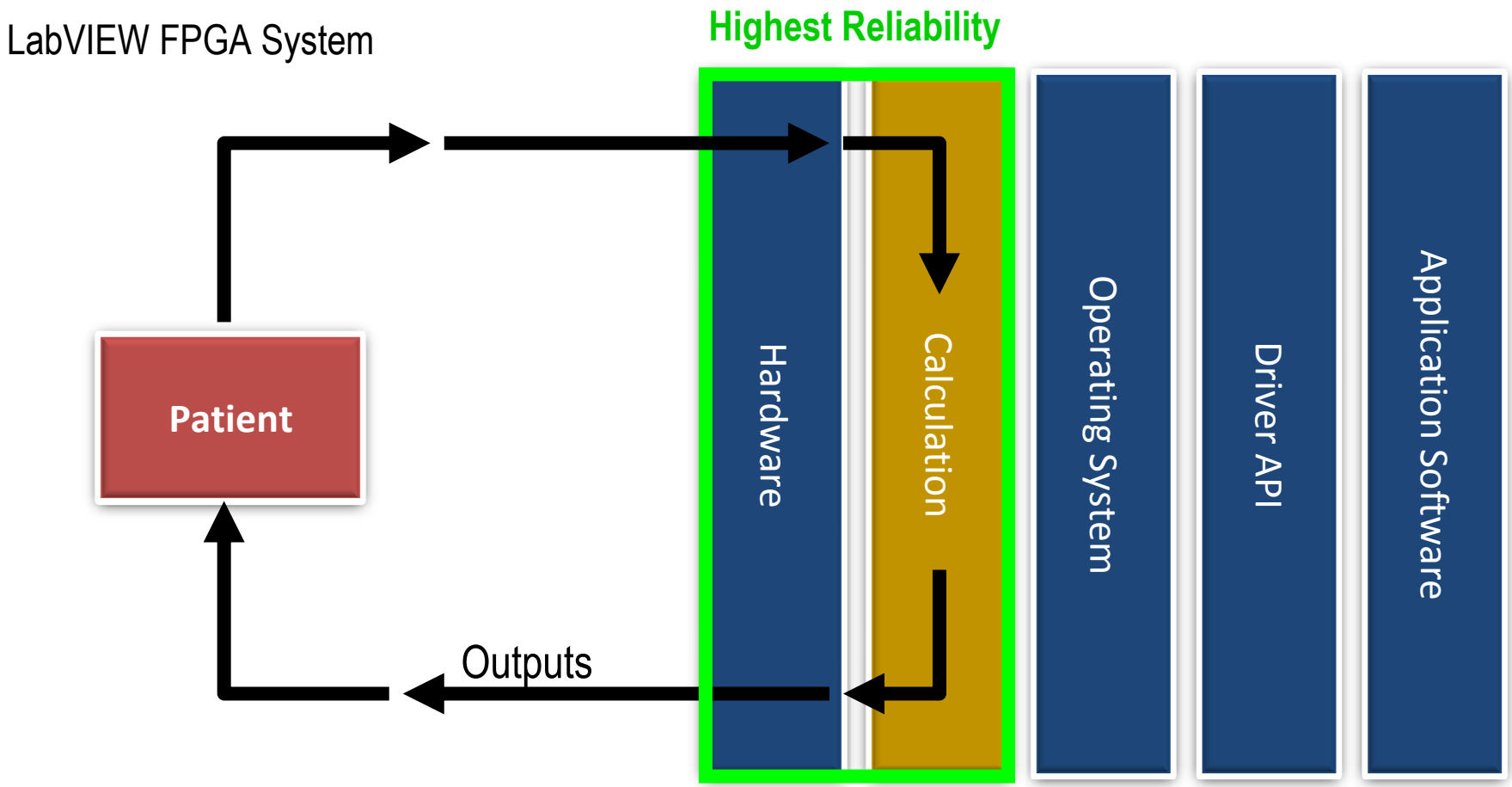
NATIONAL INSTRUMENTS

Mitigating Risk Through Hardware



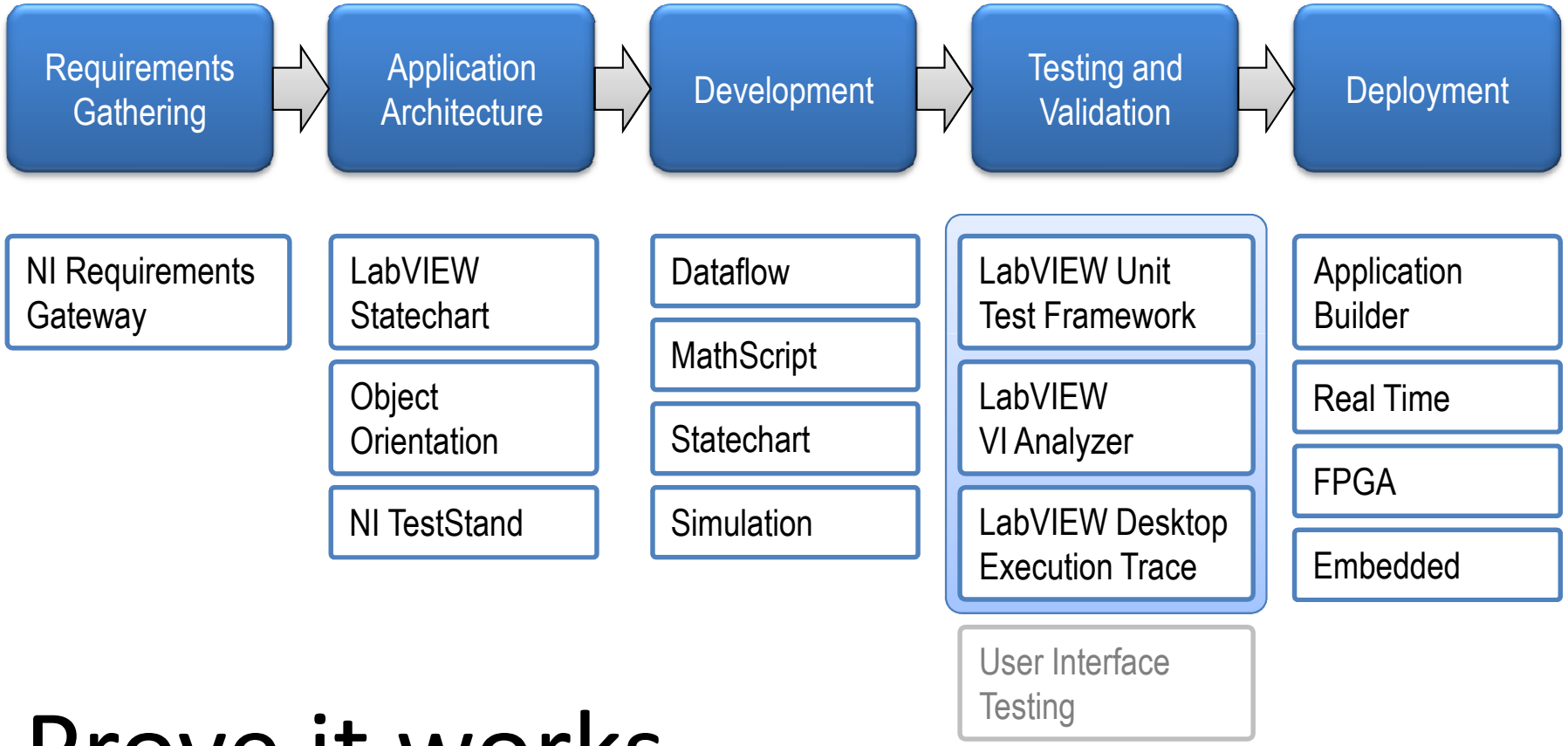
NATIONAL INSTRUMENTS

Mitigating Risk Through Hardware



NATIONAL INSTRUMENTS

The Software Engineering Process

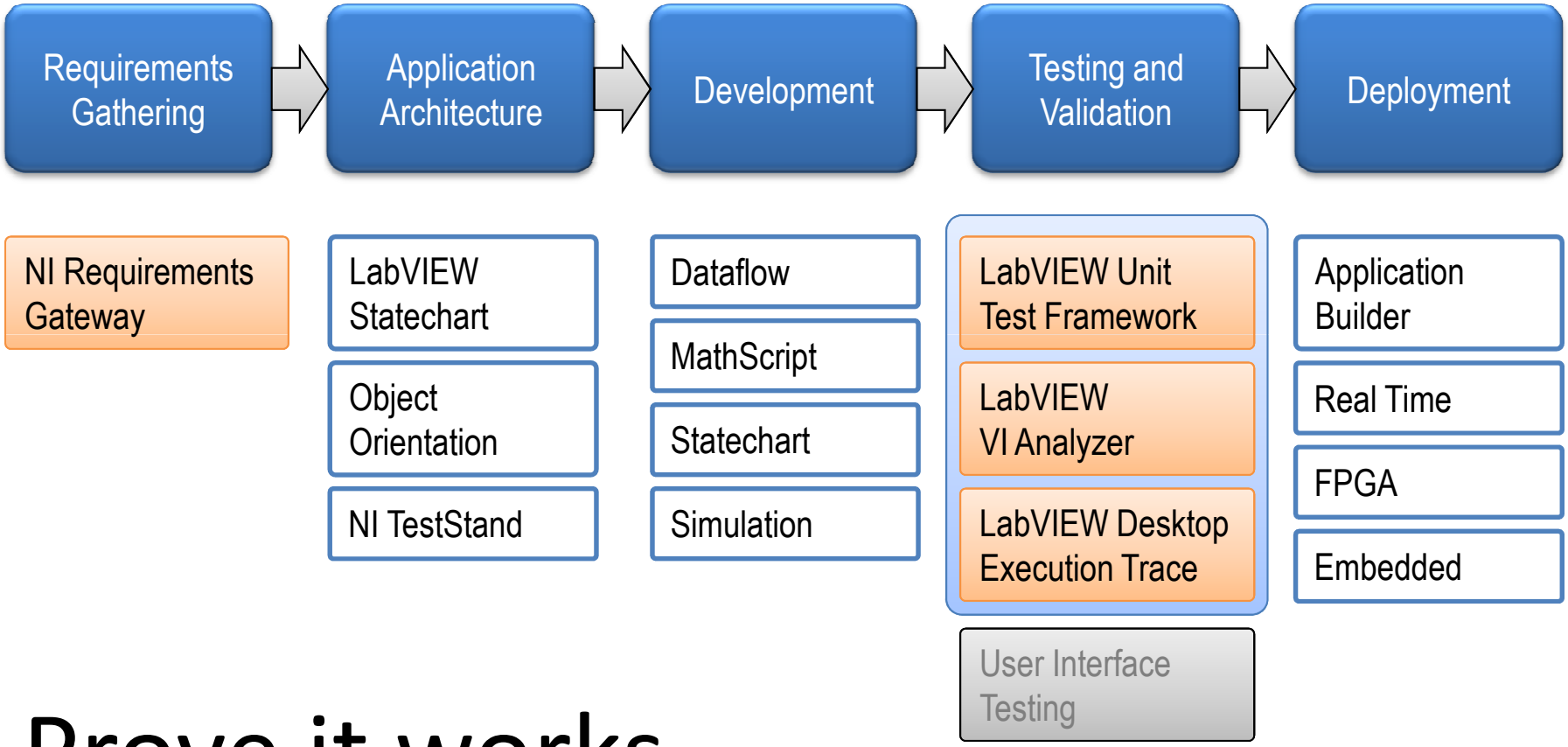


Prove it works.

Improve quality. Reduce risk. Save time.

NATIONAL INSTRUMENTS

The Software Engineering Process

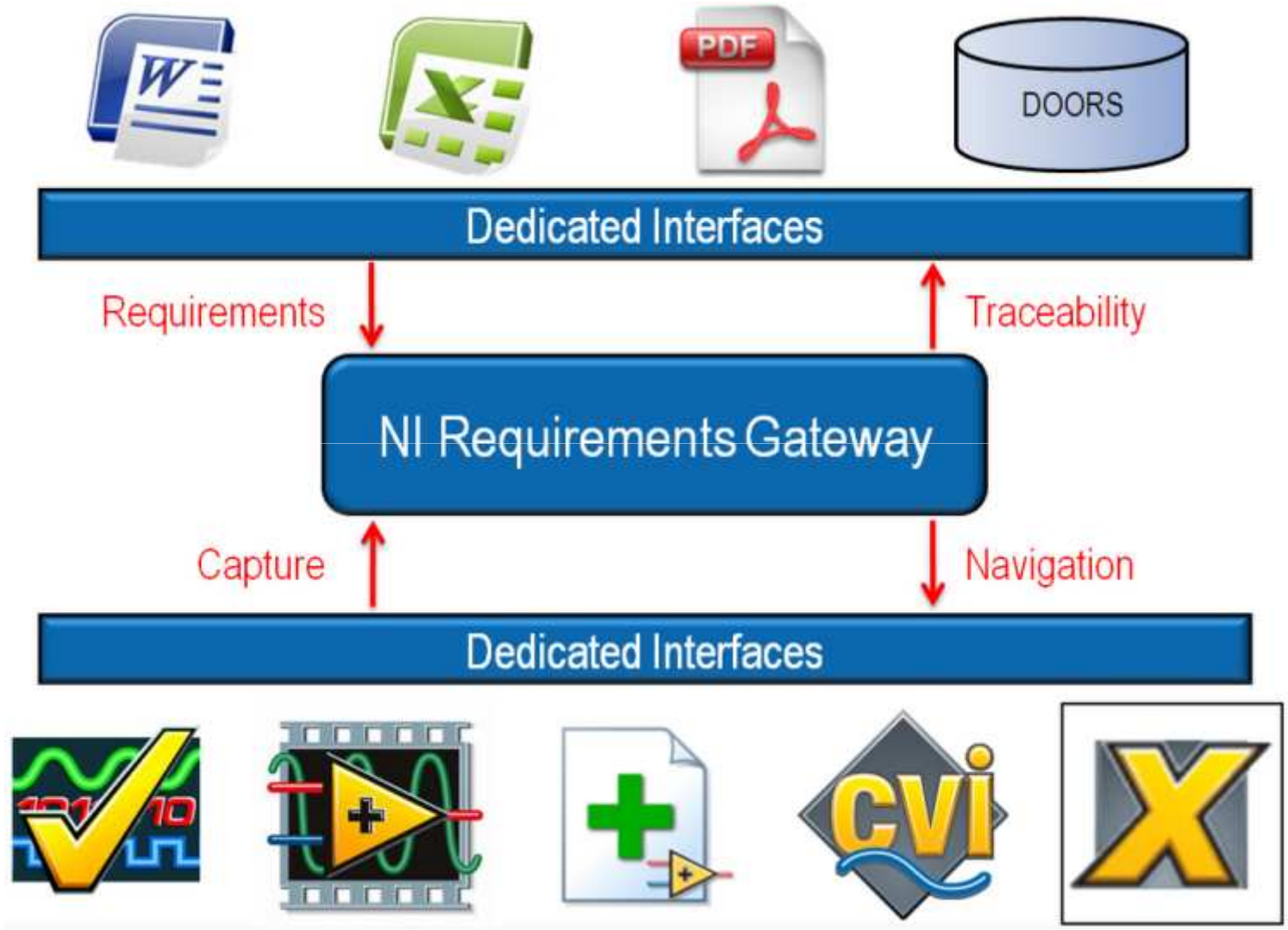


Prove it works.

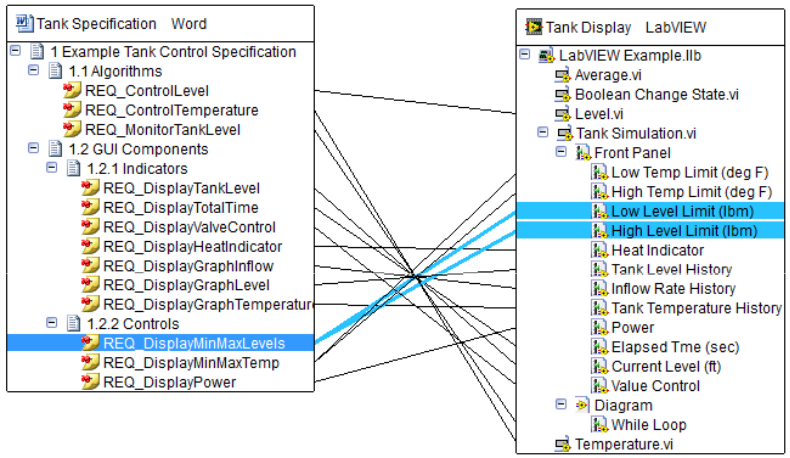
Improve quality. Reduce risk. Save time.

NATIONAL INSTRUMENTS

Requirements Traceability Solution from NI



NATIONAL INSTRUMENTS



Requirements Coverage and Project Tracking

Upstream	Text	Downstream
REQ_ControlLevel	The application must maintain a separate control algorithm for controlling tank levels.	Level.vi
REQ_ControlTemperature	The application must maintain a separate control algorithm for controlling temperature.	Temperature.vi
REQ_DisplayGraphInflow	The display must display a graph indicating the inflow rate.	Inflow Rate History
REQ_DisplayGraphLevel	The display must contains a graph indicating the tank level.	Tank Level History
REQ_DisplayGraphTemperature	The display must contains a graph indicating the tank temperature.	Tank Temperature History
REQ_DisplayHeatIndicator	The display must show whether furnace is on or off.	Heat Indicator
REQ_DisplayMinMaxLevels	The display must allow the operator to control the minimum and maximum levels for the tank.	Low Level Limit (lbm)
REQ_DisplayMinMaxLevels	The display must allow the operator to control the minimum and maximum levels for the tank.	High Level Limit (lbm)
REQ_DisplayMinMaxTemp	The display must allow the operator to control the minimum and maximum levels for the tank temperature.	Low Temp Limit (deg F)

Traceability and Documentation Generation

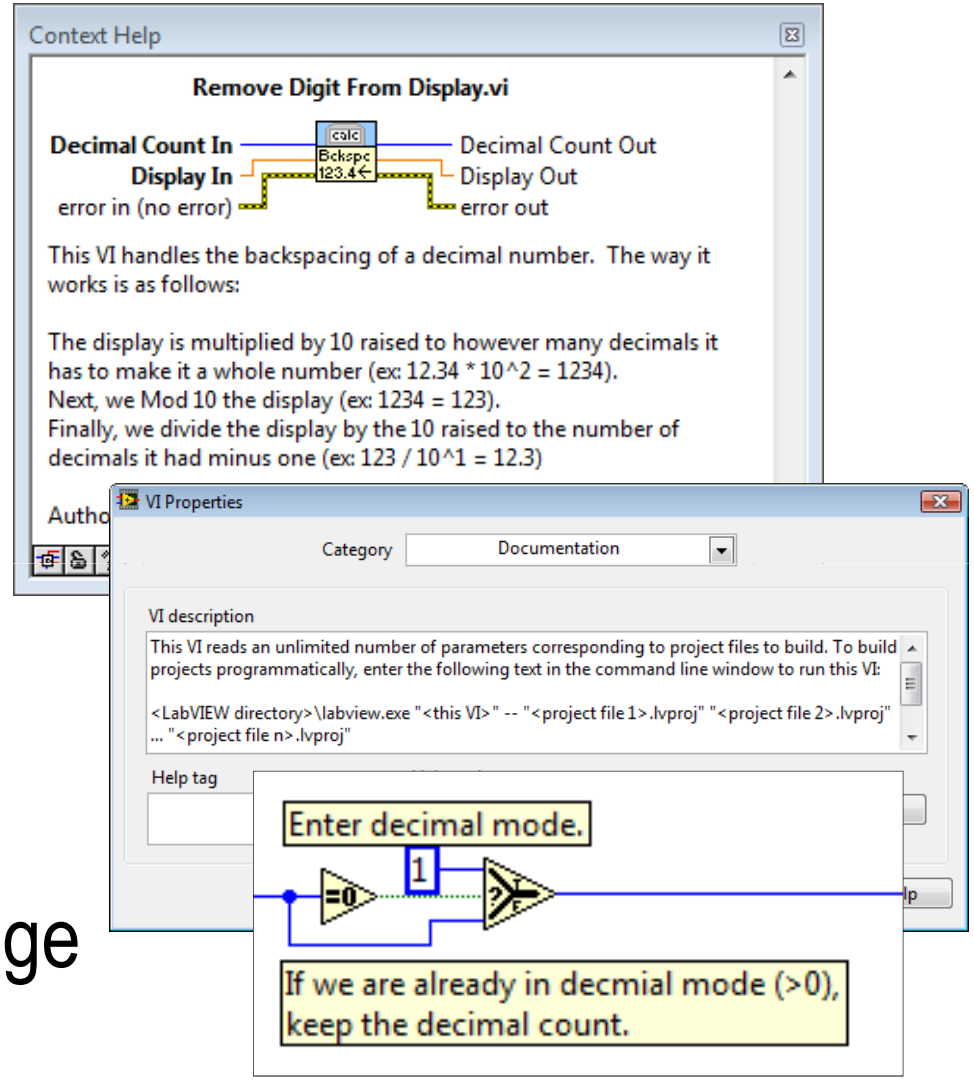
NATIONAL INSTRUMENTS

Requirements Tracking

DEMO

Documentation

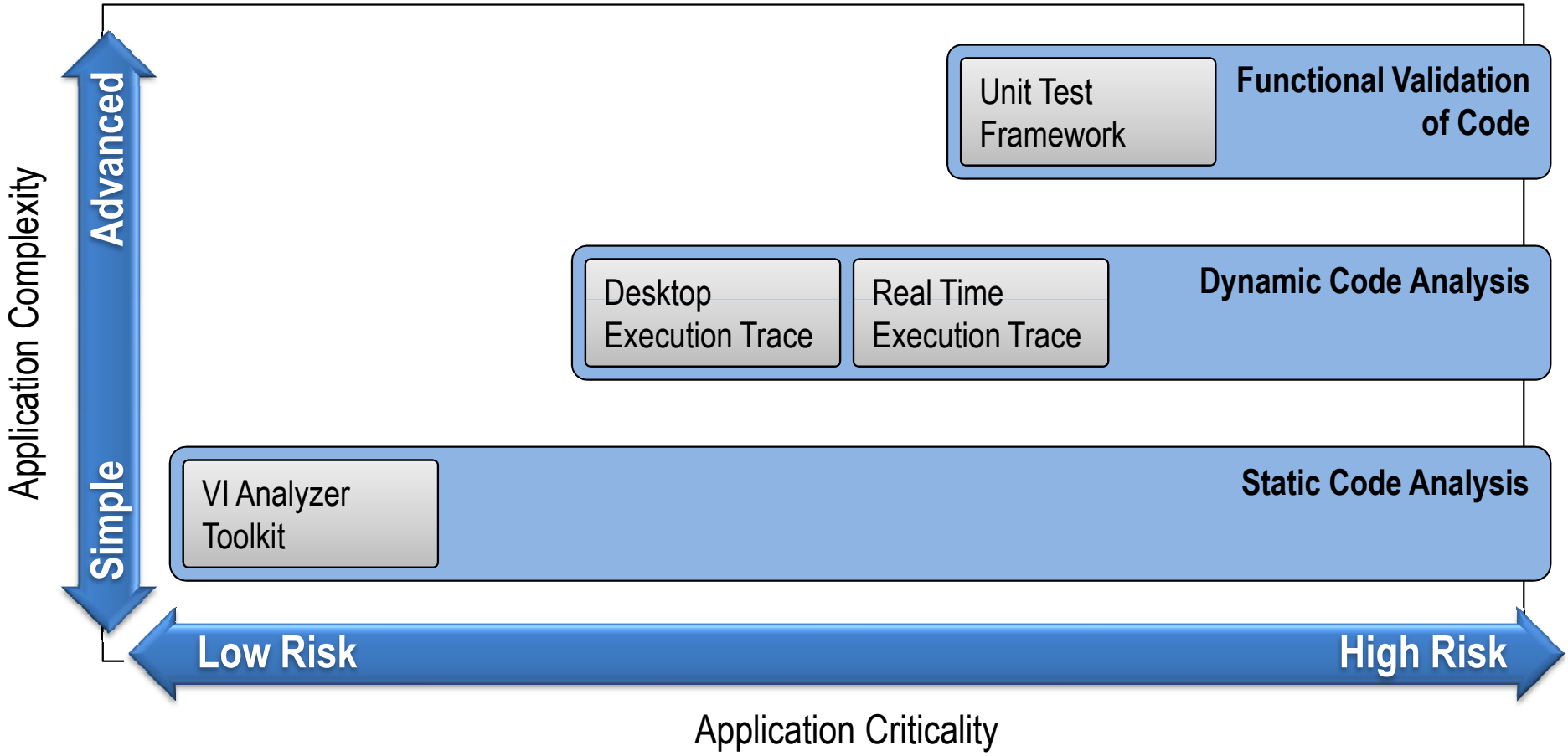
- Labeled objects
- In-line comments
- Distinct Icon
- Description and Tip
- HTML Print-Out
- Requirements Coverage



NATIONAL INSTRUMENTS

Tools for Debugging and Testing

Debugging & Testing



NATIONAL INSTRUMENTS

Performing a LabVIEW Code Review

- Walk someone through your code
- Questions to consider:
  - Is the code easy to maintain, and has it been documented?
  - What happens if the code returns an error?

    (… or if it receives an error?)
  - Is too much functionality located in a single VI?
  - Are there any race conditions?
  - Is the memory usage within acceptable limits?
- Perform code reviews frequently

NATIONAL INSTRUMENTS

Establish or Adopt Development Guidelines

Front Panel Style

- Fonts and Text Characteristics
- Colors
- Graphics and Custom Controls
- Layout
- Sizing and Positioning
- Labels
- Paths versus Strings
- Enumerated Type Controls versus Ring Controls
- Default Values and Ranges
- Property Nodes
- Key Navigation
- Dialog Boxes

Style Checklist

- VI Checklist
- Front Panel Checklist
- Block Diagram Checklist

Block Diagram Style

- Wiring Techniques
- Memory and Speed Optimization
- Sizing and Positioning
- Left-to-Right Layouts
- Block Diagram Comments
- Call Library Function Nodes and Code Interface Nodes
- Type Definitions
- Sequence Structures

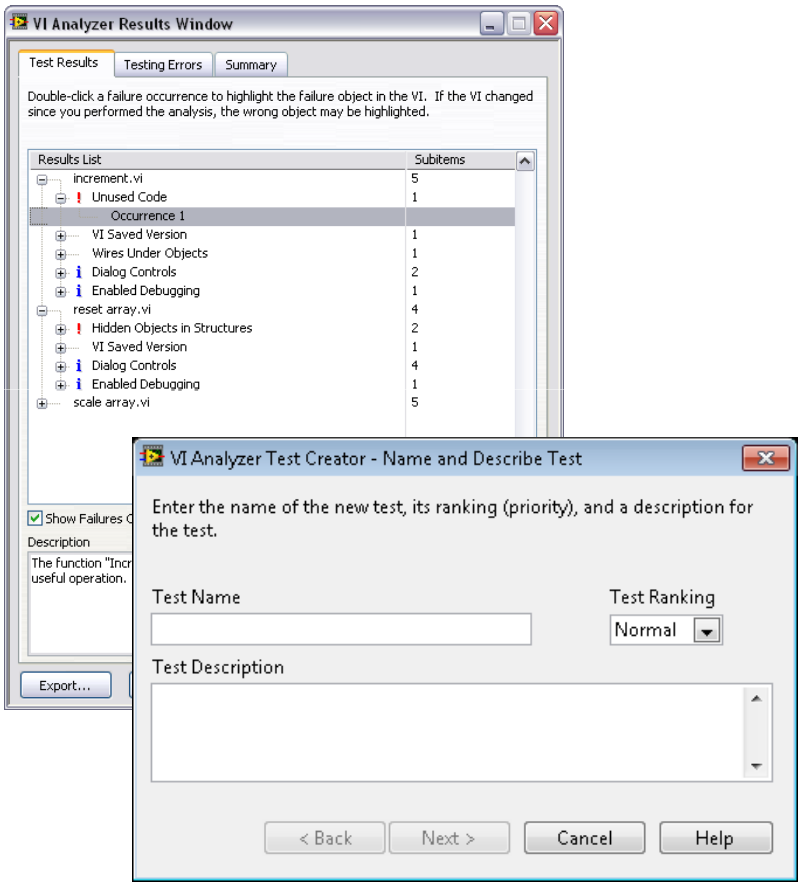
Icon and Connector Pane Style

- Icons
- Example of Intuitive Icons
- Connector Panes

NI Style Guideline: ni.com/largeapps

Preparing for a Code Review with VI Analyzer

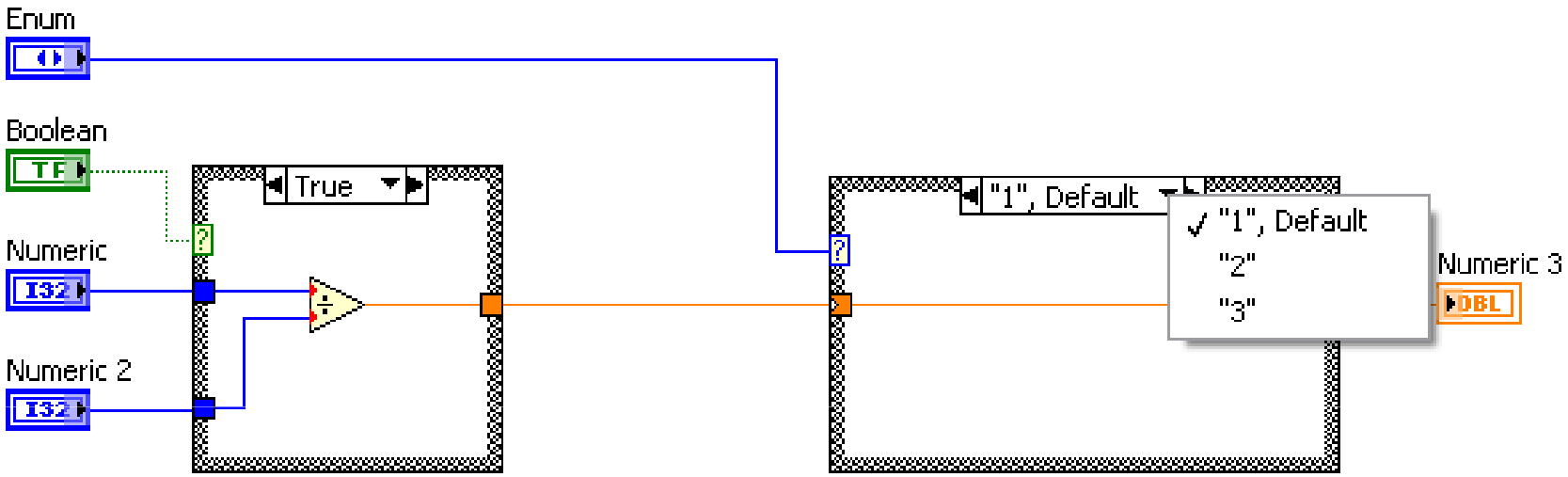
- Automate code analysis with 80+ configurable tests
  - Performance
  - Style
  - Complexity
- Interactively inspect failures
- Generate custom reports
- Code complexity metrics
- Write your own tests with VI Scripting ***LabVIEW 2010***



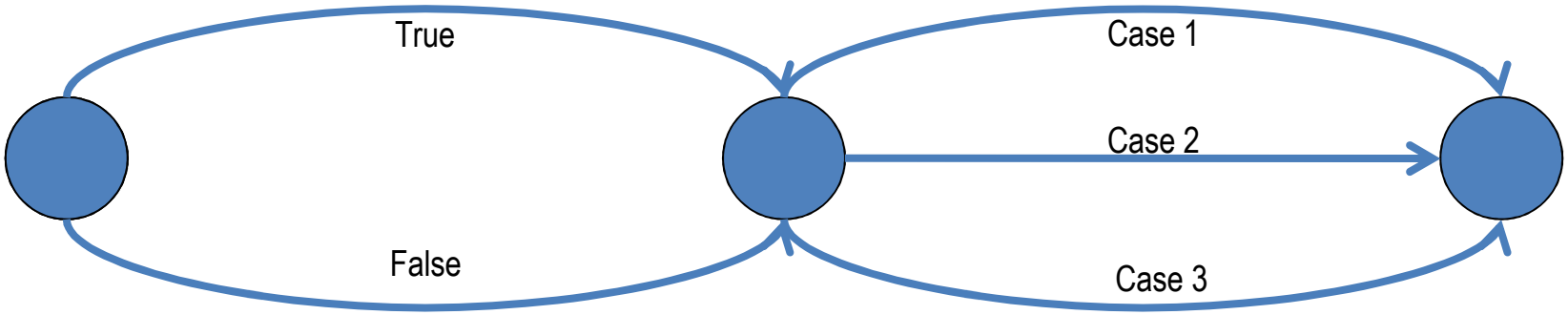
NATIONAL INSTRUMENTS

VI Analyzer Code Complexity Metrics

Tests for Industry Standard Metric Calculations



Edges = 5 Nodes = 3 Paths = 1



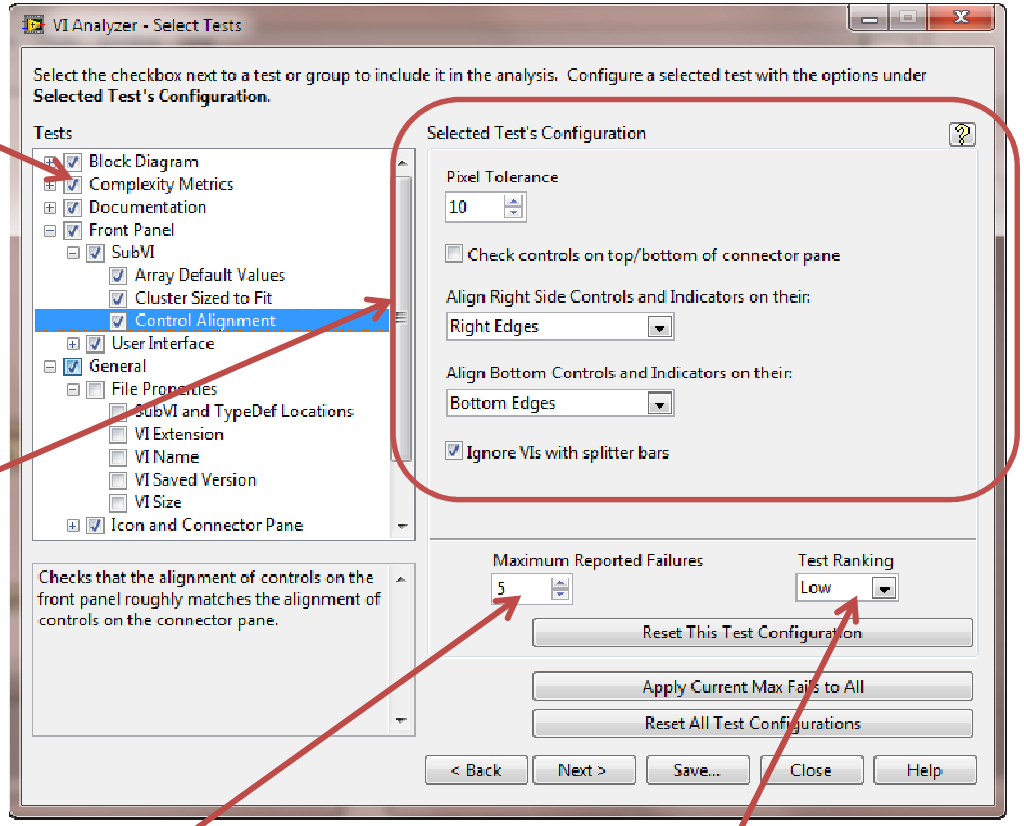
Cyclomatic complexity: 5 – 3 + 2 (1) = 4

NATIONAL INSTRUMENTS

Creating Custom Tests

Select tests to run

Test specific configuration settings

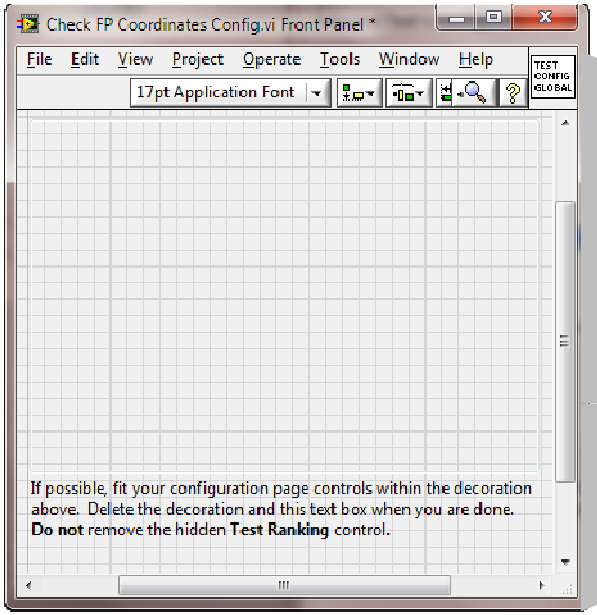


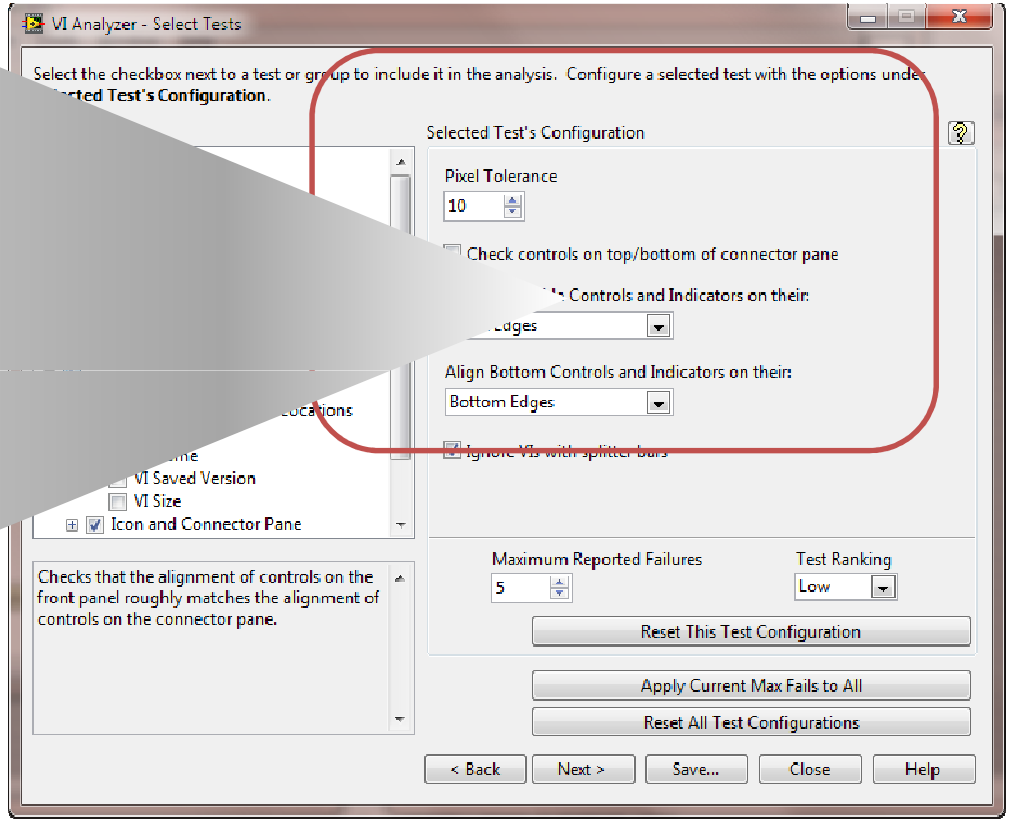
Number of failures to report

Test Priority

NATIONAL INSTRUMENTS

Define Configuration Options





Access values in test code using Global variables

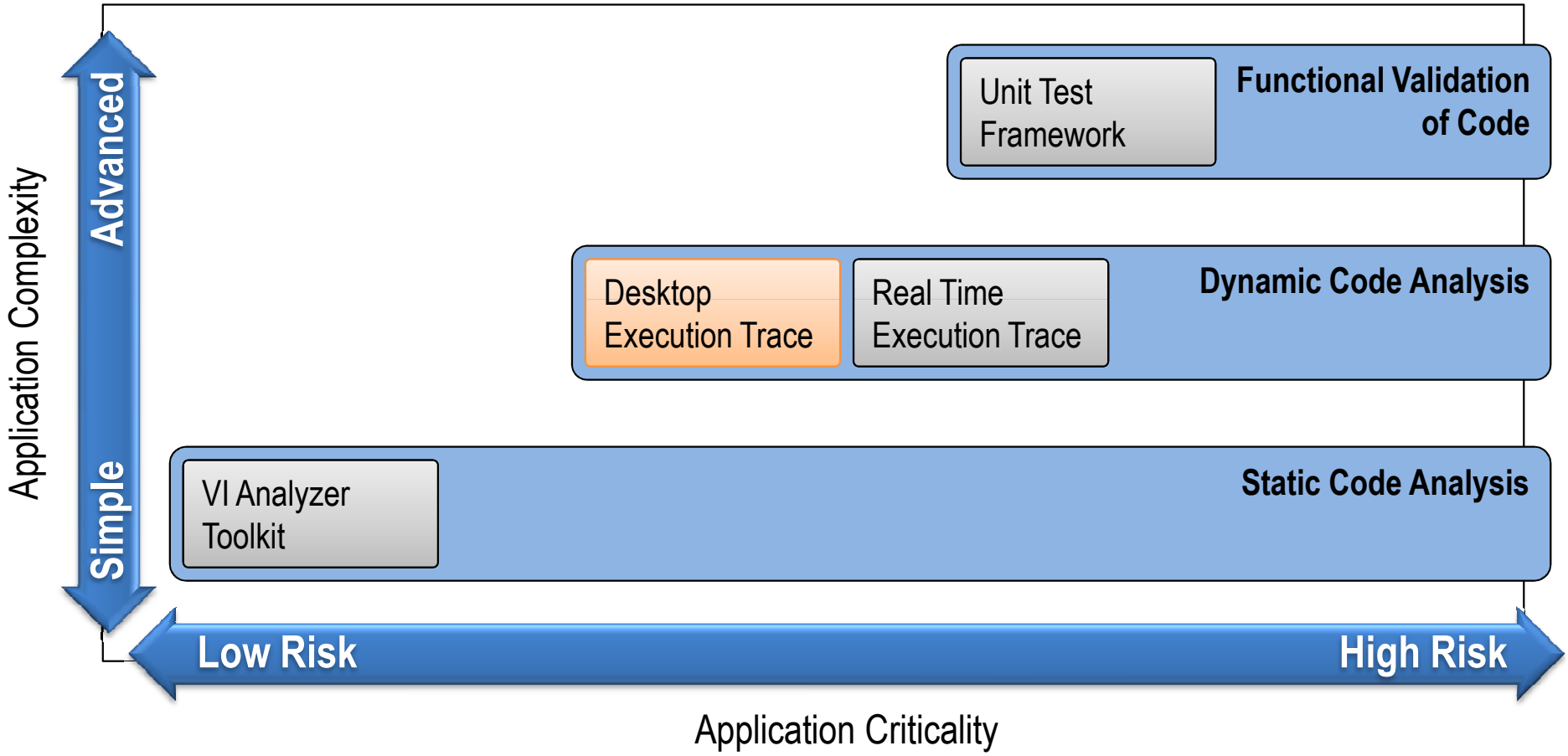
NATIONAL INSTRUMENTS

VI Analyzer for Peer Reviews

DEMO

Tools for Debugging and Testing

Debugging & Testing



NATIONAL INSTRUMENTS

Goals of Dynamic Code Analysis:

- What is consuming system memory?
- Am I capturing all the errors in my application?
- What was the last event to occur before…?
- What was the call-chain that led us to…?
- What thread is it executing in?
- Am I actually entering a specific event-case?
- What happened inside a structure?
- What order to these events occur in?
- Is a daemon process running in the background?
- Does the code behave different in an executable?

Trace Production Systems Remotely

LabVIEW Desktop Execution Trace Toolkit

Run-Time Execution Information

Network

VIs and Debuggable Executables

LabVIEW Real-Time Execution Trace Toolkit

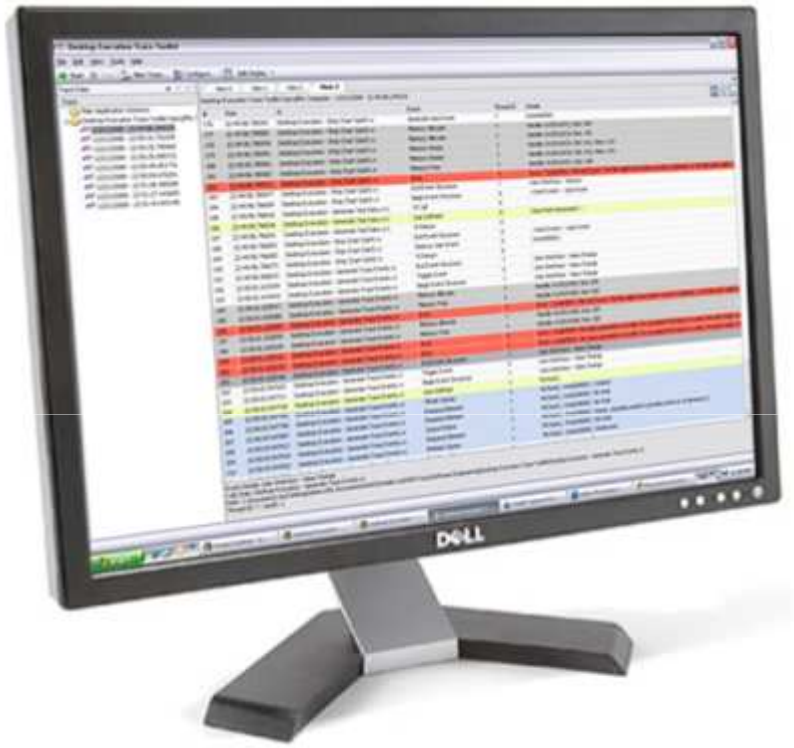
Run-Time Execution Information

Network

Deployed Real-Time Applications

NATIONAL INSTRUMENTS

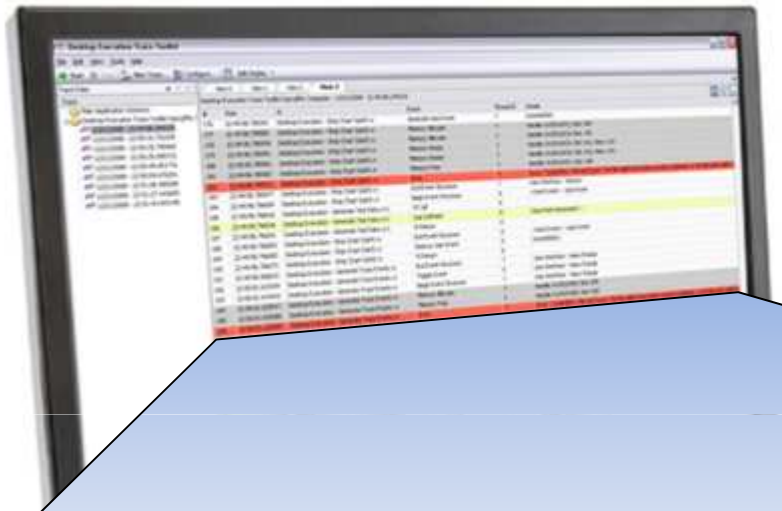
Desktop Execution Trace Toolkit



Trace During Run-Time:

- Event Structures
- Memory Allocation
- Queues / Notifiers
- Reference Leaks
- Thread ID
- Unhandled Errors
- Dynamic / Static SubVIs
- Custom User Strings

Desktop Execution Trace Toolkit



Trace During Run-Time:

- Event Structures
- Memory Allocation
- Queues / Notifiers
- ference Leaks

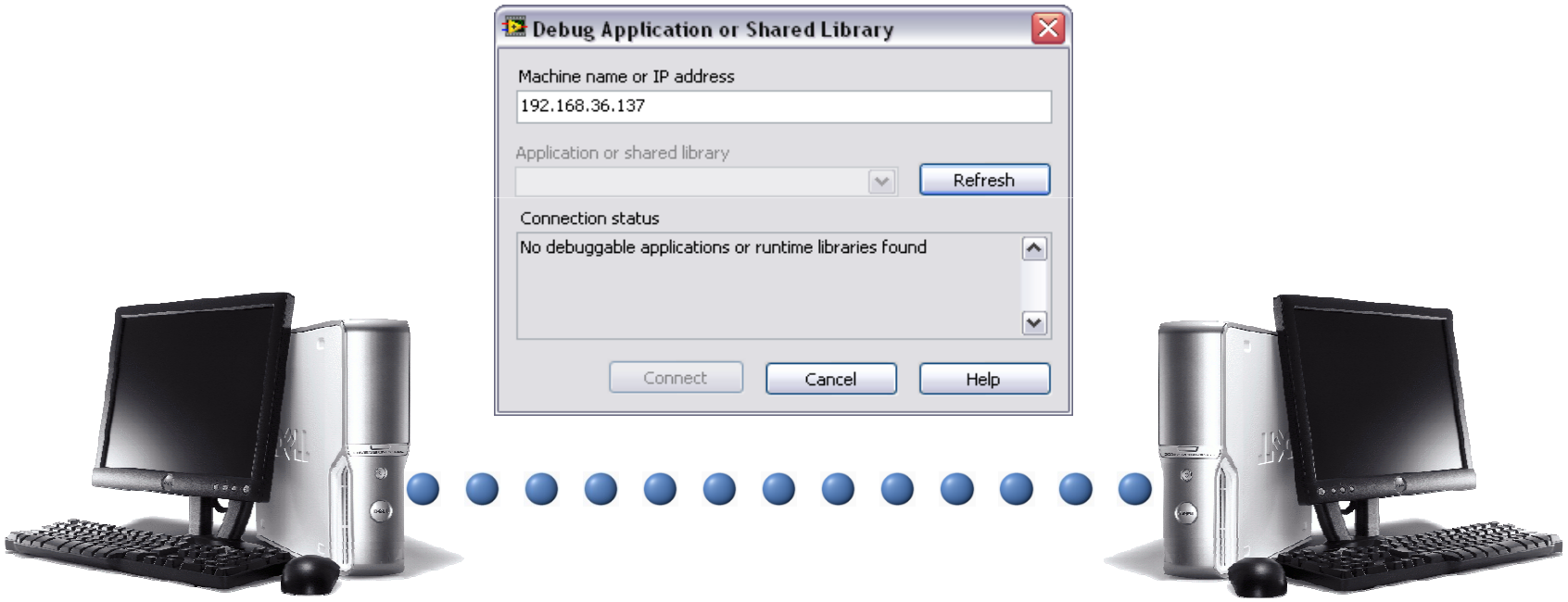
Strip Chart SubVI.vi	Memory Resize	7	Handle: 0x25CA3C8; Old: 142; New: 118
Strip Chart SubVI.vi	Memory Allocate	7	Handle: 0x25CA3C8; Size: 142
Strip Chart SubVI.vi	Memory Resize	7	Handle: 0x25CA3C8; Old: 142; New: 118
Strip Chart SubVI.vi	Error	7	Error: 7 (LabVIEW: File not found. The file might have
Generate Trace Events.vi	User Defined	7	MyTestQ
Generate Trace Events.vi	Obtain Queue	7	MyTestQ - 0x66200002 : Created
Generate Trace Events.vi	Enqueue Element	7	MyTestQ - 0x66200002 : No Wait

Dynamic Code Analysis with Desktop Execution Trace Toolkit

DEMO

Remote Debugging of Executables

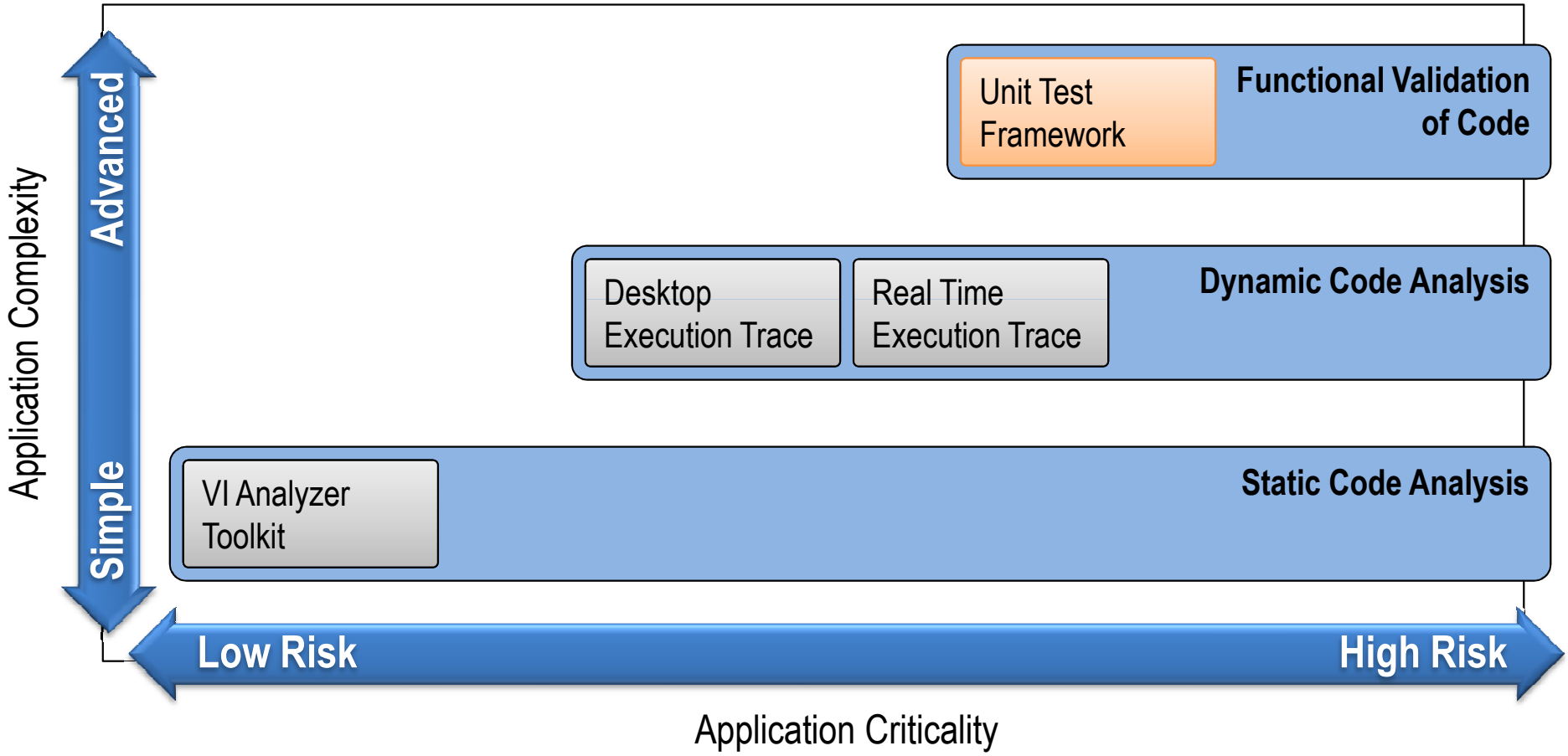
- Include block diagrams with an executable
- Remotely troubleshoot using LabVIEW debug tools



NATIONAL INSTRUMENTS

Tools for Debugging and Testing

Debugging & Testing

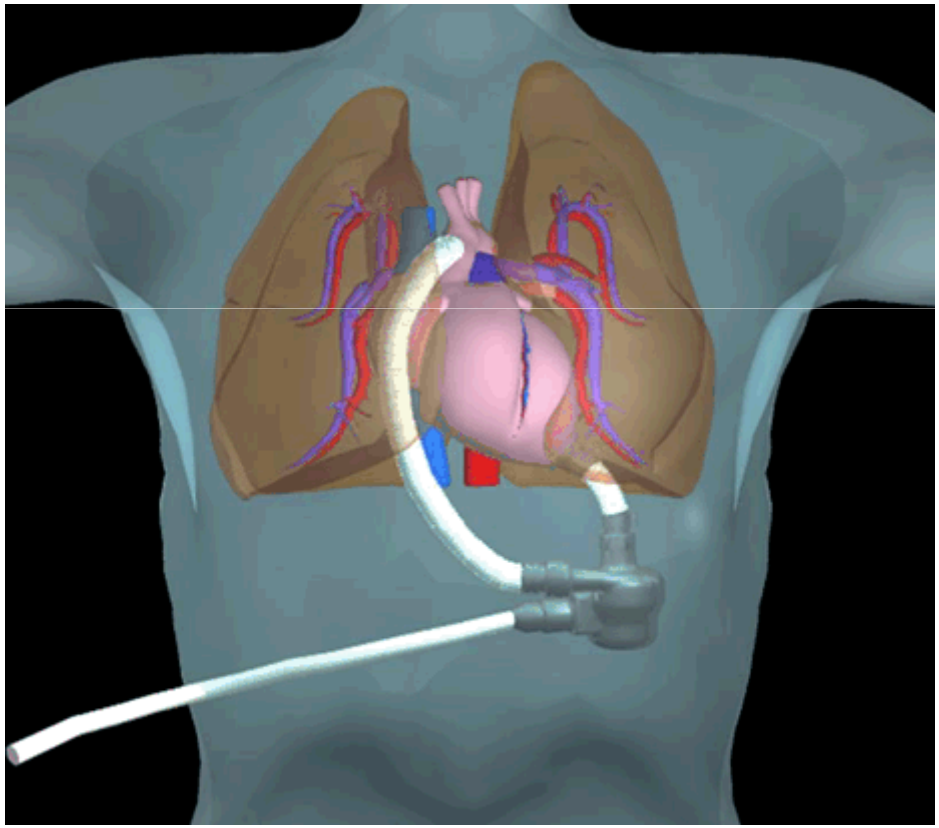


NATIONAL INSTRUMENTS

Data Science Automation

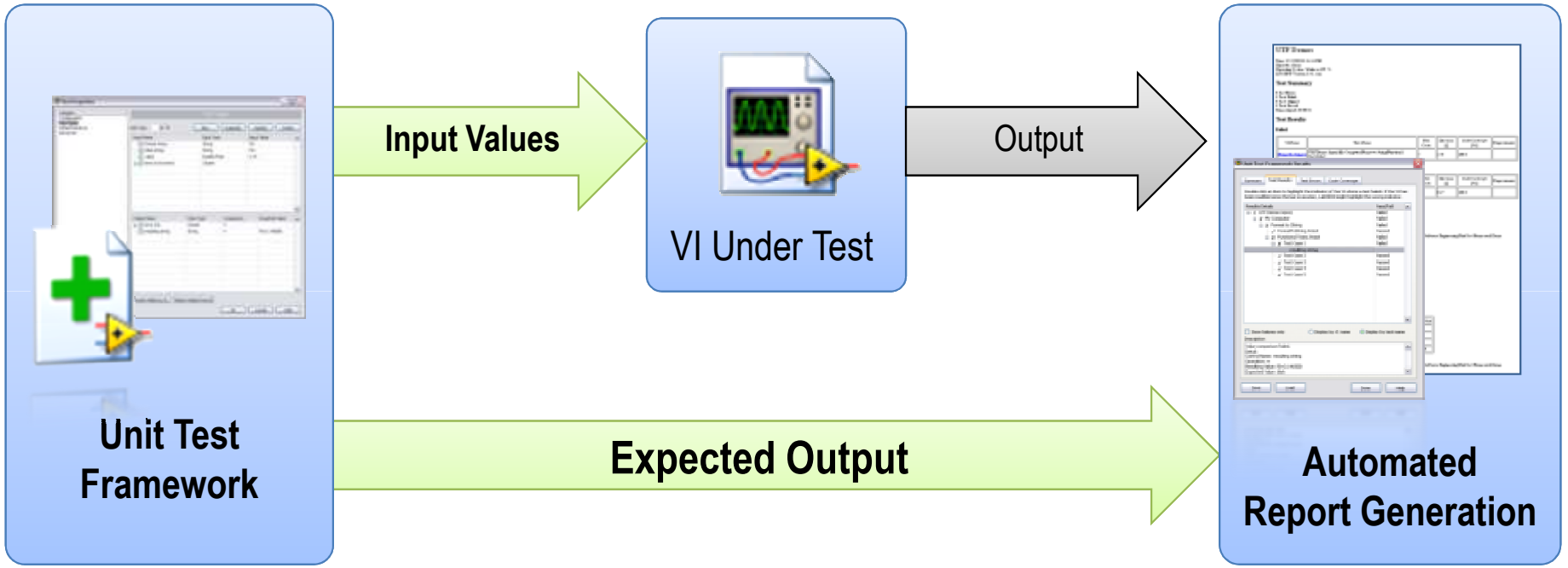
Developed Long-term testing solution for artificial hearts

"The Unit Test Framework's flexibility and ease of use certainly enabled much faster and reliable completion of this portion of the project potentially saving **hundreds of hours and thousands of dollars** over the life of the project."



NATIONAL INSTRUMENTS

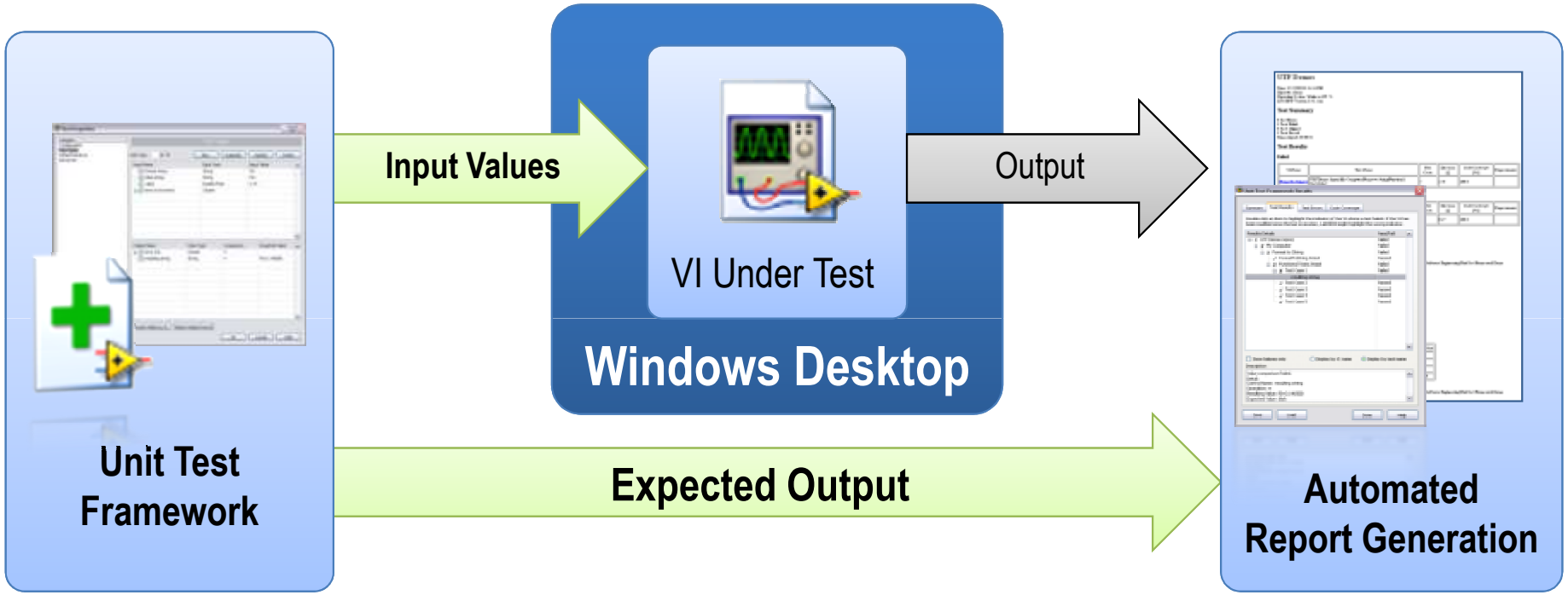
LabVIEW Unit Test Framework



Test vector = Input value(s) + Expected output(s)

NATIONAL INSTRUMENTS

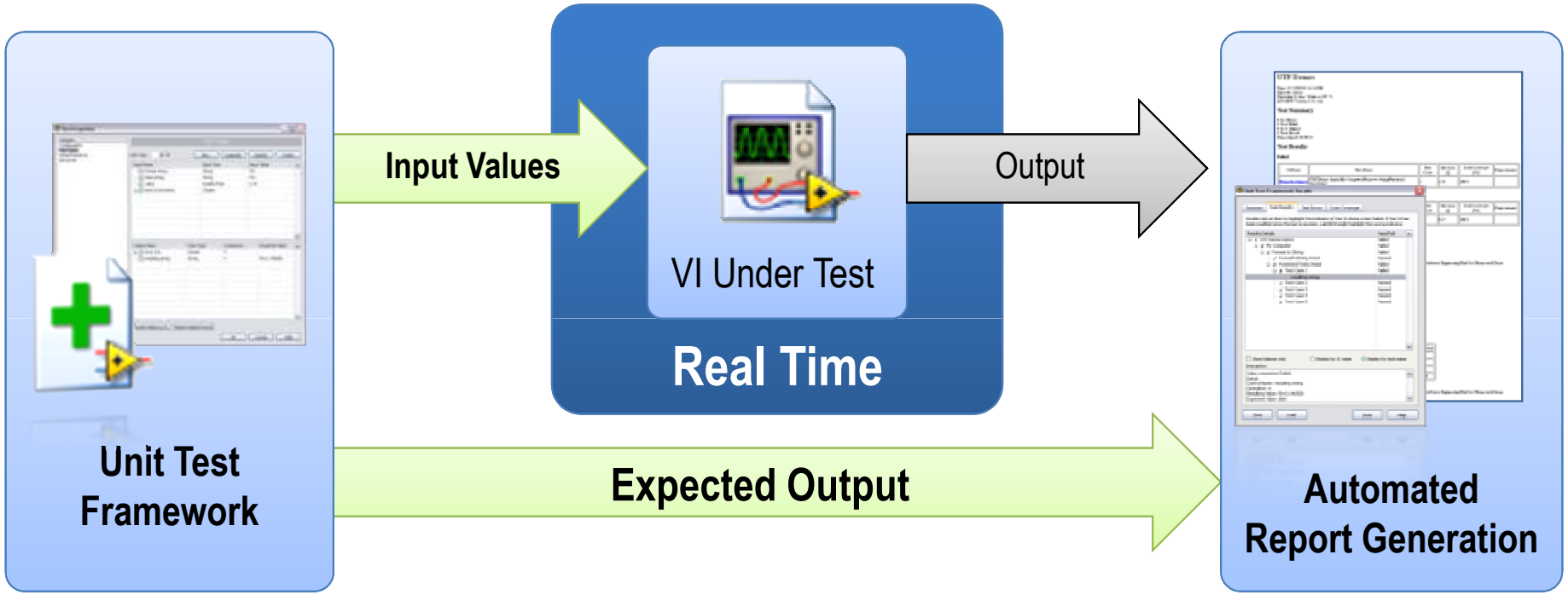
LabVIEW Unit Test Framework



Test vector = Input value(s) + Expected output(s)

NATIONAL INSTRUMENTS

LabVIEW Unit Test Framework



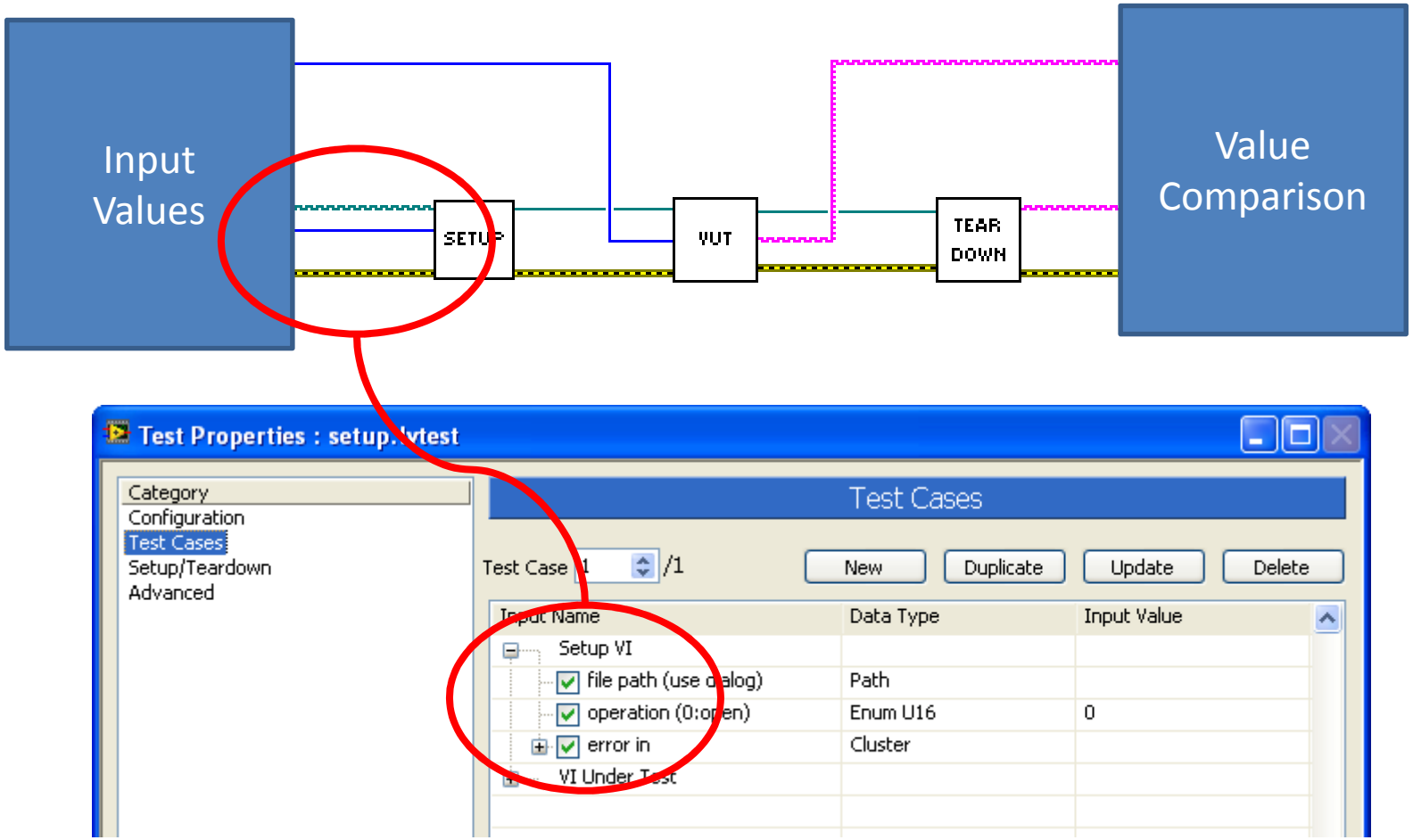
Test vector = Input value(s) + Expected output(s)

NATIONAL INSTRUMENTS

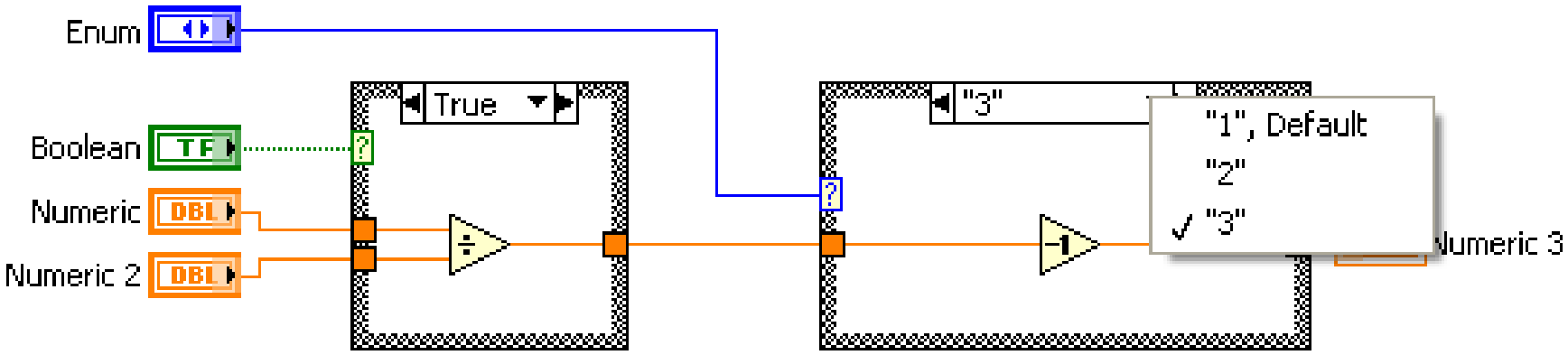
Unit Testing

DEMO

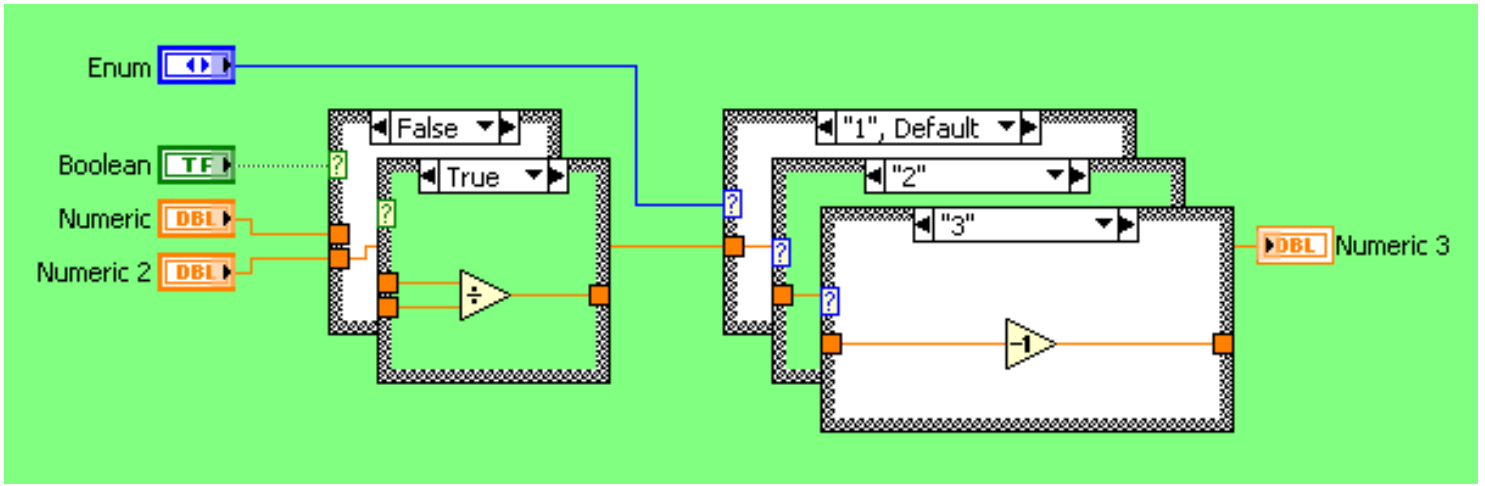
Setup / Teardown VIs



Code Coverage Example



6 Diagrams. 3 Diagrams Executed. 50% Code Coverage

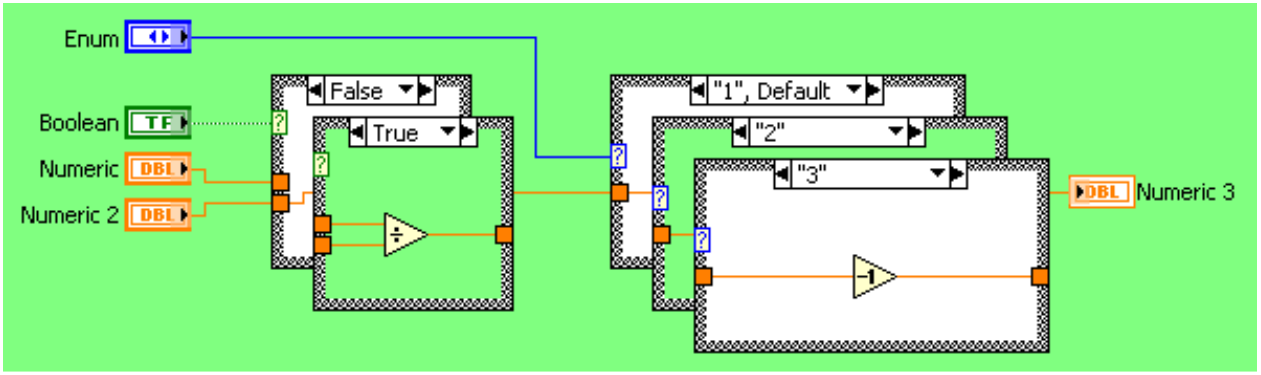


NATIONAL INSTRUMENTS

Code Coverage Example

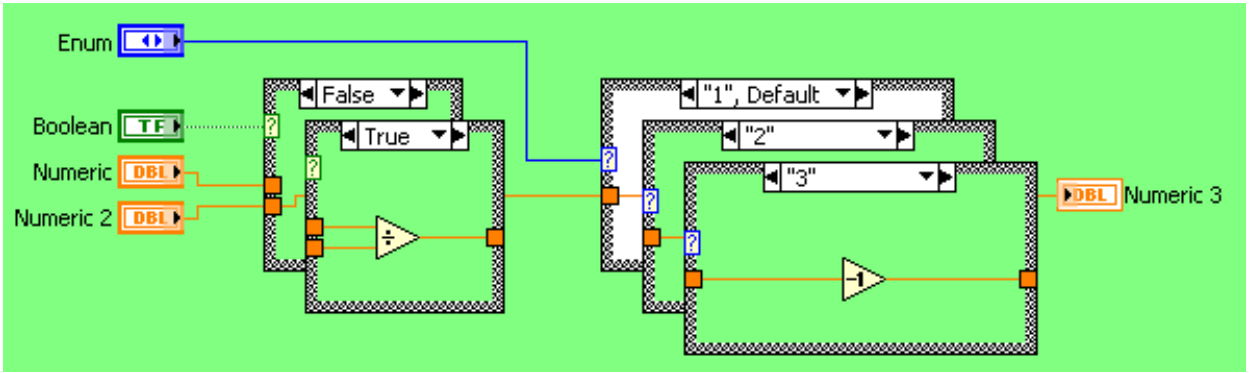
1ST Test Vector:

Block diagram, 2 Case diagrams executed. (2 + 1) / 6 = 50% Code Coverage



2nd Test Vector (aggregates covered code from 1st pass)

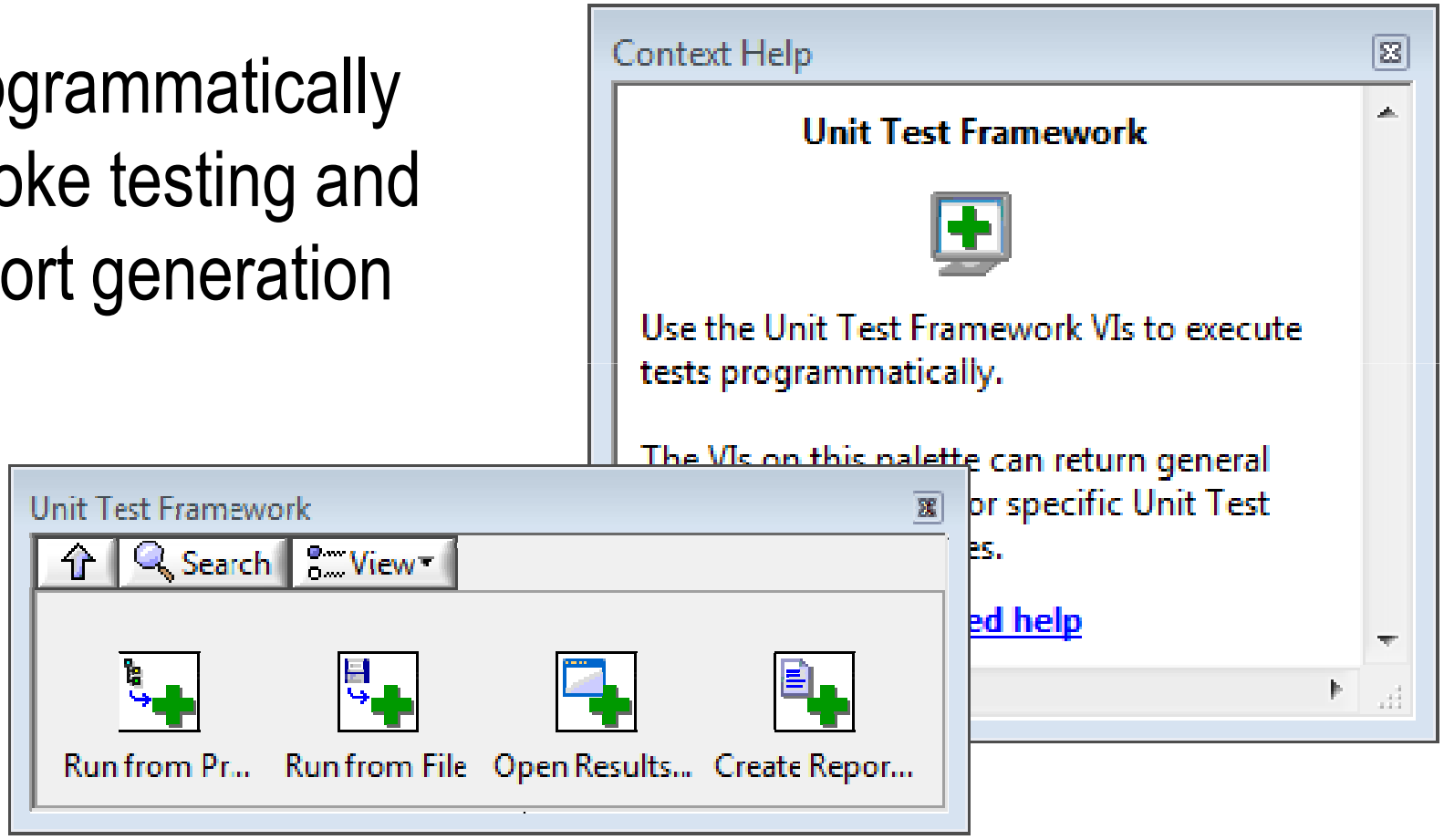
Block diagram, 5 Case diagrams executed. (4 + 1) / 6 = 83.33% Code Coverage



NATIONAL INSTRUMENTS

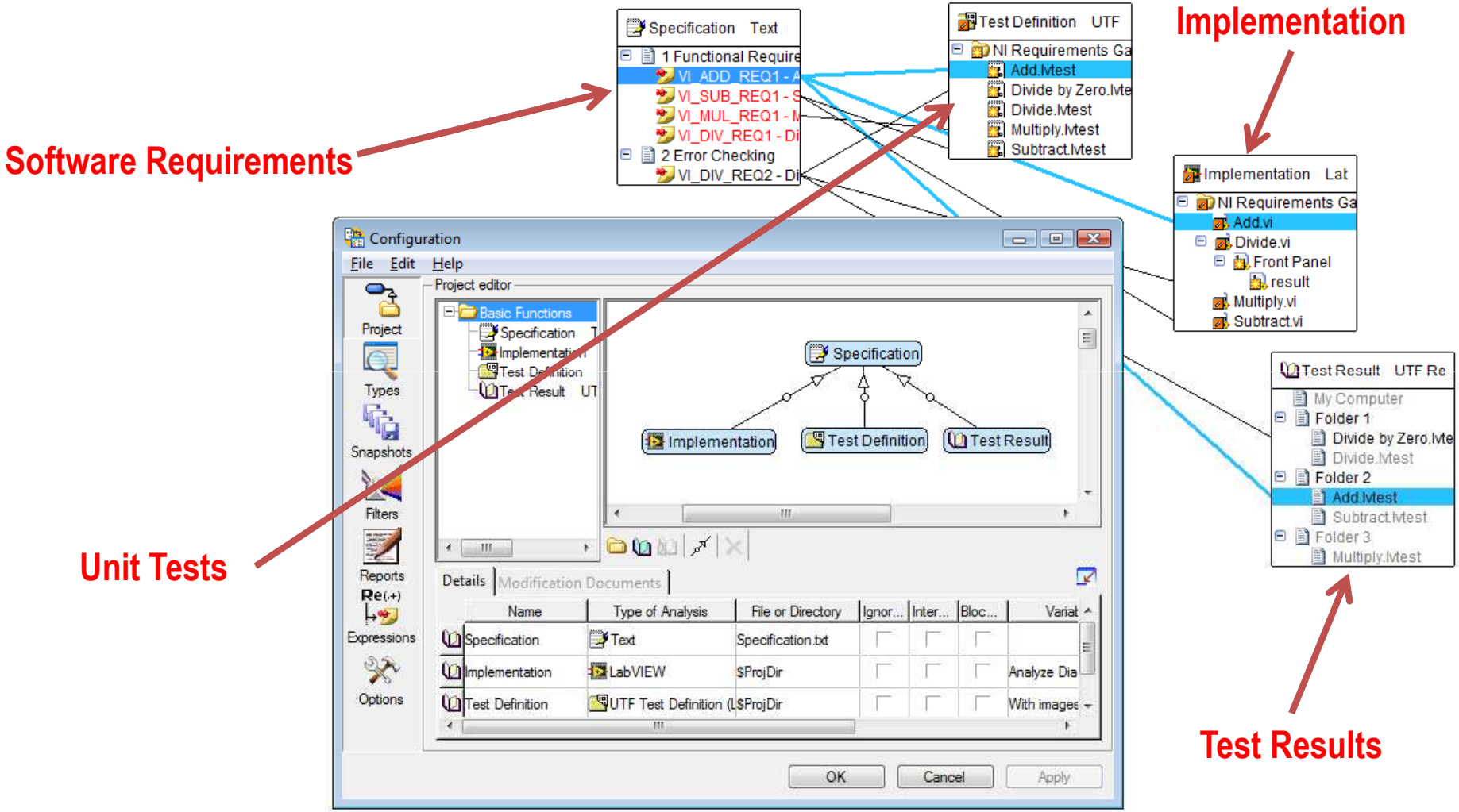
Programmatic Unit Testing

- Programmatically invoke testing and report generation



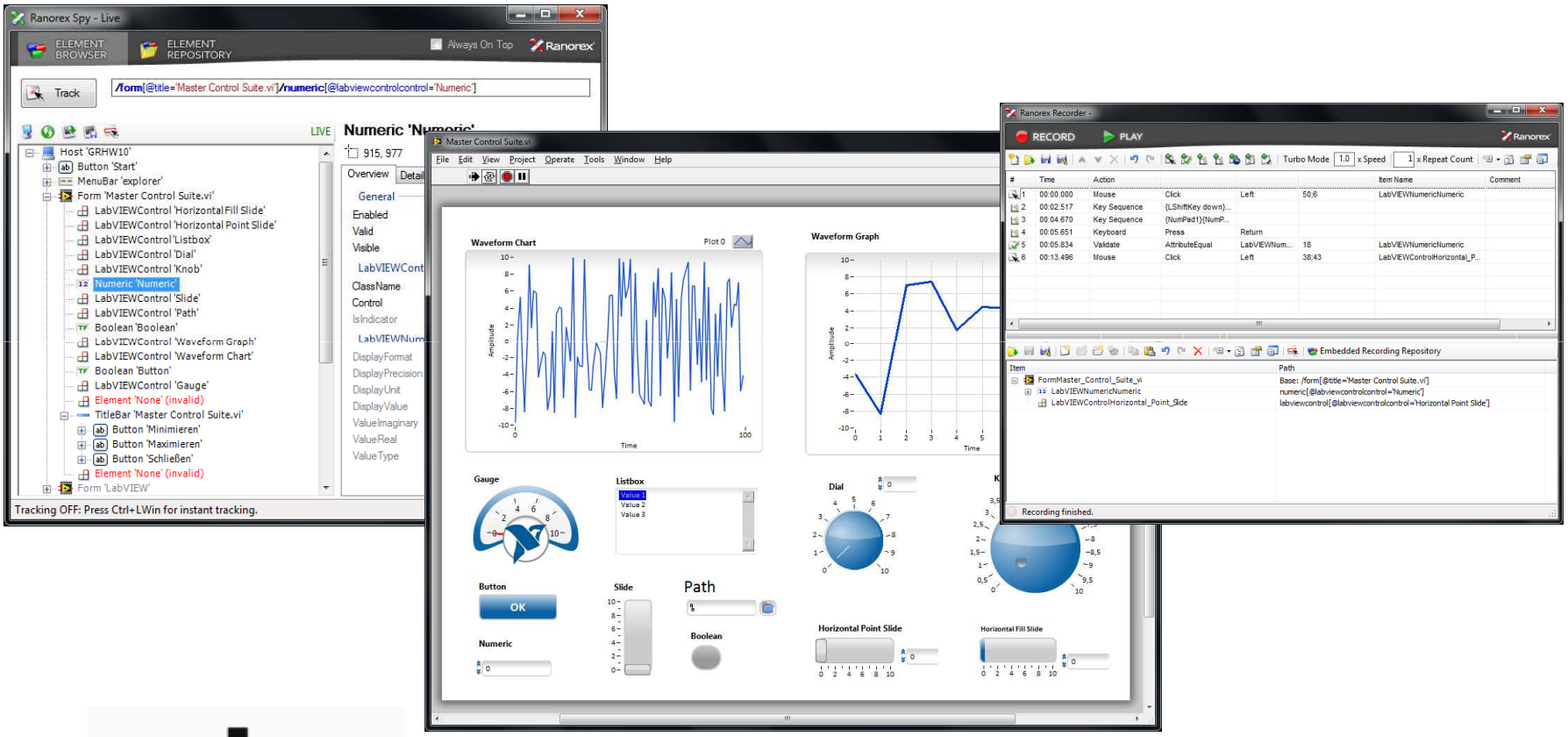
NATIONAL INSTRUMENTS

Integration with Requirements Gateway



NATIONAL INSTRUMENTS

LabVIEW User Interface Testing



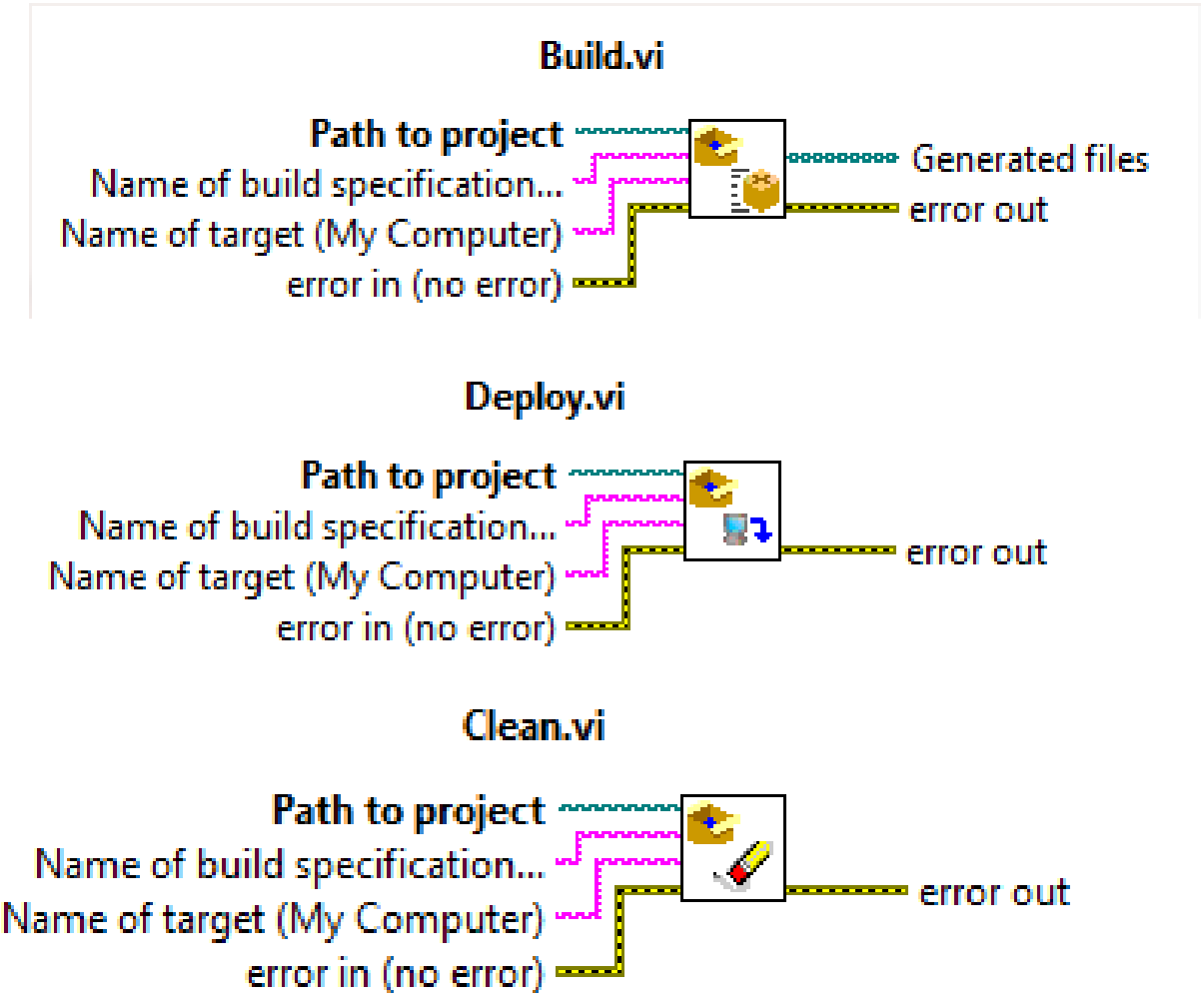
nte SYSTEMS

Ranorex

NATIONAL INSTRUMENTS LabVIEW™

Application Builder API *Coming in LabVIEW 2011*

Automate build and deployment processes

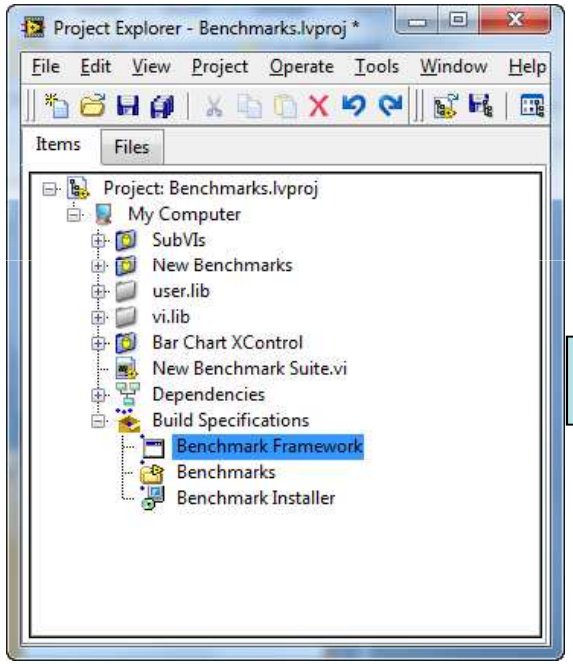


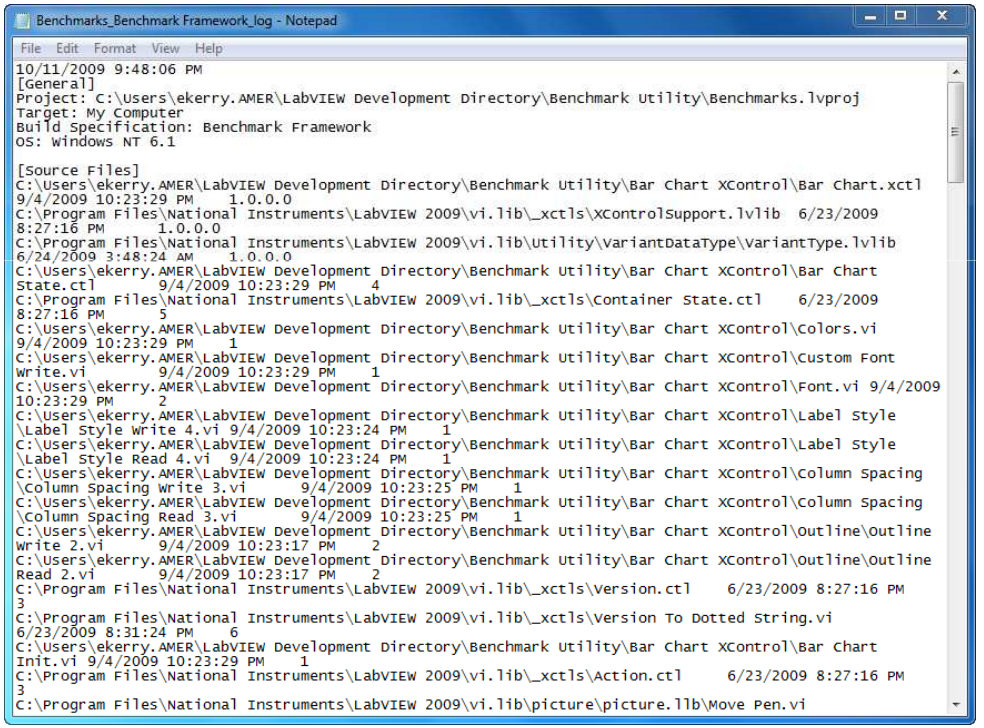
NATIONAL INSTRUMENTS

App Builder Log-File Generation

Document version, date and time of Vis included in build

Available in LabVIEW 2009





NATIONAL INSTRUMENTS

Section Three

Advanced Software Architectures and Data Communication

NATIONAL INSTRUMENTS

What is a Design Pattern?

Definition: A well-established solution to a common problem.

Why Should I Use One?

Save time and improve the longevity and readability of your code.

… or else…

Designing for SMoRES

Criteria for a well designed software application:

Scalable: how simple is N + 1?

Modular: is the application broken up into well-defined components that stand on their own?

Reusable: is the code de-coupled from the current application well-enough such that it could be reused in a future project?

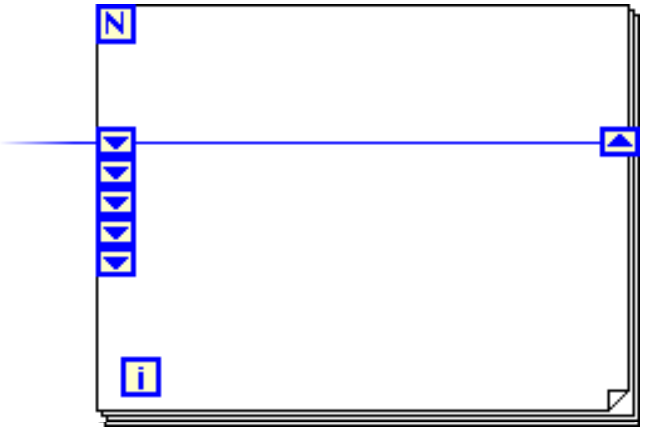
Extensible: how painful is it to add new functionality?

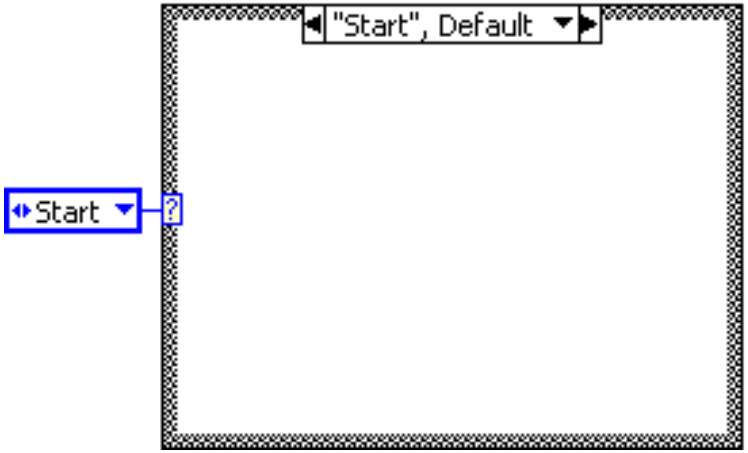
Simple: what is the simplest solution that satisfies all of the listed criteria and the requirements of the application?

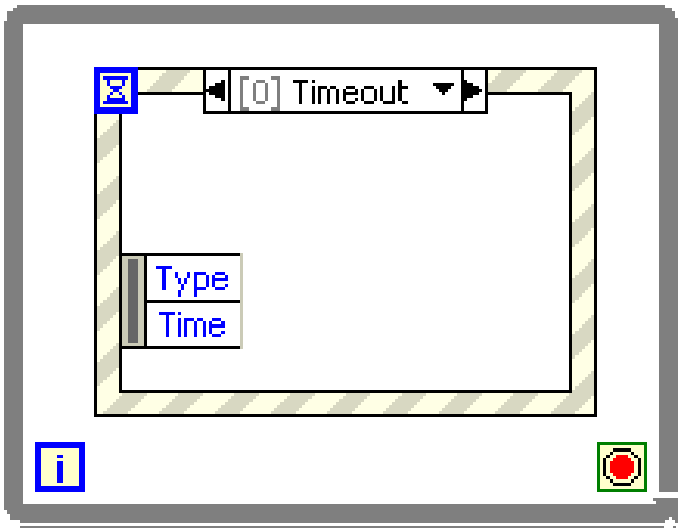
NATIONAL INSTRUMENTS

You Should Already Be Familiar With..

- Loops
- Shift Registers
- Case Structures
- Enumerated Constants
- Event Structures
- LabVIEW Classes





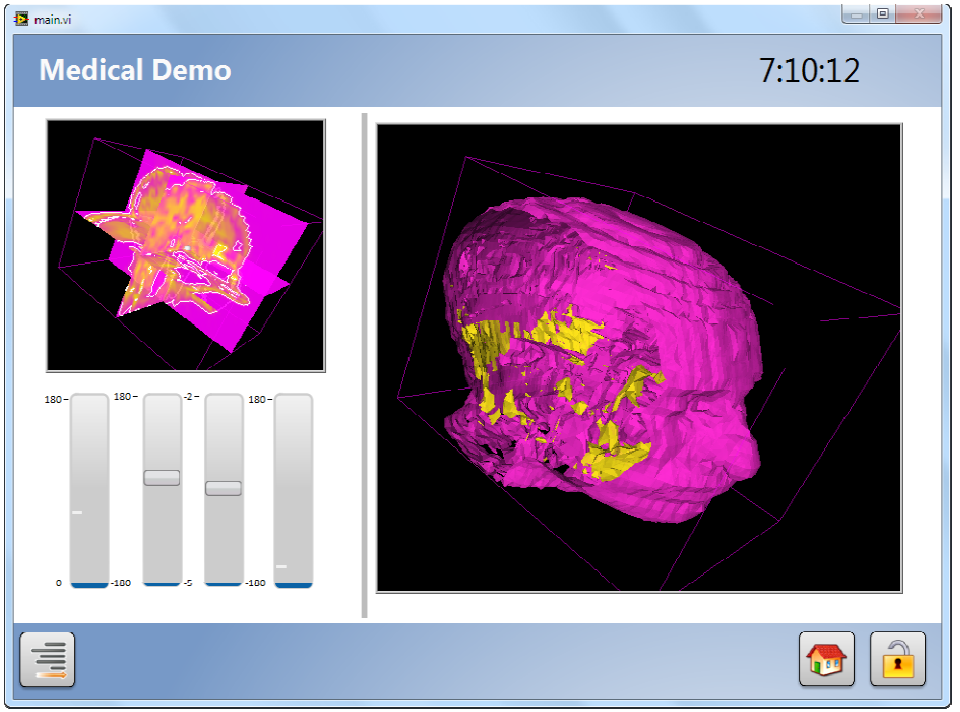


NATIONAL INSTRUMENTS

User Interface Considerations

Common Requirements

1. Highly-Responsive
2. Data appears to be streaming
3. Multiple interfaces



Considerations

1. Polling is inefficient
2. The screen has finite pixels

NATIONAL INSTRUMENTS

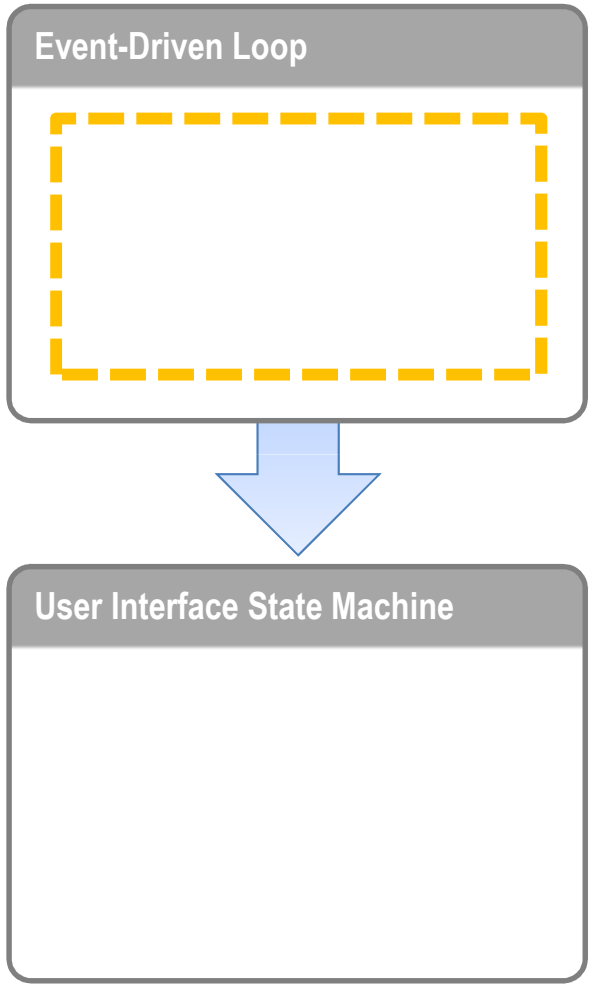
User Interfaces Should Be Responsive

Best Practices

1. Use an event structure
2. Do as little as you can in the event cases
3. Delegate actions to consumers
4. Defer panels updates for smooth updates

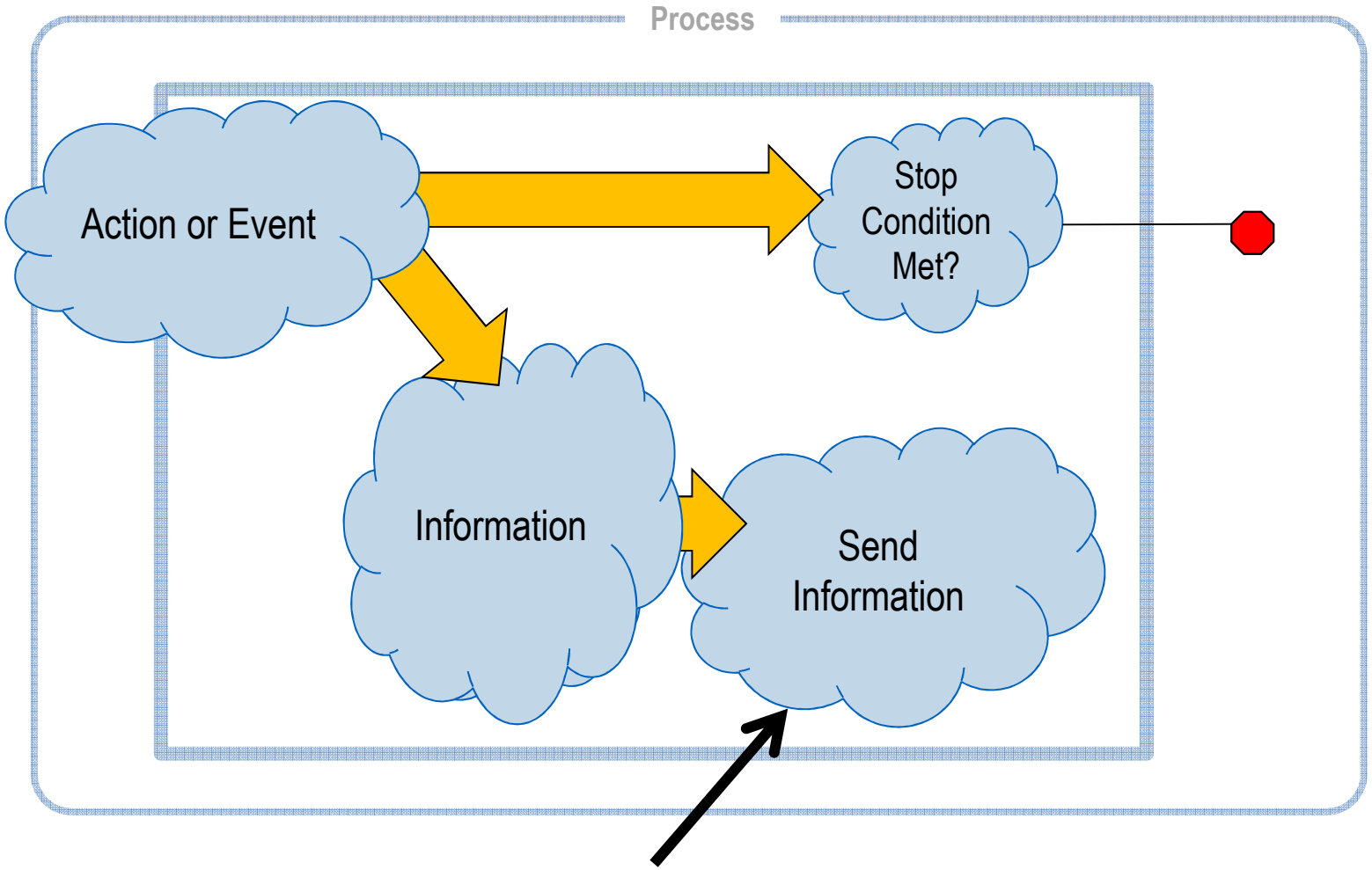
Considerations

1. How do you send commands?
2. How do you send data?
3. How fast do you *really* have to update the front panel?



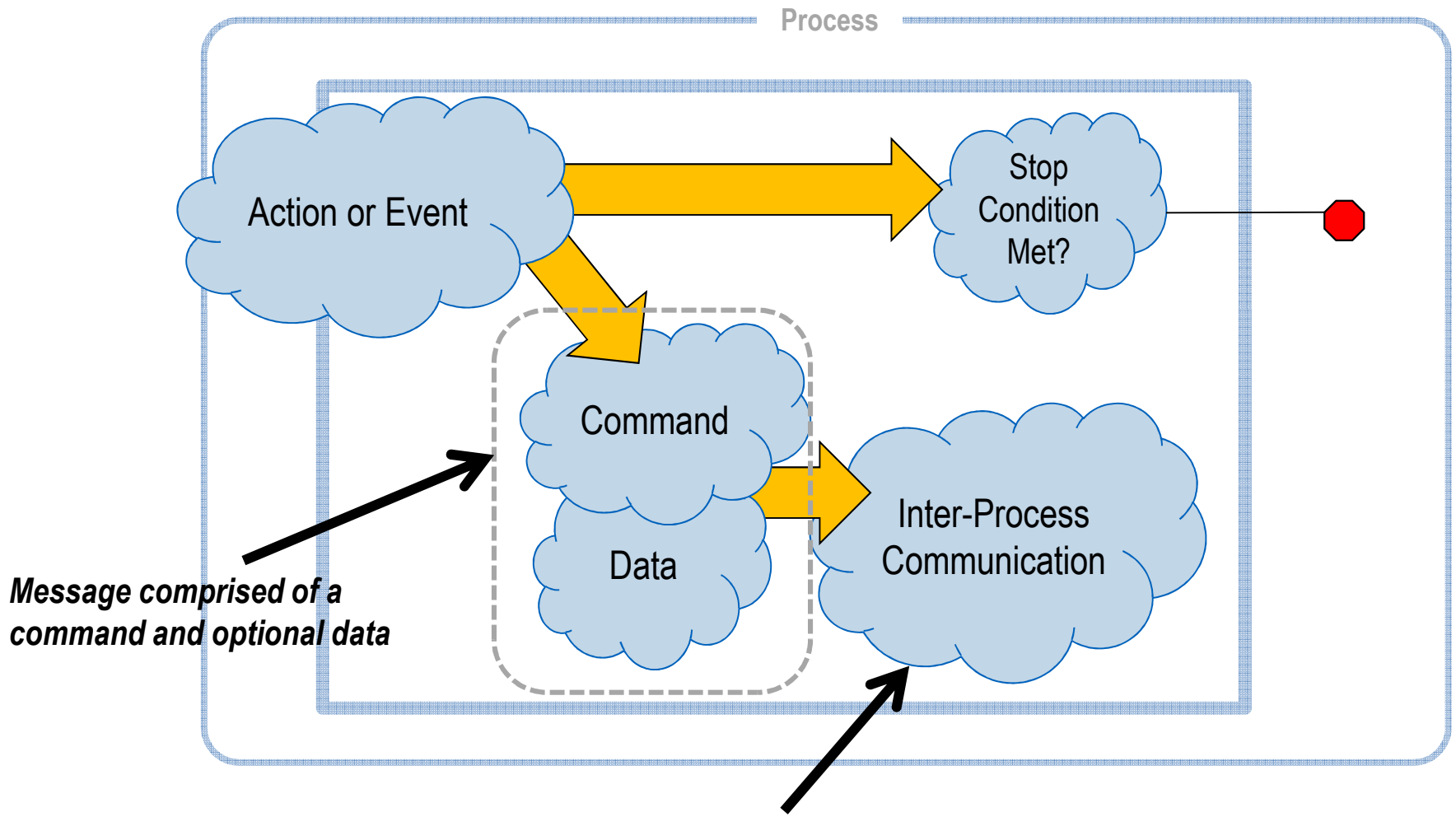
NATIONAL INSTRUMENTS

Anatomy of a Producer Process



NATIONAL INSTRUMENTS

Anatomy of a Message Producer Process

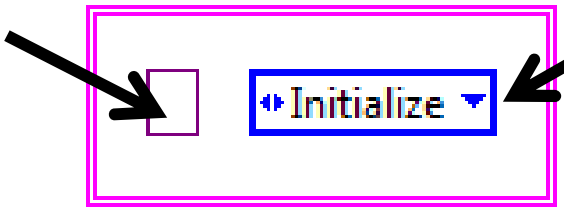


NATIONAL INSTRUMENTS

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

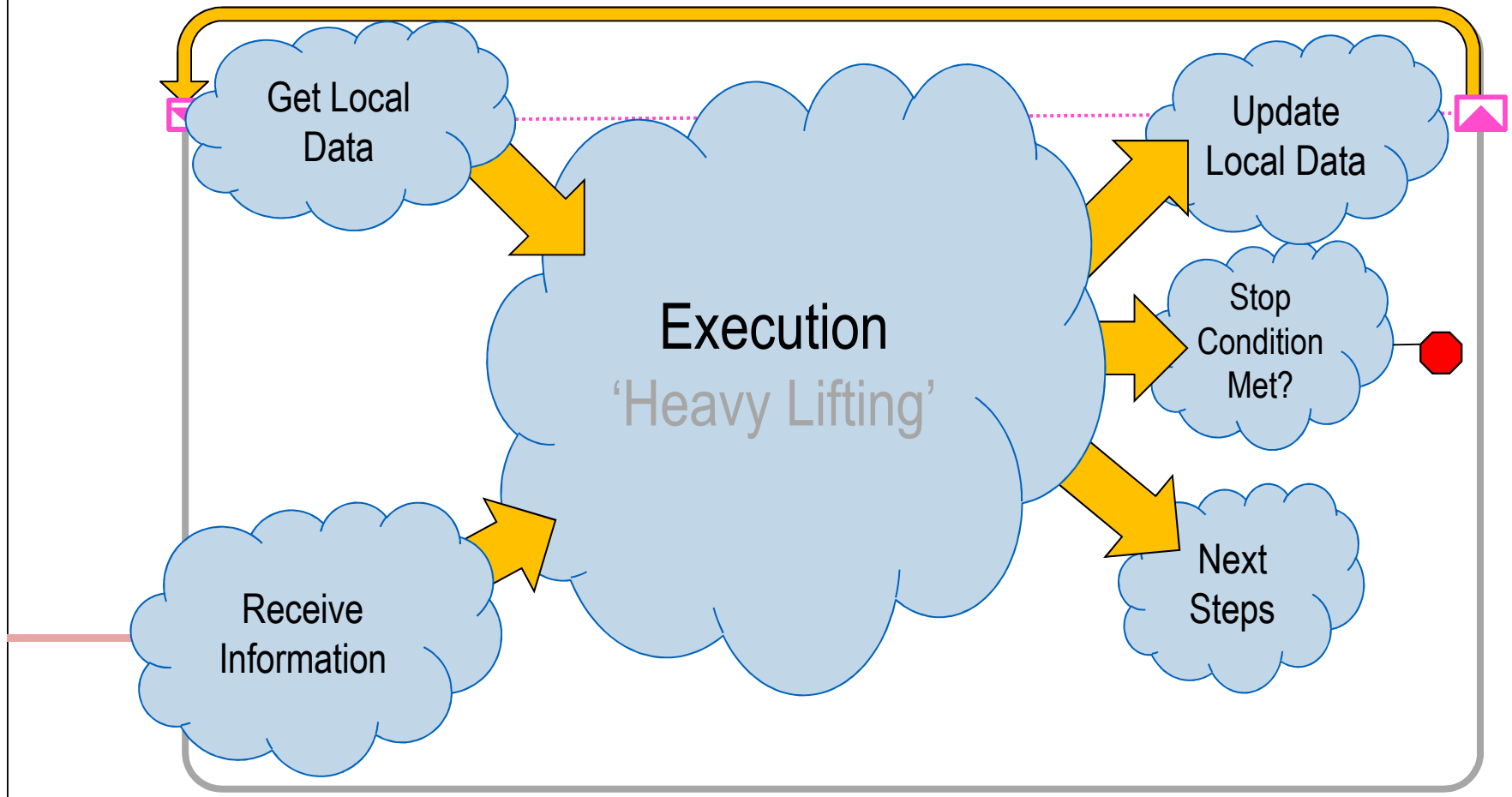
Command	Data
Initialize UI	Cluster containing configuration data
Populate Menu	Array of strings to display in menu
Resize Display	Array of integers [Width, Height]
Load Subpanel	Reference of VI to Load
Insert Header	String
Stop	-

NATIONAL INSTRUMENTS

Basic State Machine

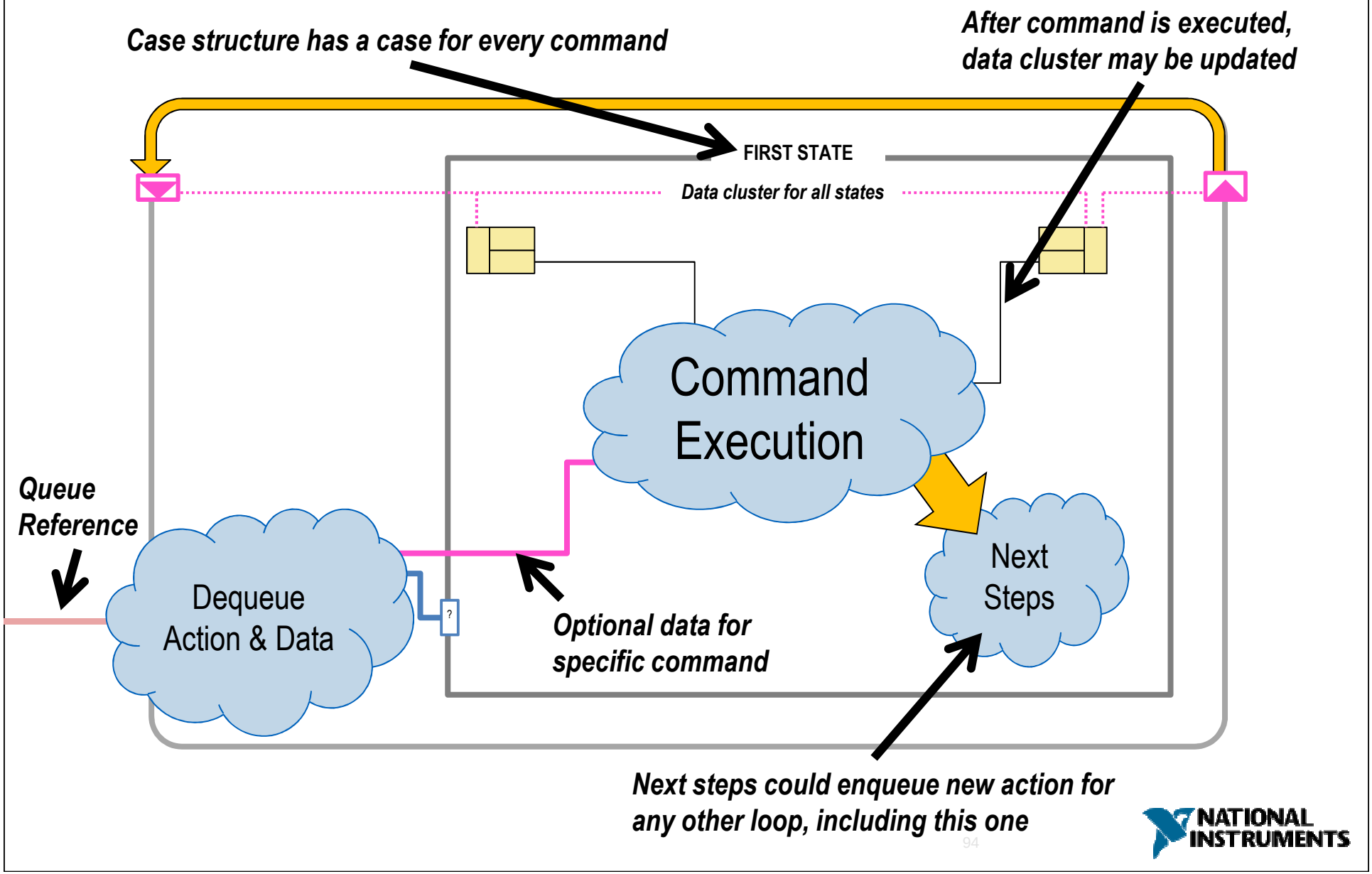
DEMO

Anatomy of a Consumer Process



NATIONAL INSTRUMENTS

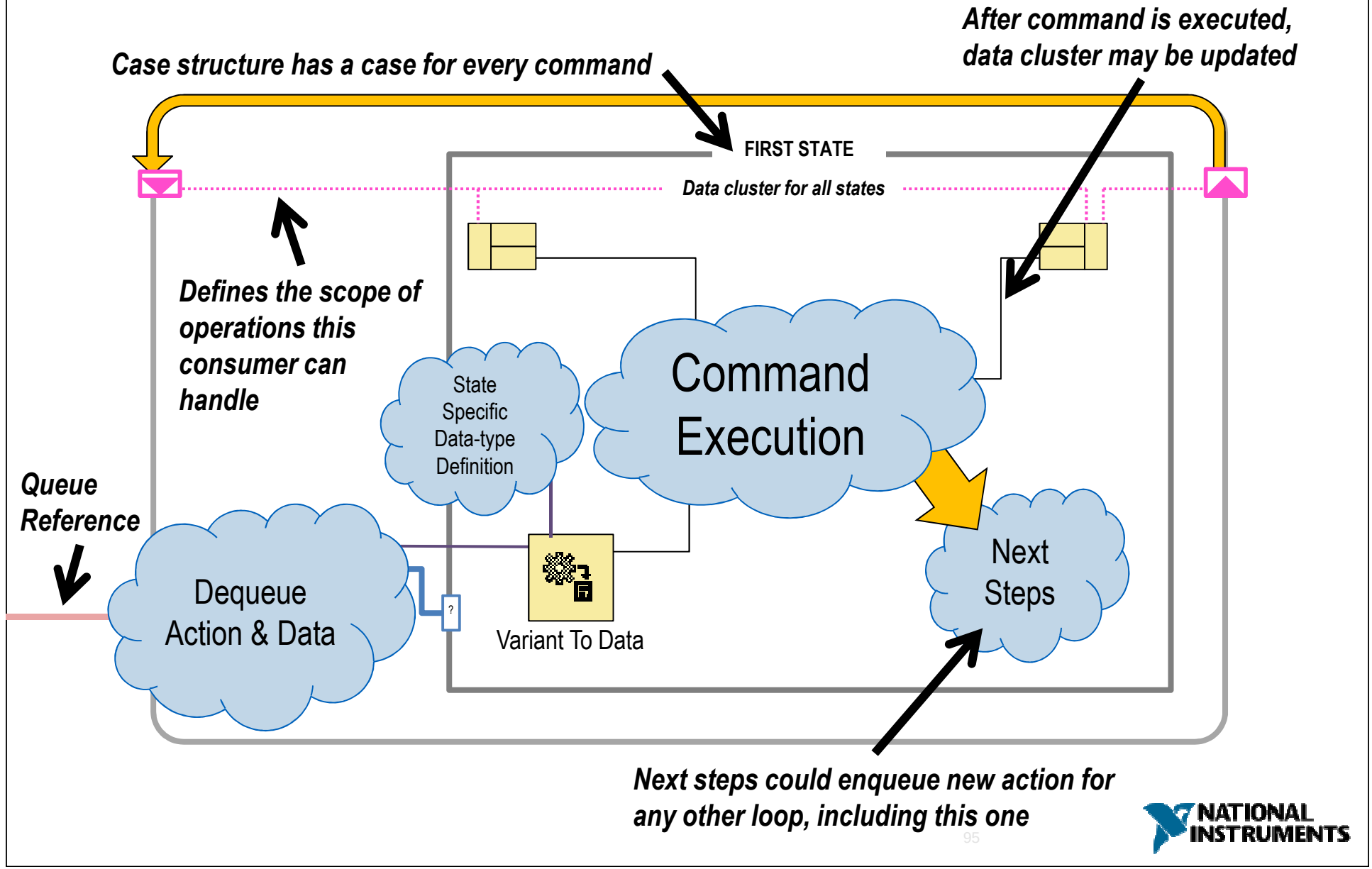
Anatomy of a Message Consumer Process



NATIONAL INSTRUMENTS

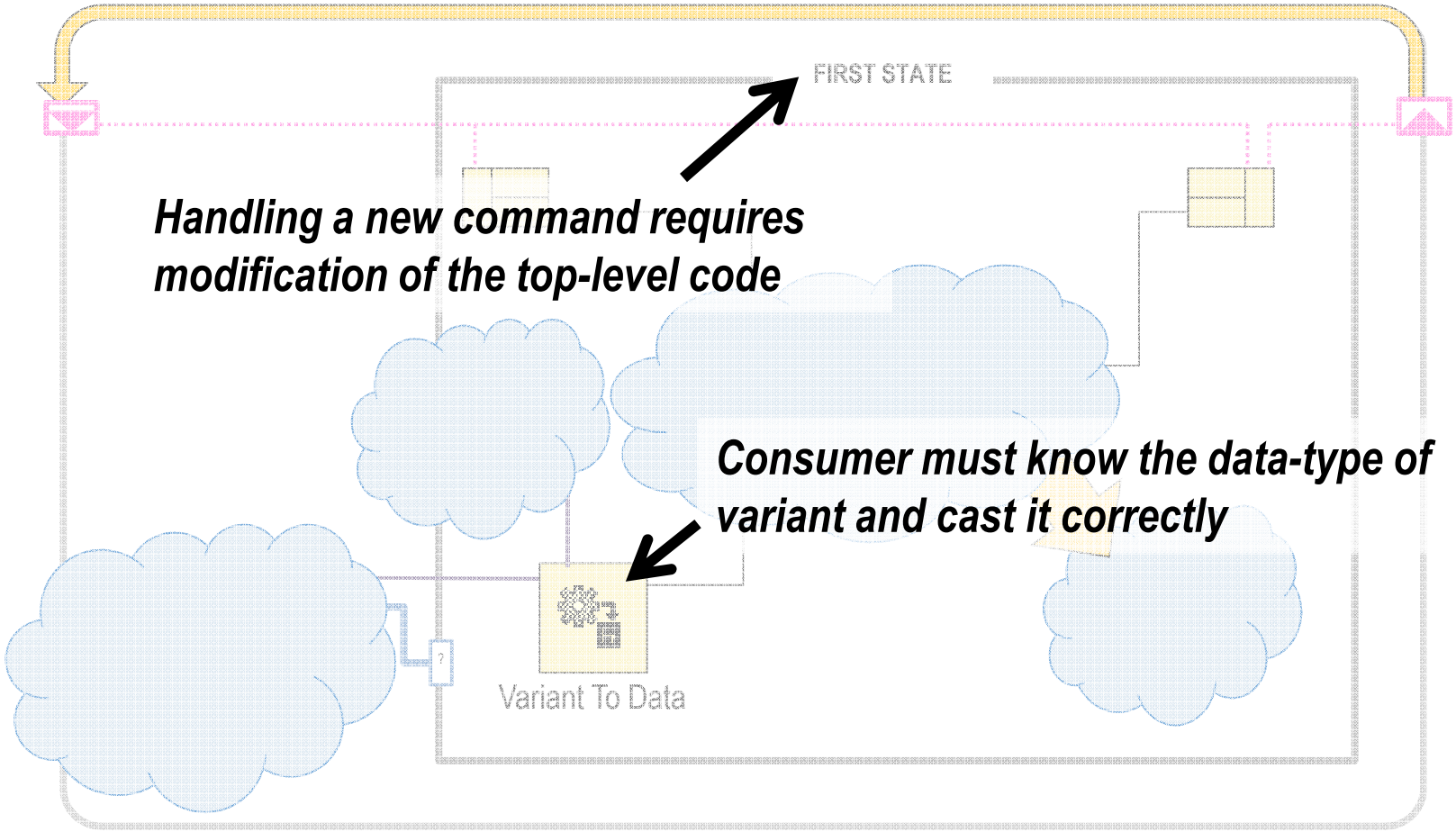
Anatomy of a Message Consumer Process

Also known as 'Command Pattern'



NATIONAL INSTRUMENTS

Message Consumer Considerations



NATIONAL INSTRUMENTS

Why Use Object-Orientation in LabVIEW?

- Encapsulate and protect data
- Catch errors at compile time instead of run-time
- Easily extend functionality
- Eliminate large data clusters
- Implement established design patterns
- Map real-world problems to software

Understanding Object-Oriented Programming

Class: A collection of data and the methods that interact with that data.

Object: A specific instance of a class

Examples of classes

Class:	Car	Person
Data:	Make Model Year Mileage	First Name Last Name Date of Birth Gender
Methods:	Check Brakes Rotate Tires Change Oil	Get Full Name Get Age

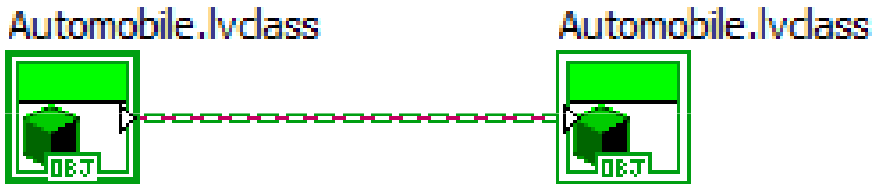
Examples of objects

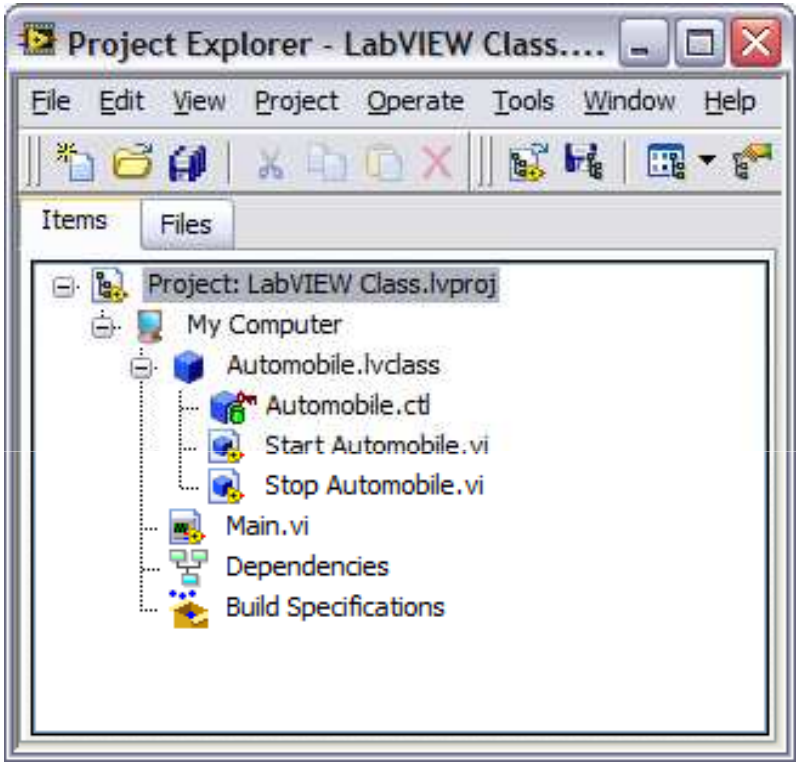
Class:	Car	Person
Object:	•1964 Ford Mustang •2004 Honda Accord	•Adam Kemp •Steven Harrison

What Is a LabVIEW Class?

Class: A collection of data and the methods that interact with that data.

Object: A specific instance of a class



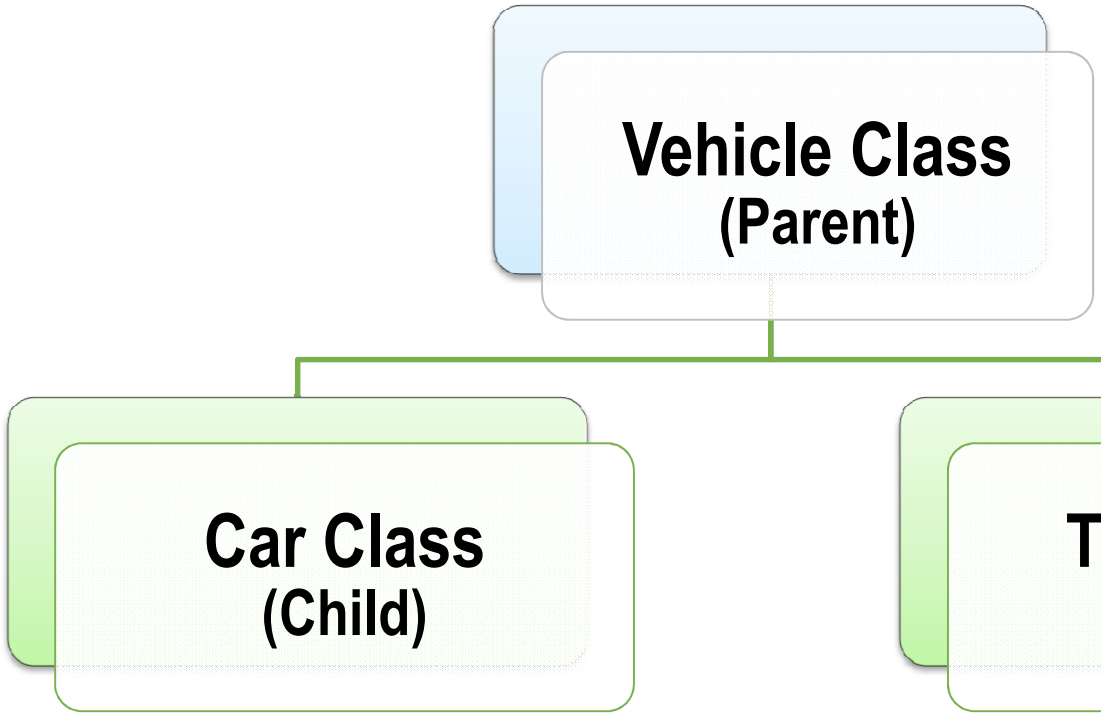


A LabVIEW Class is…

- A glorified cluster
- A user-defined data type
- A type of Project Library

NATIONAL INSTRUMENTS

What Is Inheritance?



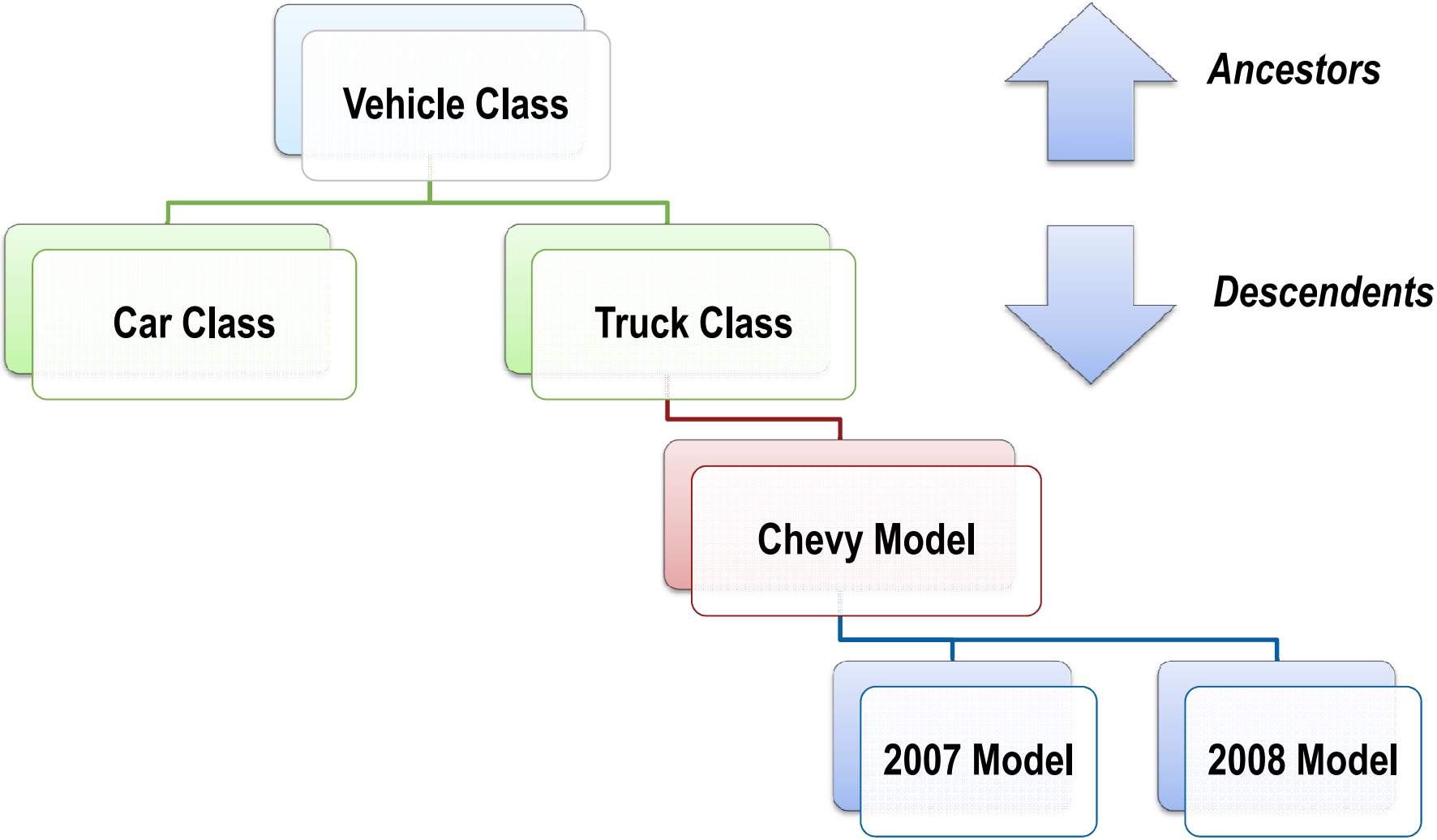
Example methods:

- Initialize
- Get Cargo Capacity

A car is a type of vehicle.

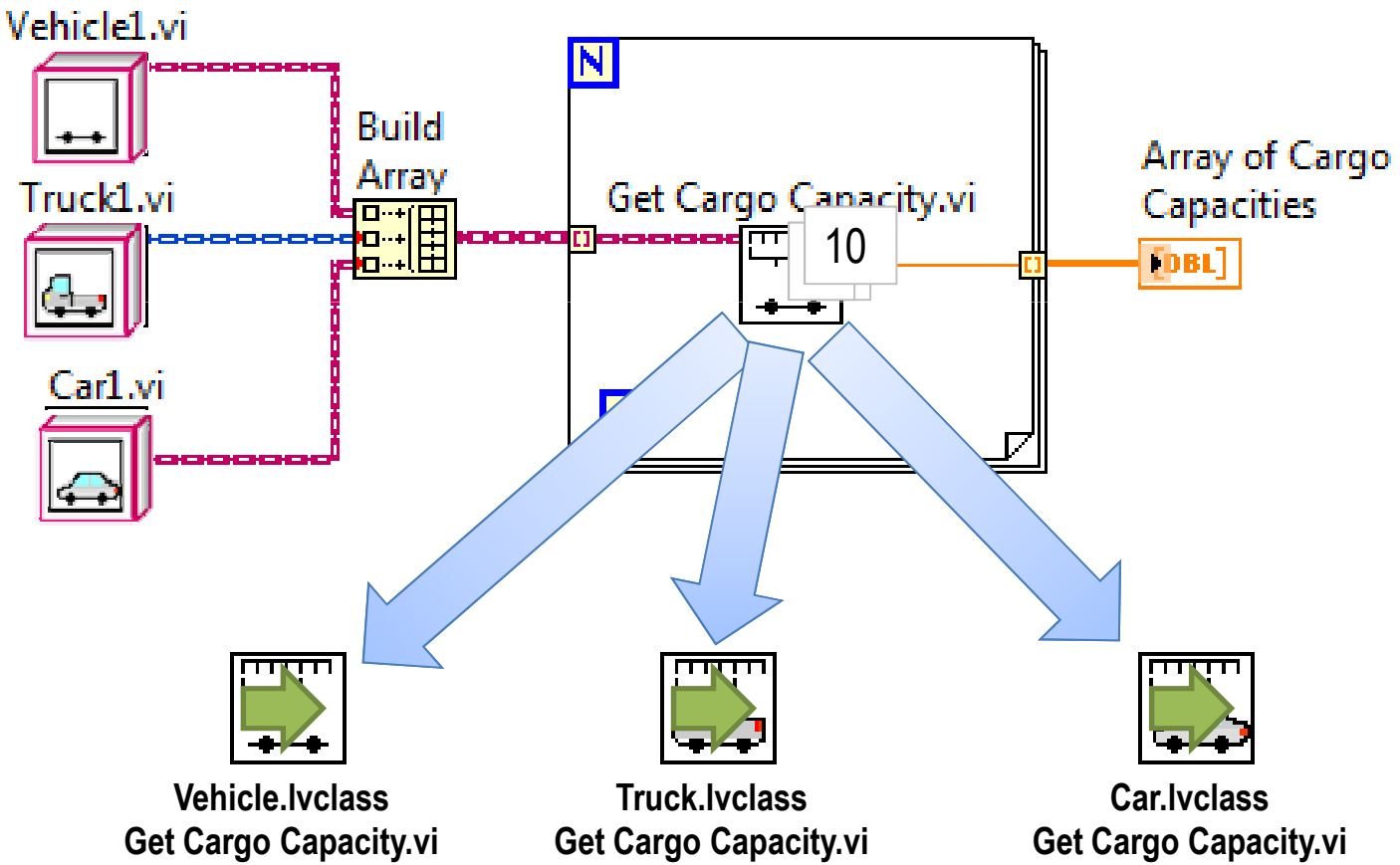
A truck is a type of vehicle.

Inheritance Example



NATIONAL INSTRUMENTS

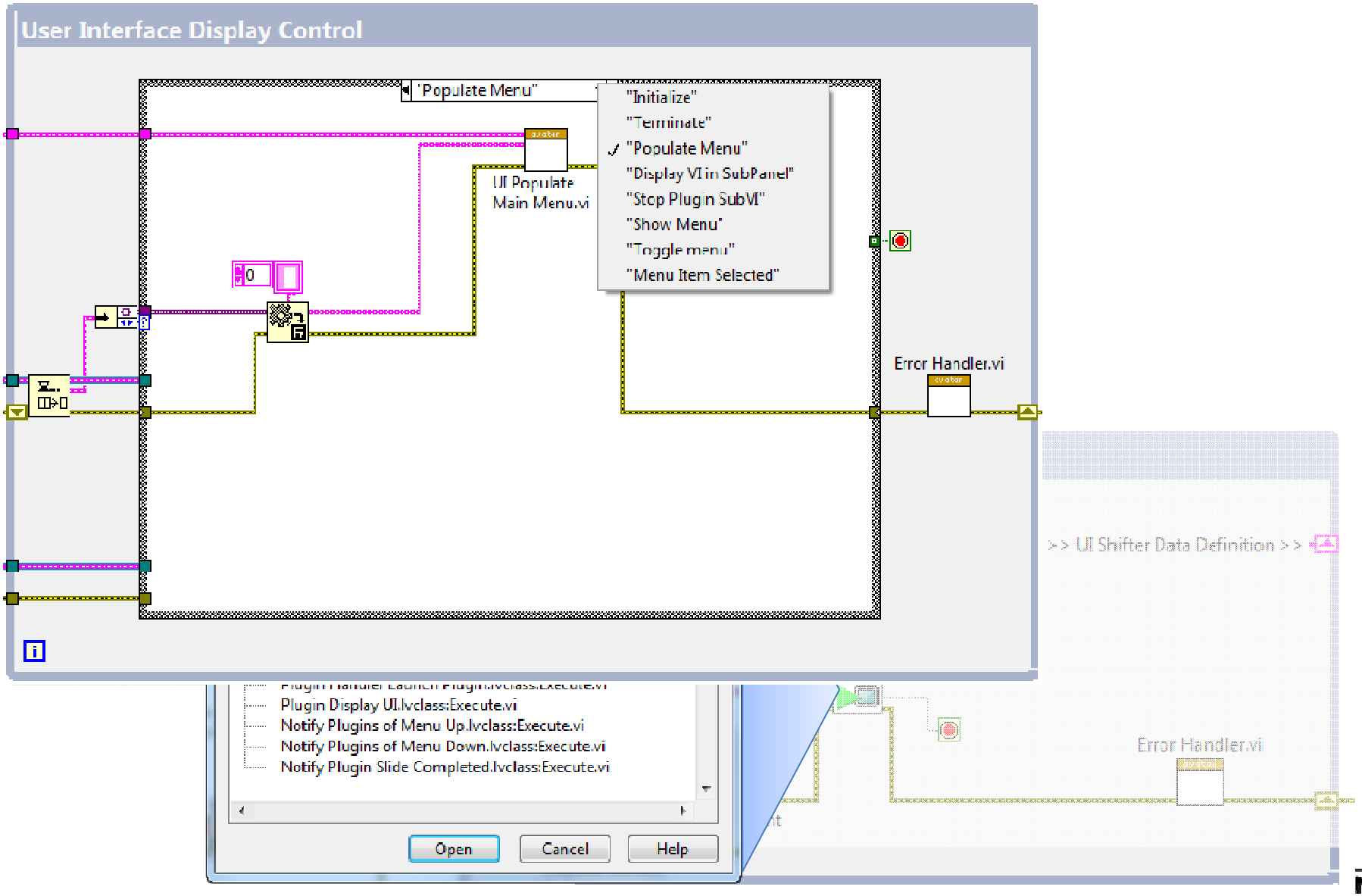
Understanding Dynamic Dispatch



NATIONAL INSTRUMENTS

State Machine vs. Command Pattern

These diagrams represent functionally equivalent code

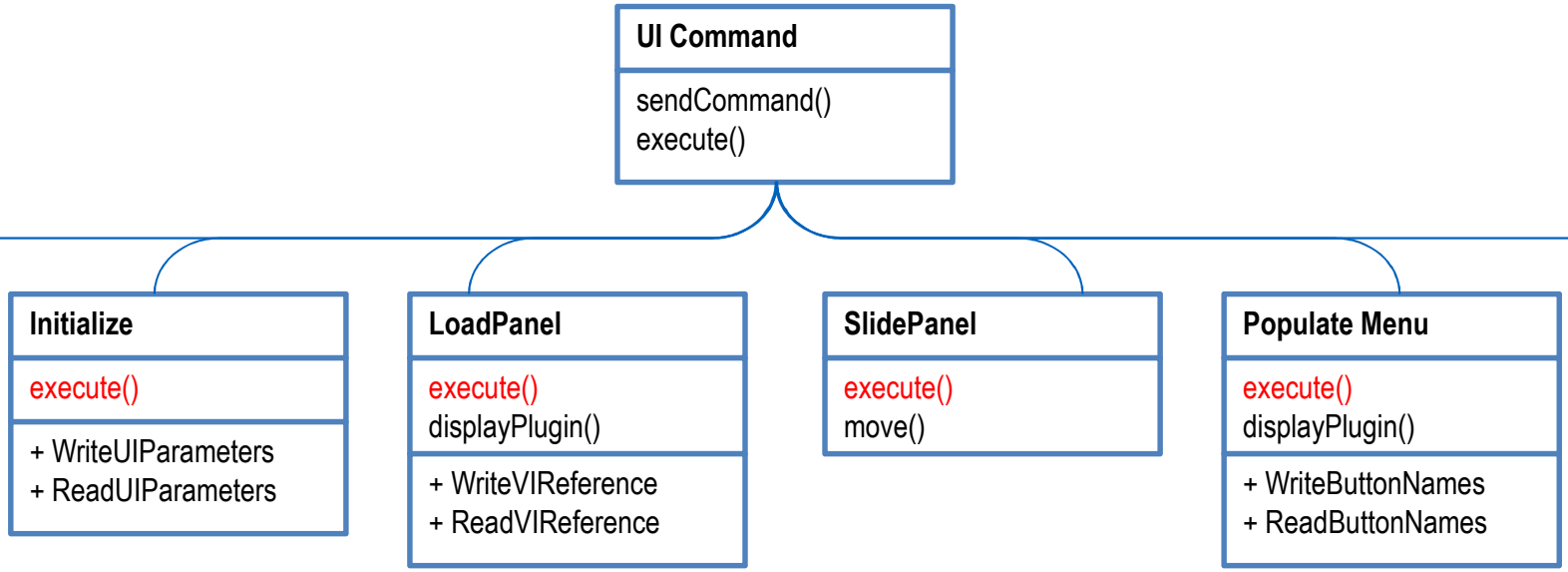


Traditional PC-QSM Approach

DEMO

OO-Based State Machine Design

Also known as 'Command Pattern' or 'Chain of Command Pattern'

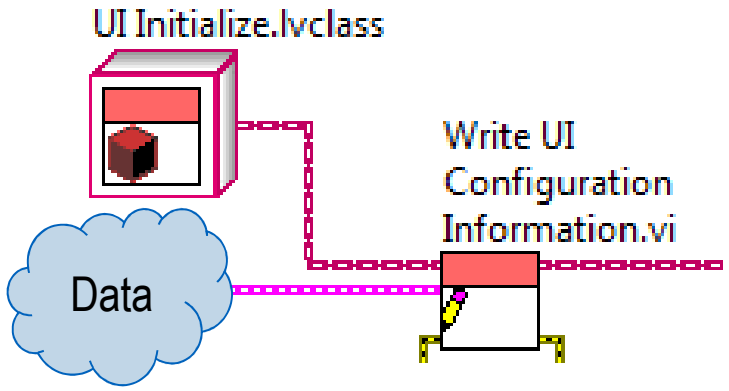


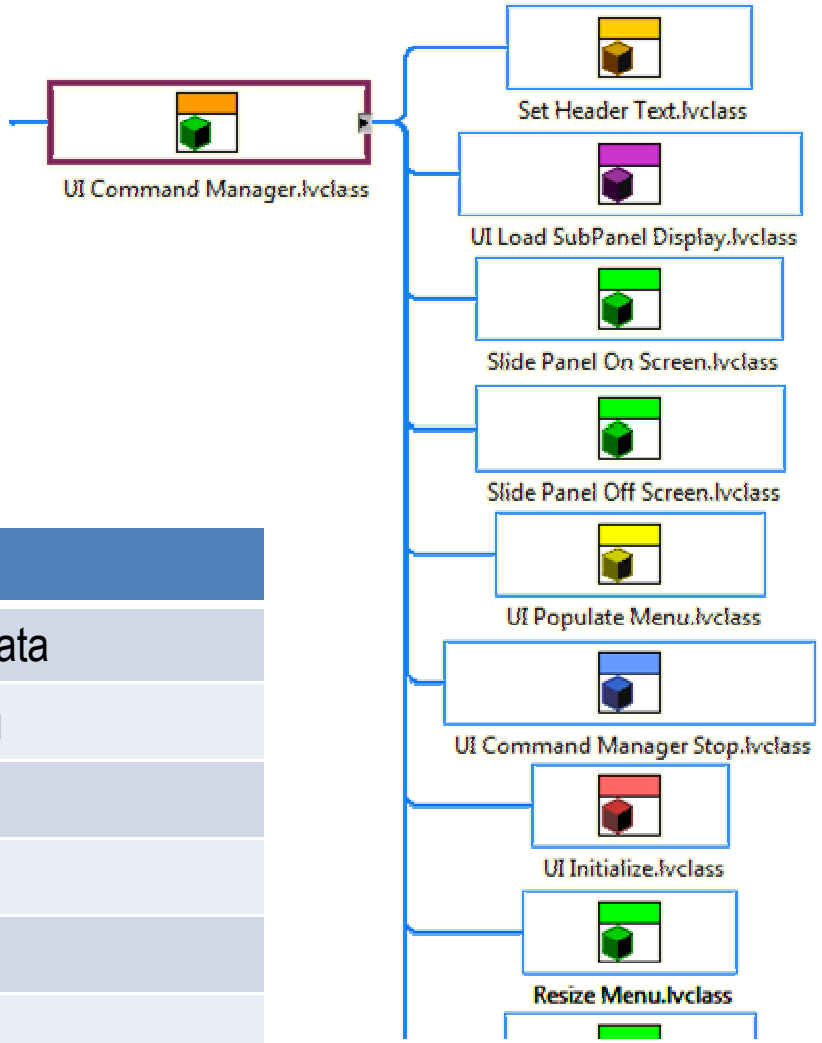
Uses **Dynamic Dispatching** to determine (at run-time) which version of the execute method gets run

NATIONAL INSTRUMENTS

Command Class Hierarchy

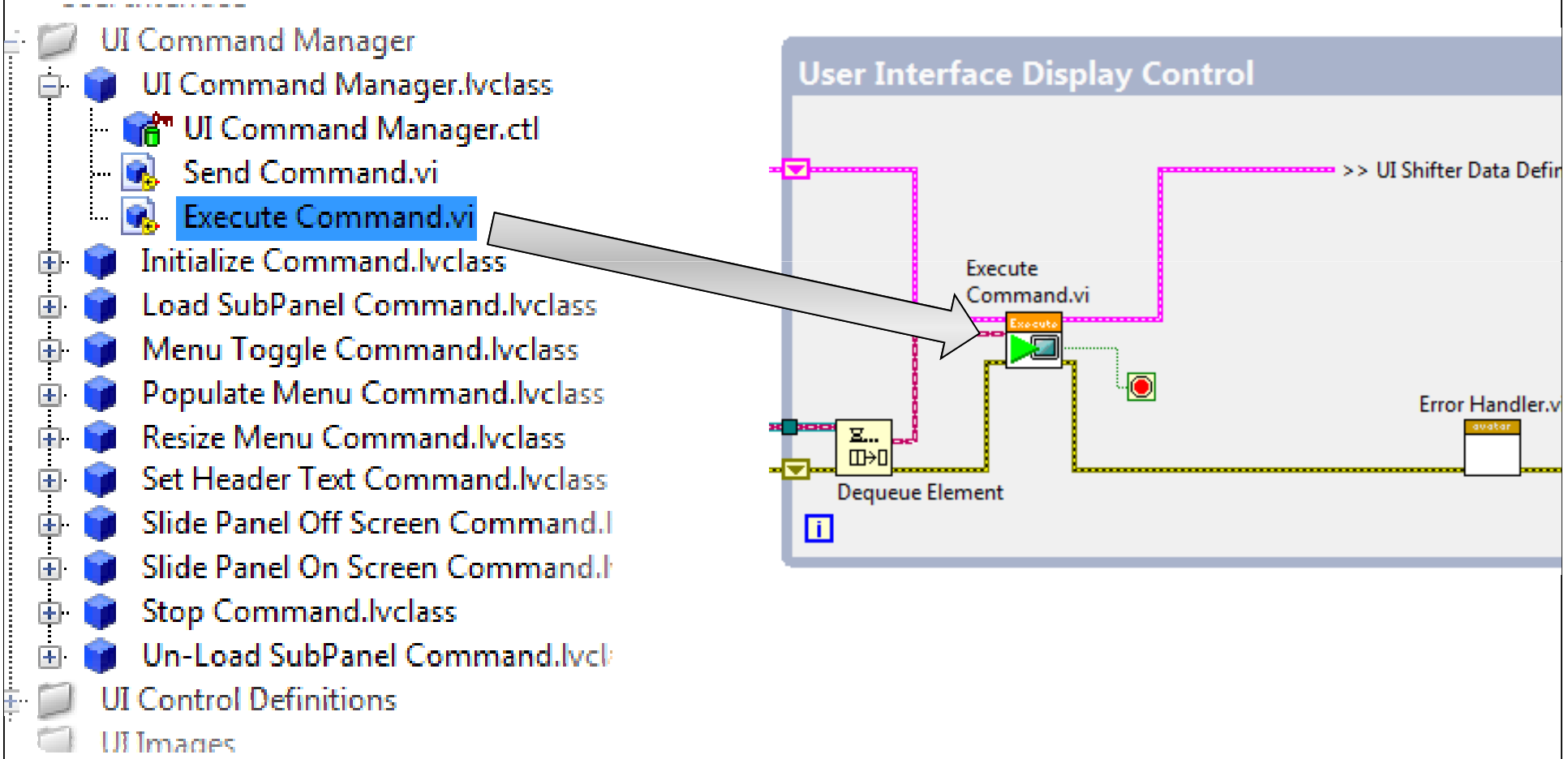
Objects represent the commands to be executed





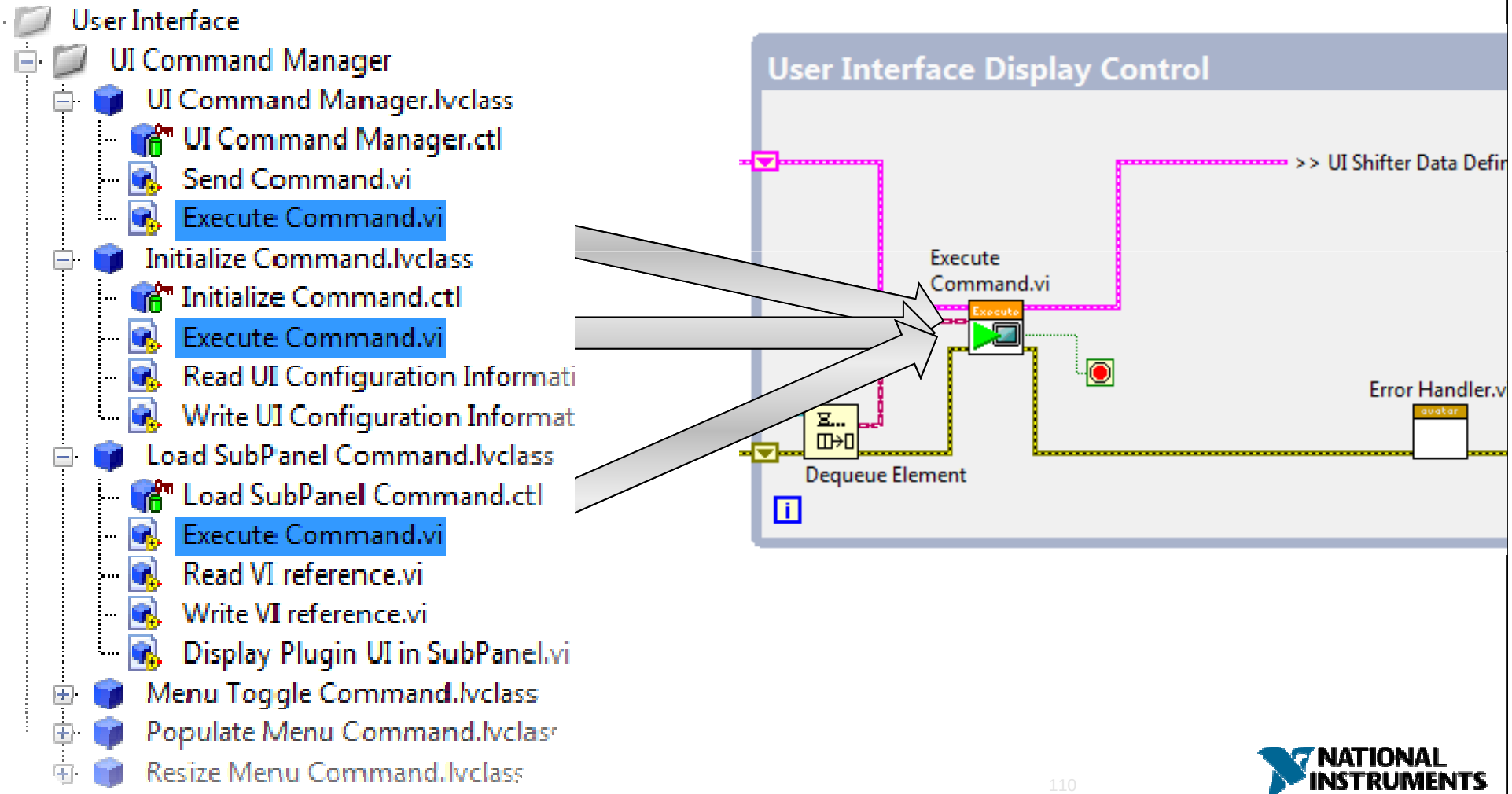
Command	Data
Initialize UI	Cluster containing configuration data
Populate Menu	Array of strings to display in menu
Resize Display	Array of integers [Width, Height]
Load Subpanel	Reference of VI to Load
Insert Header	String
Stop	-

Dynamic Dispatching Commands



NATIONAL INSTRUMENTS

Dynamic Dispatching Commands

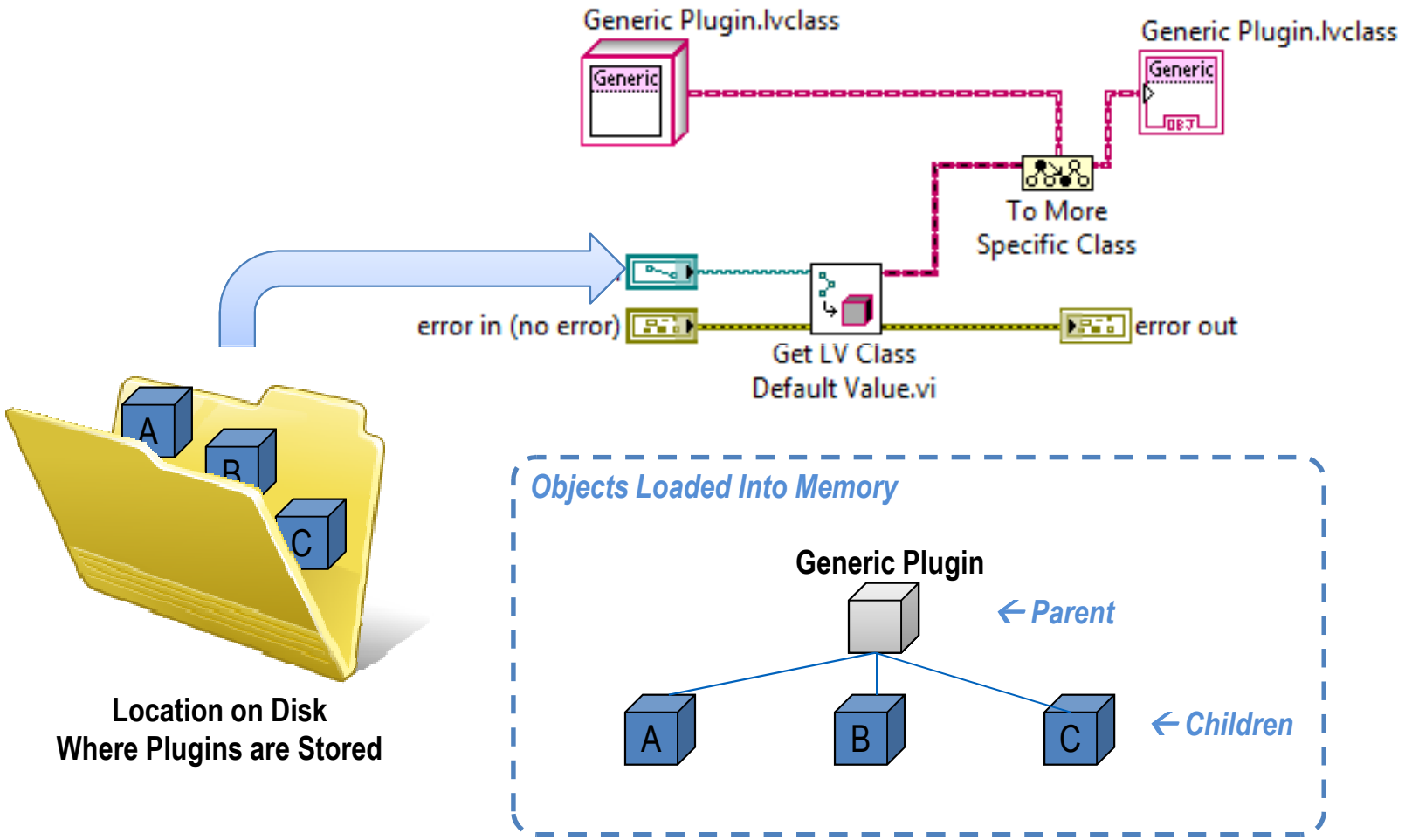


NATIONAL INSTRUMENTS

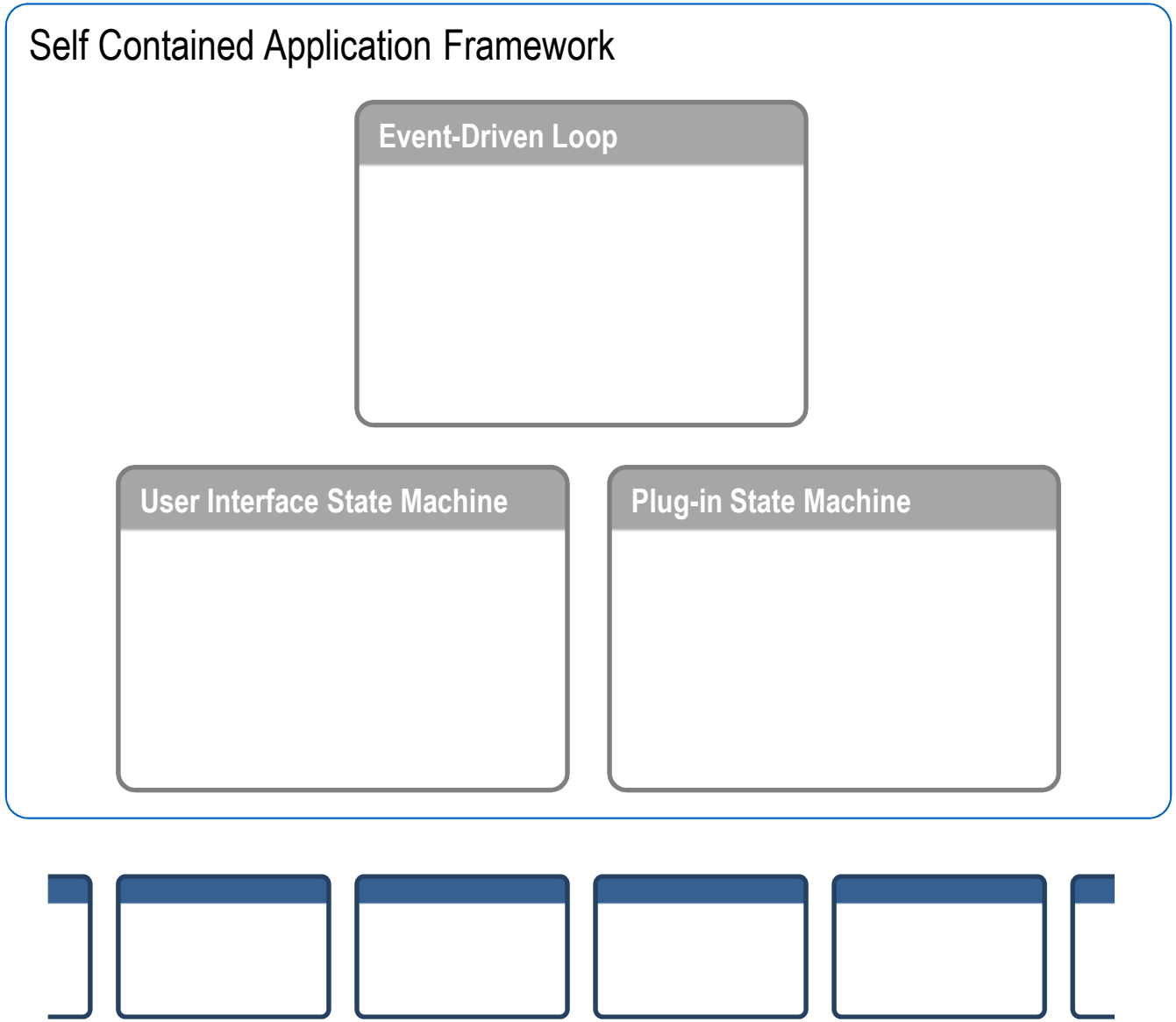
Creating and Sending a New Command

DEMO

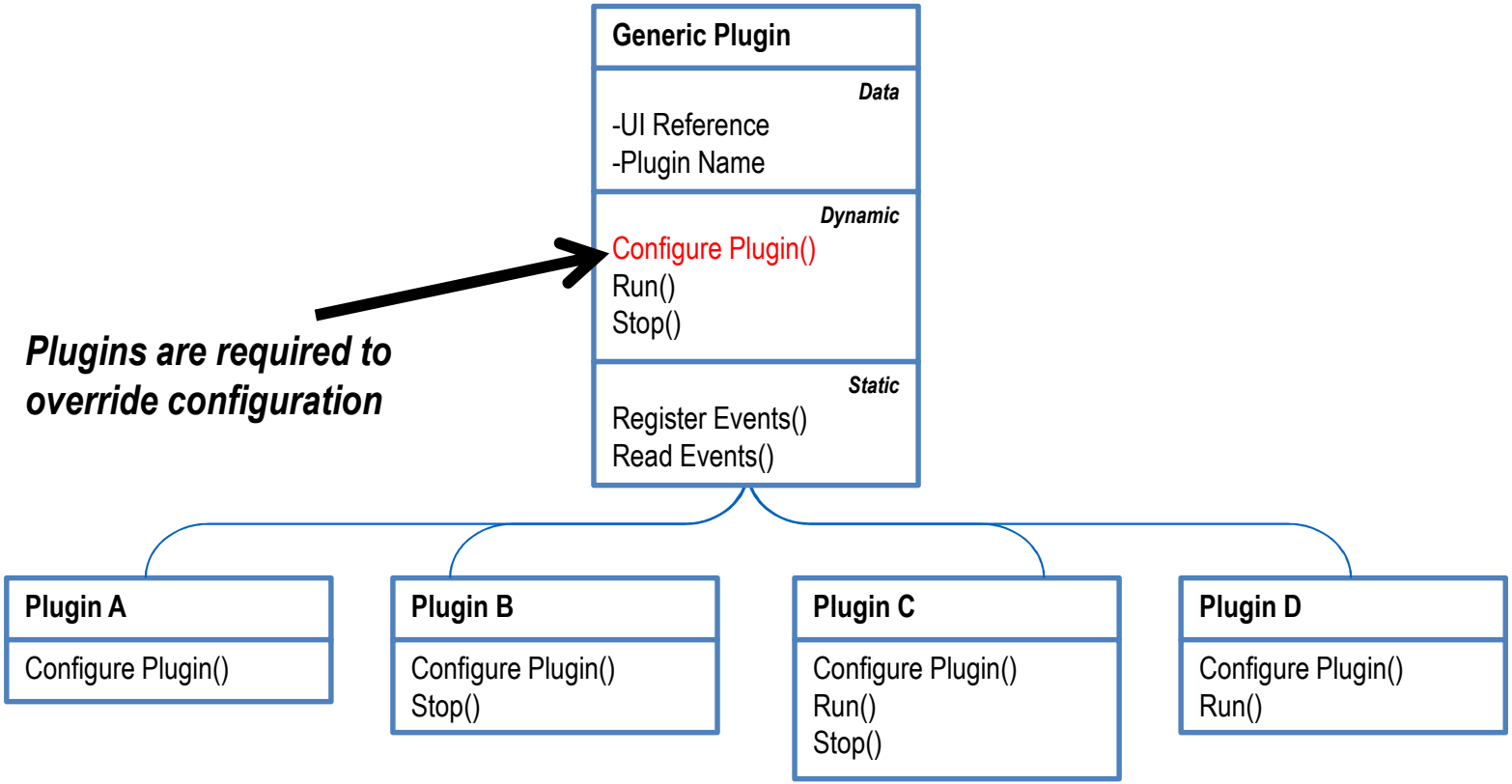
The Basics of an Object Factory



NATIONAL INSTRUMENTS



Factory Design Pattern Example

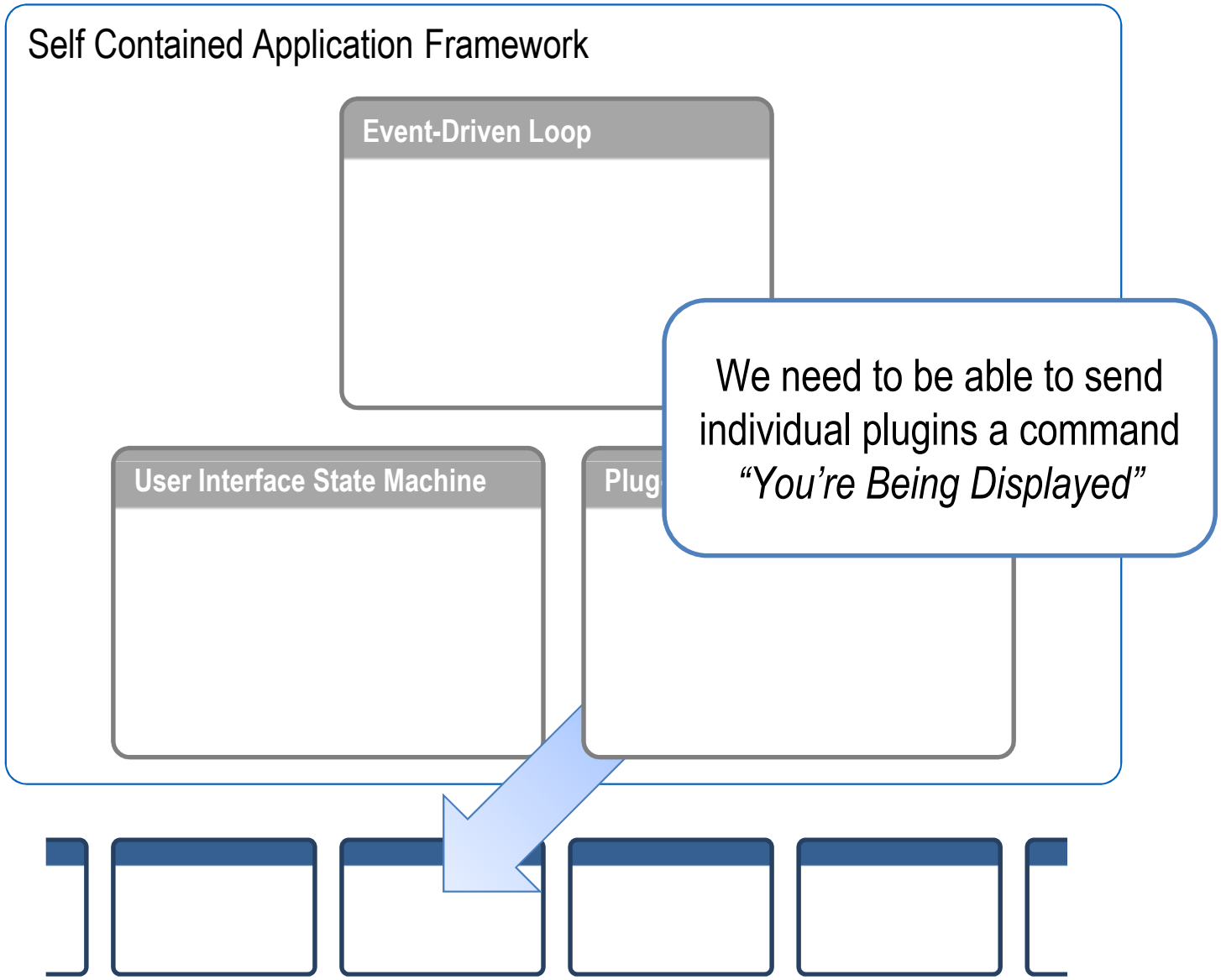


NATIONAL INSTRUMENTS

Creating a Plugin

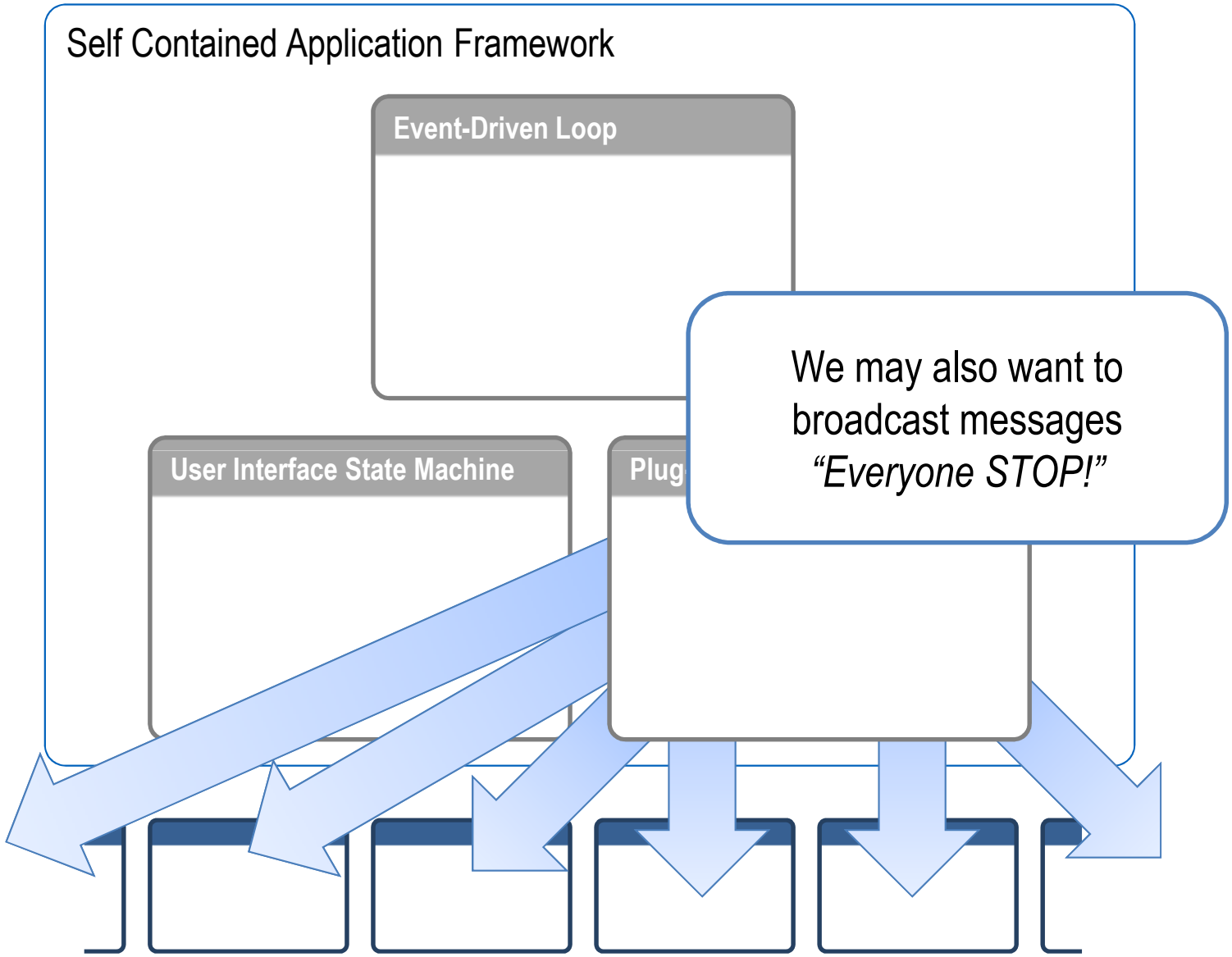
DEMO

Plugins Need To Receive Commands



NATIONAL INSTRUMENTS

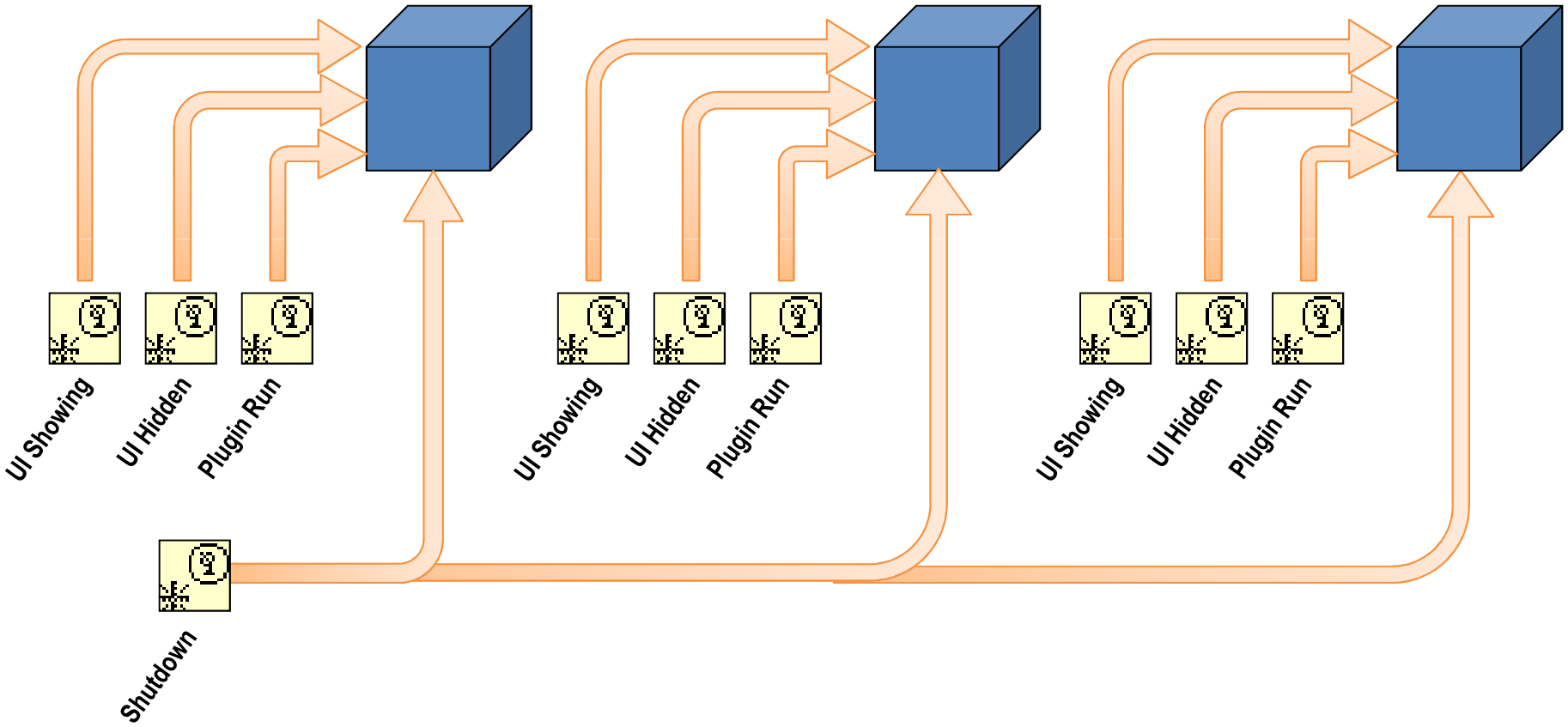
Plugins Need To Receive Commands



NATIONAL INSTRUMENTS

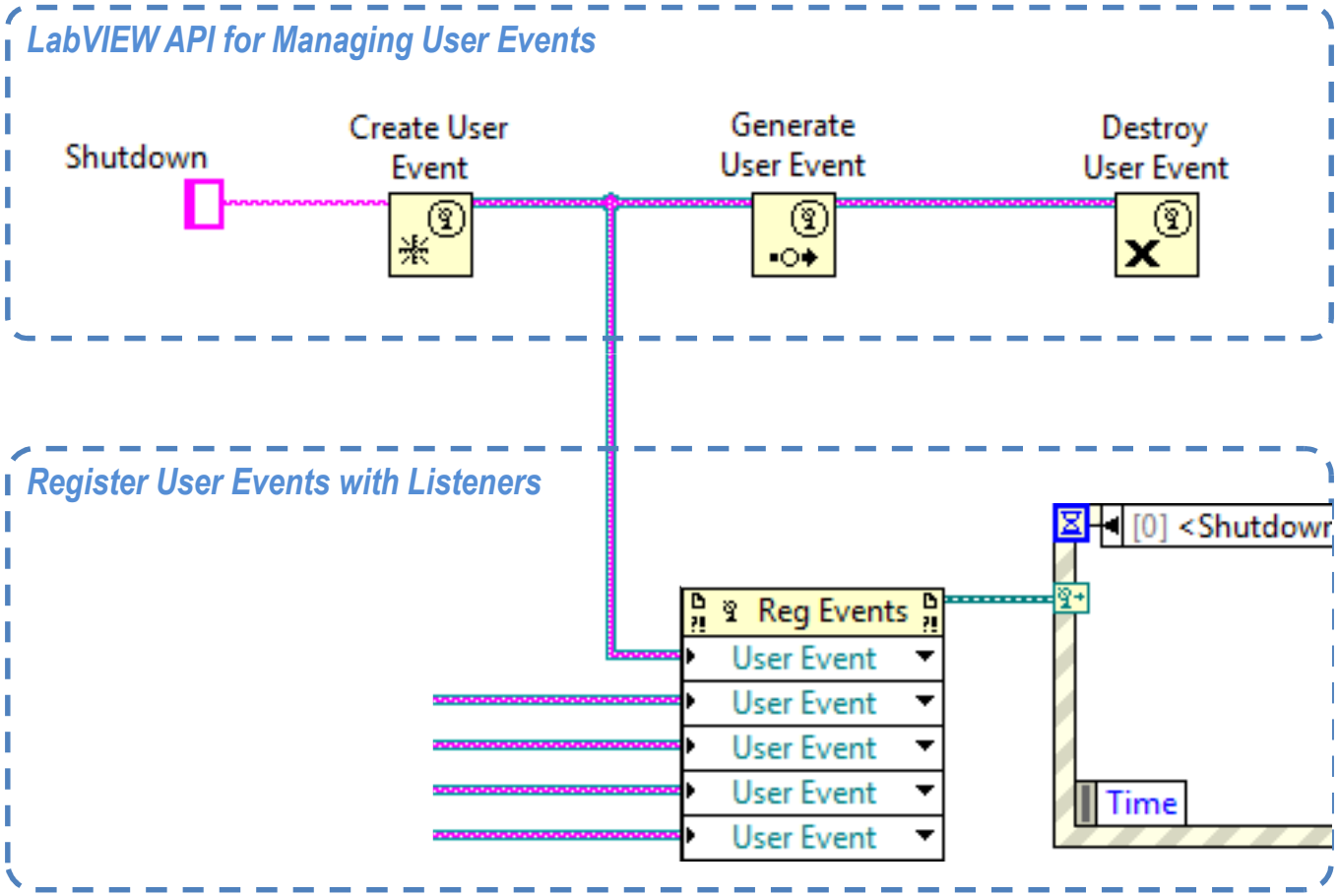
Registering Events with Plug-in Objects

Unique events are registered for messages that will be sent to specific plug-ins
The same event can be registered with multiple plugins in order to broadcast a message



NATIONAL INSTRUMENTS

Using User Events



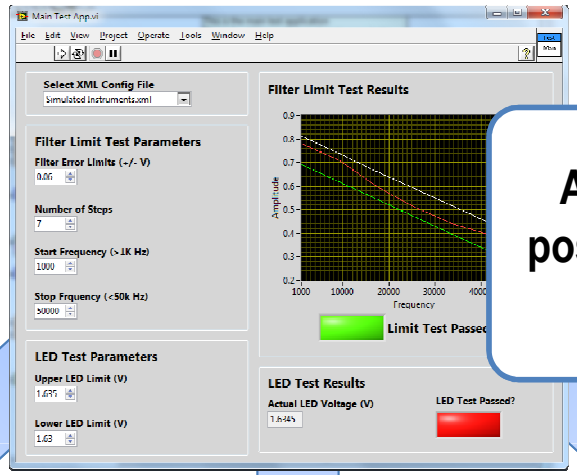
NATIONAL INSTRUMENTS

Sending Messages to Plugins Using Dynamic Events

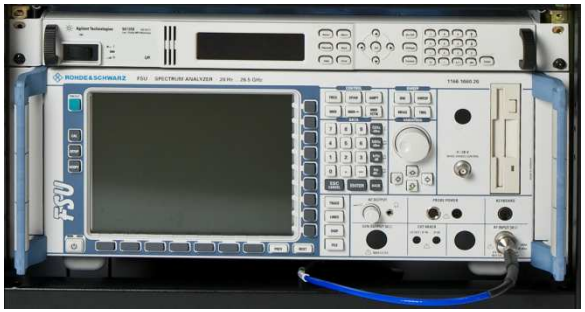
DEMO

Abstract the Hardware in Software

Mitigate Obsolescence with a Hardware Abstraction Layer (HAL)



A plugin architecture makes it possible to interchange hardware without modifying code



Traditional Benchtop

Modular PXI

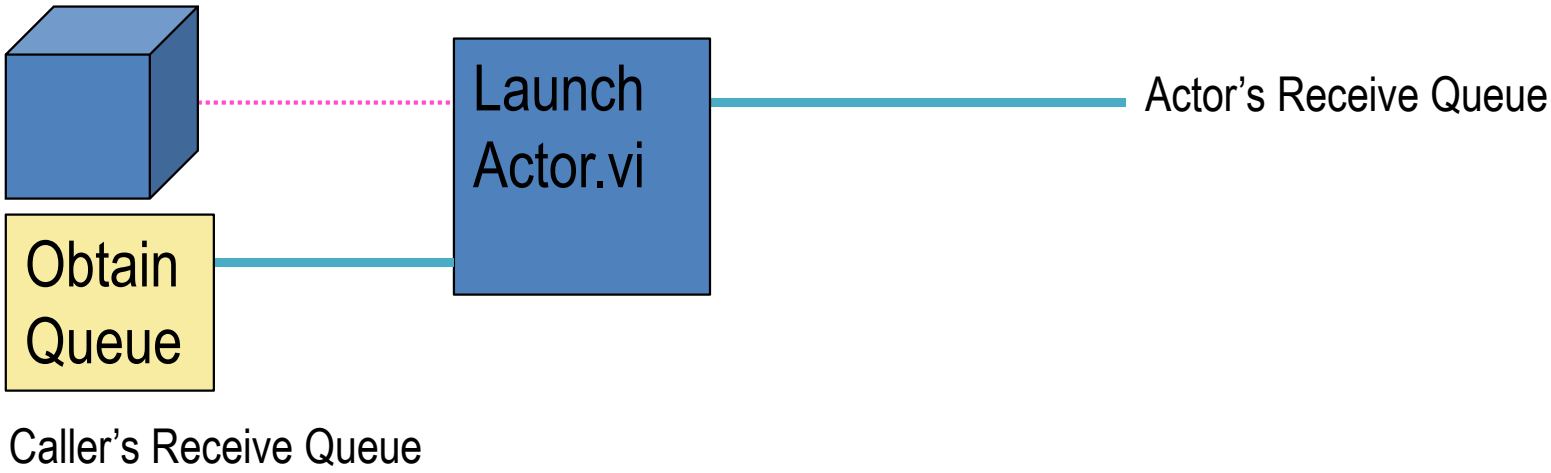
Simulated (software only)

NATIONAL INSTRUMENTS

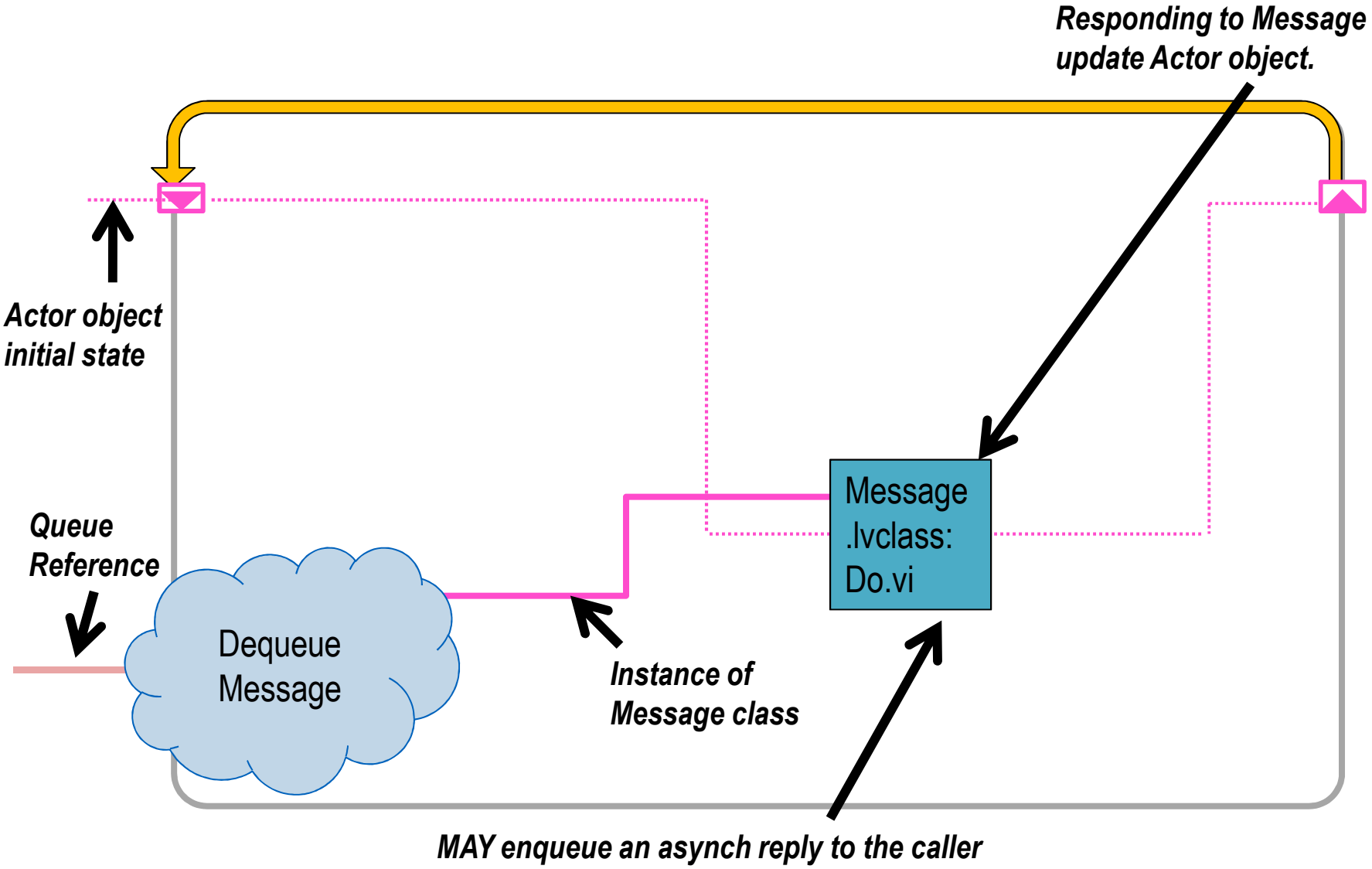
Hardware Abstraction Layers

DEMO

Launching an actor

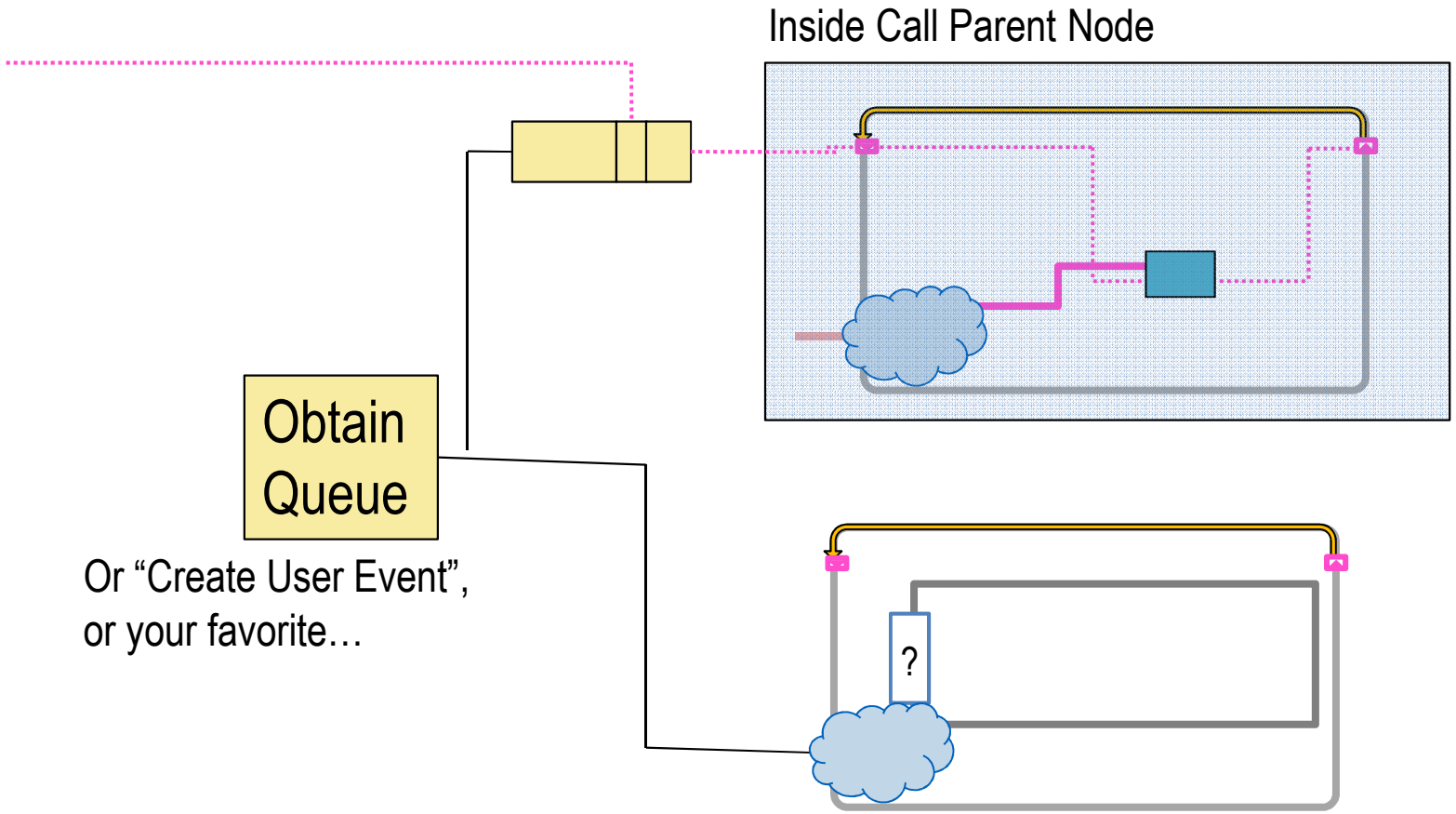


Actor.lvclass:Actor Core.vi



NATIONAL INSTRUMENTS

Actor Core.vi – Descendant Override



Actor Framework

DEMO

Section Four

Data Communication Best Practices

NATIONAL INSTRUMENTS

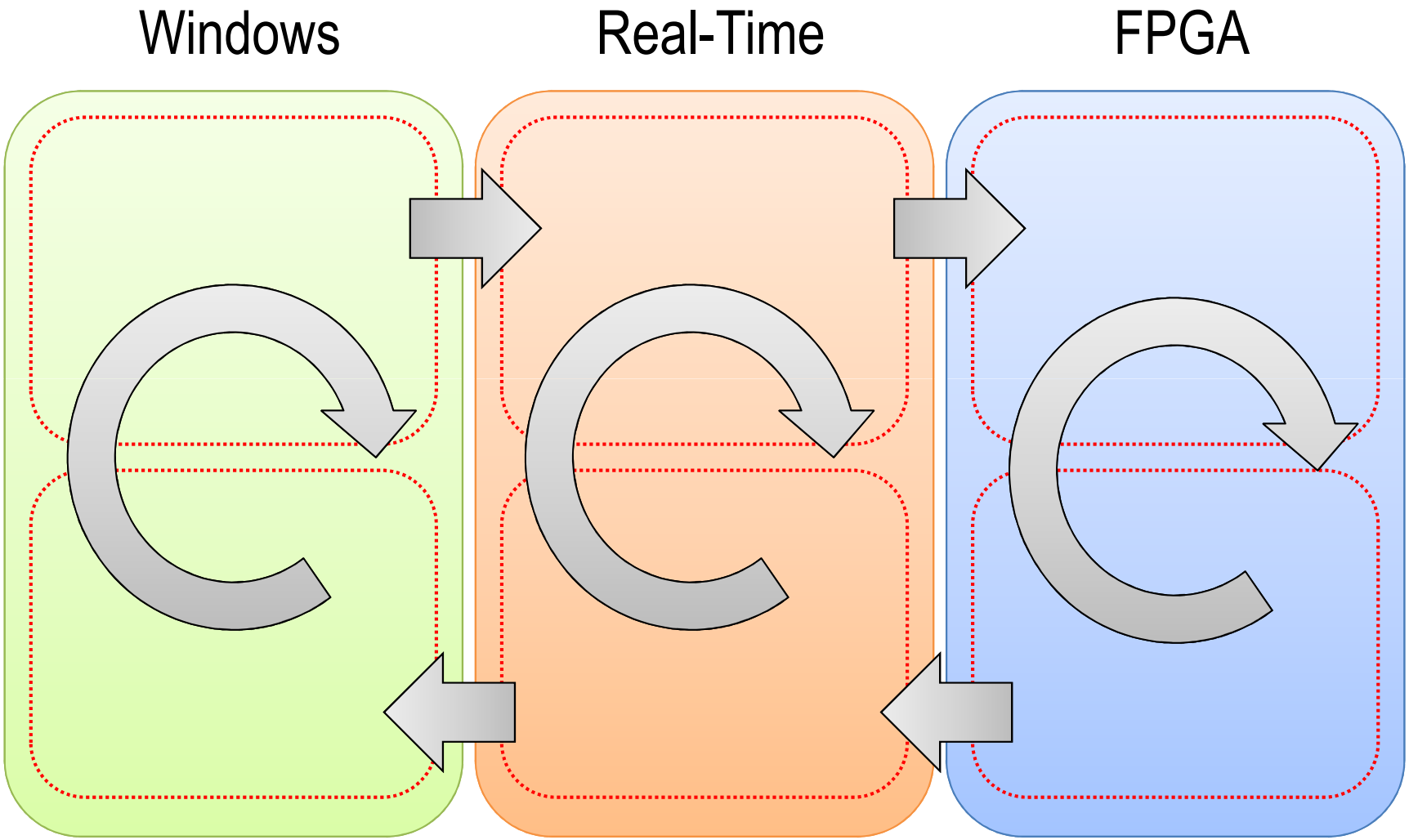
Data Communication Options in LabVIEW

1. TCP and UDP
2. Network Streams
3. Shared Variables
4. DMAs
5. Web Services
6. Peer-to-Peer Streaming
7. Queues
8. Dynamic Events
9. Functional Global Variables
10. RT FIFOs
11. Datasocket
12. Local Variables
13. Programmatic Front Panel Interface
14. Target-scoped FIFOs
15. Notifier
16. Simple TCP/IP Messaging (STM)
17. AMC
18. HTTP
19. FTP
20. Global variables

… just to name a few …

NATIONAL INSTRUMENTS

Communication Spans Processes and Targets



NATIONAL INSTRUMENTS

The Pitfalls of Variables

DEMO

Common Pitfalls of Data Communication

Race conditions- two requests made to the same shared resource

Deadlock- two or more depended processes are waiting for each other to release the same resource

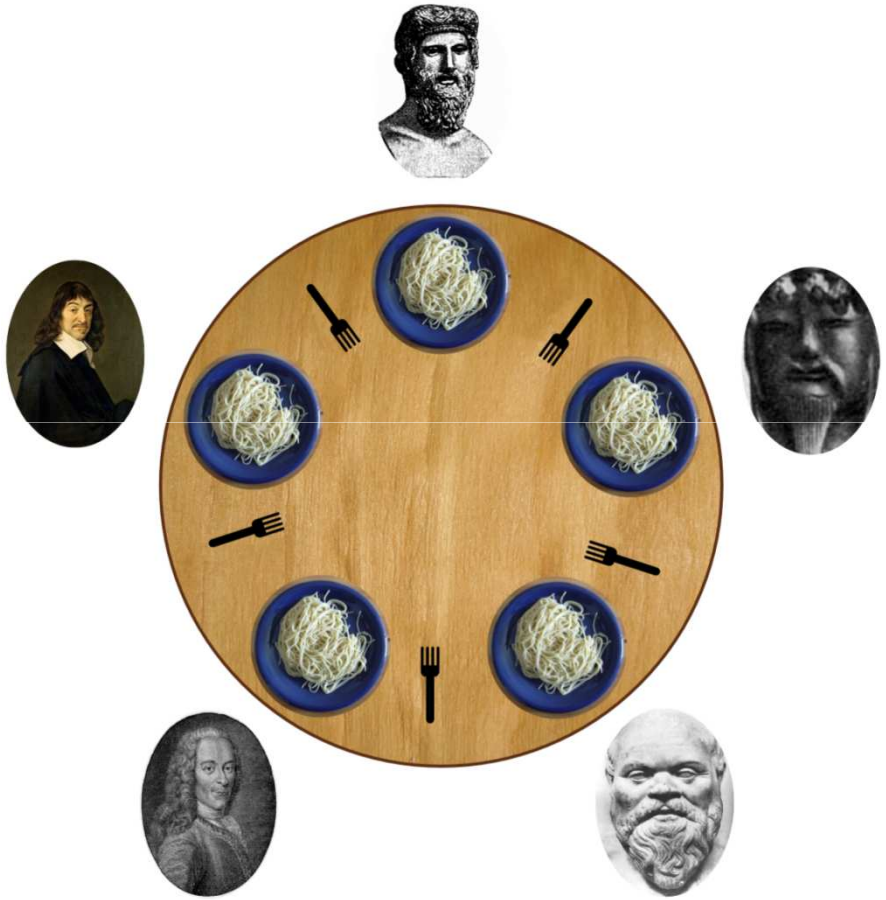
Data loss- gaps or discontinuities when transferring data

Performance degradation- poor processing speed due to dependencies on shared resources

Buffer overflows- writing to a buffer faster than it is read from the buffer

Stale data- reading the same data point more than once

The Dining Philosophers



NATIONAL INSTRUMENTS

Message/Command

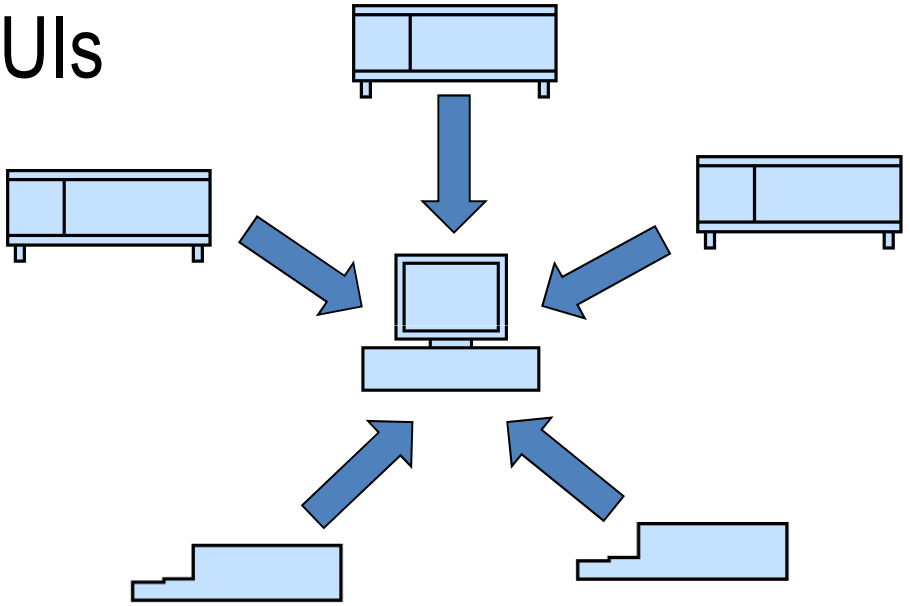
- Commander (Host) and Worker (Target) Systems
- Must be lossless* (can be buffered)
- Minimal latency
- Typically processed one at a time
- Reads are destructive
- Example: stop button, alarm, error

*some commands may need to pre-empt other commands based on priority

NATIONAL INSTRUMENTS

Update/Monitor

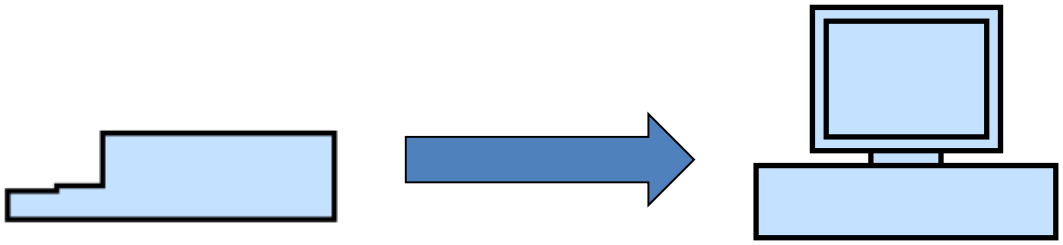
- Periodic transfer of latest value
- Often used for HMIs or GUIs
- N Targets: 1 Host
- Can be lossy
- Non-buffered



Example: monitoring current engine temperature

Stream/Buffer

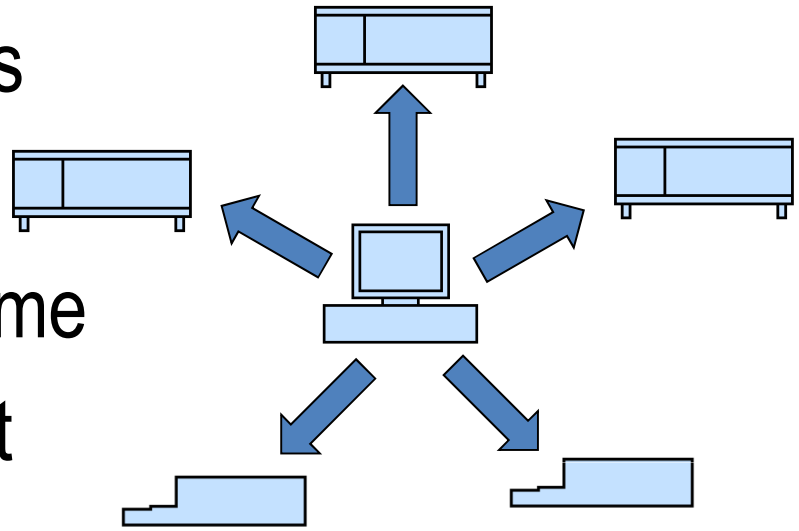
- Continuous transfer, but not deterministic
- High throughput
- No data loss, buffered
- 1 Target: 1 Host; Unidirectional



Example: High speed acquisition on target, sent to host PC for data logging

Variable/Tag

- Set Points and PID Constants
- Initial configuration data
- Can be updated during run-time
- Only latest value is of interest
- 1 Host: N Targets



Example: reading/writing the set-point of a thermostat,
.ini configuration files

NATIONAL INSTRUMENTS

Choosing Transfer Types

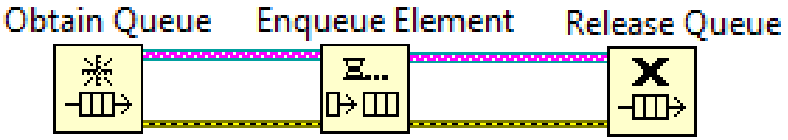
	Message	Update	Stream	Variable (Tag)
Examples	• Exec Action • Error	• Heartbeat • Movie	• Waveform • Image	• Setpoint
Fundamental Features	• Buffering • Blocking (Timeout) • Single-Read	• Nonhistorical • Blocking (Timeout)	• Buffering • Blocking (Timeout)	• Nonhistorical
Optional Features	• Ack	• Broadcast	• Multi-layer Buffering	• Dynamic Lookup • Group Mgmt • Latching
Performance	• Low-Latency	• Low-Latency	• High-Throughput	• Low-Latency • High-Count
Configuration	• N Targets: 1 Host	• N Targets:1 Host	• 1 Target:1 Host • Unidirectional	• N Targets: 1 Host

NATIONAL INSTRUMENTS

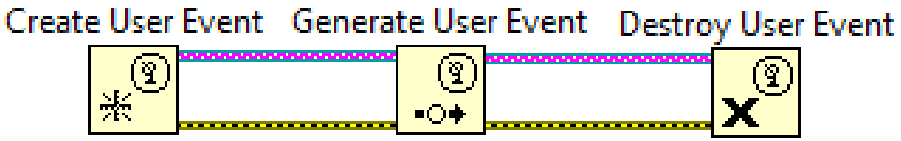
Inter-process Communication Options

Shared Variables
Update GUI loop with latest value

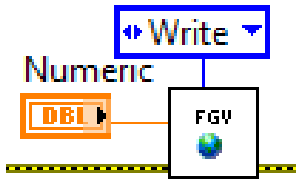
Queues
Stream continuous data between loops on a non-deterministic target



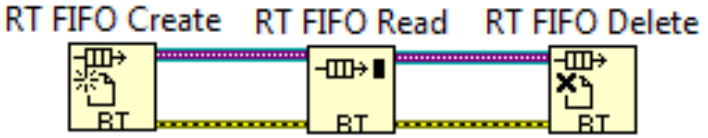
Dynamic Events
Register Dynamic Events to execute sections of code



Functional Global Variables (FGV)
Use a non-reentrant subVI to protect critical data



RT FIFOs
Stream continuous data between time critical loops on a single RT target



NATIONAL INSTRUMENTS

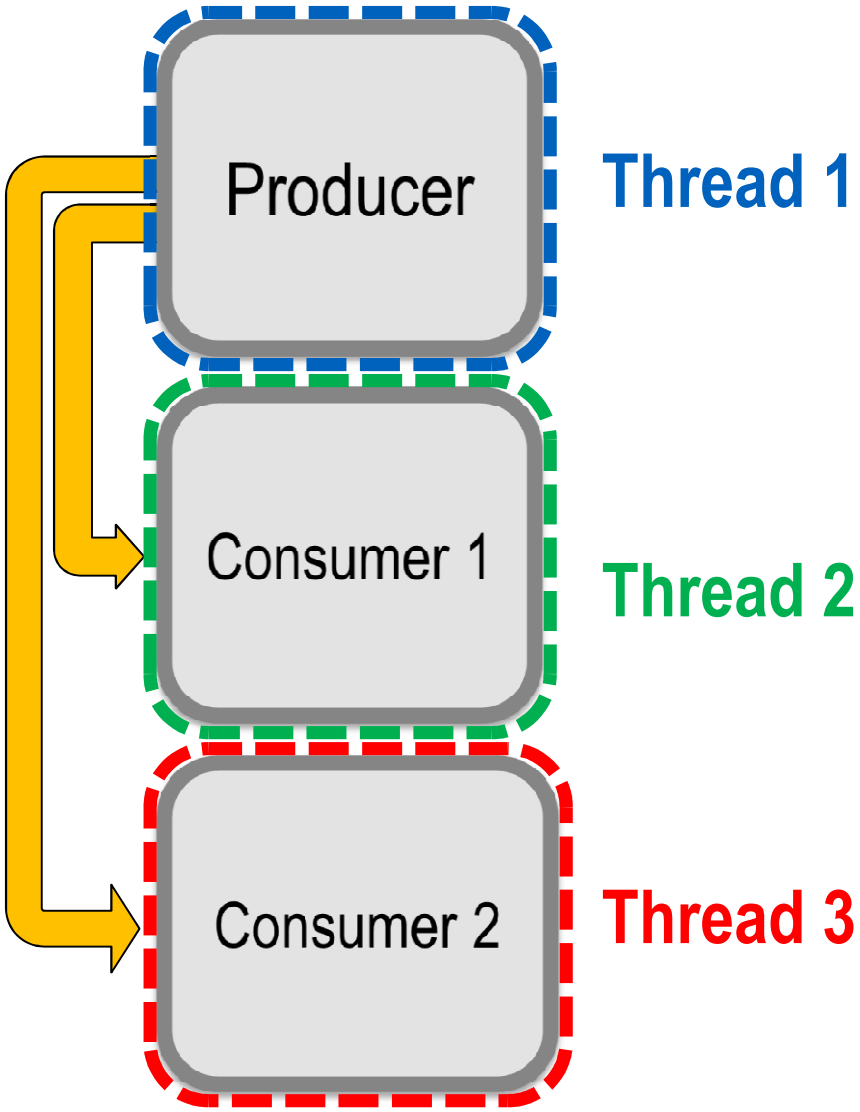
Producer Consumer

Best Practices

1. One consumer per queue
2. Keep at least one reference to a named queue available at any time
3. Consumers can be their own producers
4. Do not use variables

Considerations

1. How do you stop all loops?
2. What data should the queue send?



NATIONAL INSTRUMENTS

LabVIEW FIFOs

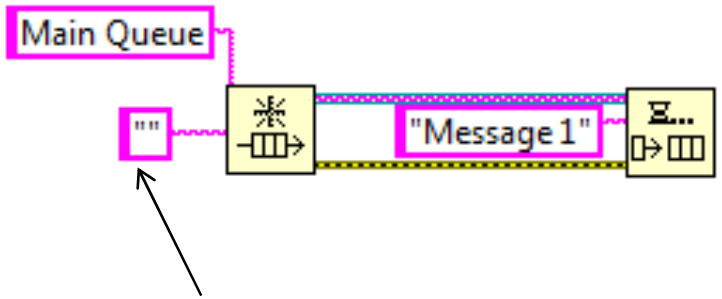
- Queues
- RT FIFOs
- Network Streams
- DMAs
- User Events

In general, FIFOs are good if you need lossless communication that preserves historical information

NATIONAL INSTRUMENTS

Queues

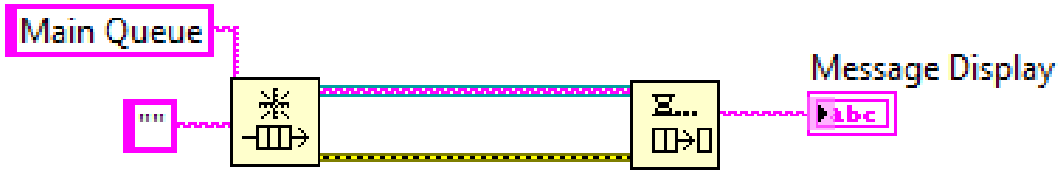
Adding Elements to the Queue



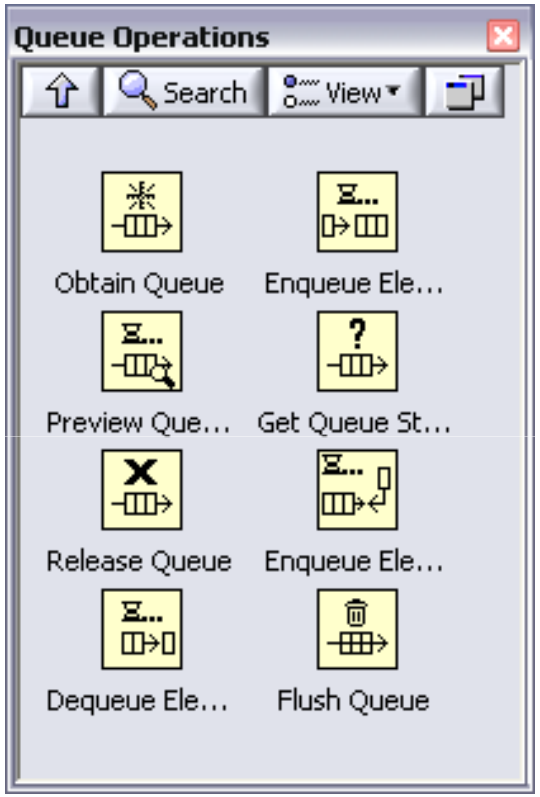
Select the data type the queue will hold

Reference to existing queue in memory

Dequeueing Elements



Dequeue will wait for data or time-out (defaults to -1)



NATIONAL INSTRUMENTS

Choosing Transfer Types for Inter-process

	Message	Update	Stream	Variable (Tag)
Windows	• Queue • Shared Variable (Blocking, Buffered)	• SE Queue • Notifier • Shared Variable (Blocking)	• Queue • Shared Variable (Blocking, Buffered)	• Local/Global Variable • SE Queue • FGV • Shared Variable • DVR
RT	• Same as Windows • RT FIFO	• Same as Windows • SE RT FIFO	• Same as Windows • RT FIFO	• Same as Windows
FPGA	• FIFO (2009)	• SE FIFO (2009)	• FIFO	• Local/Global Variable • FGV

NATIONAL INSTRUMENTS

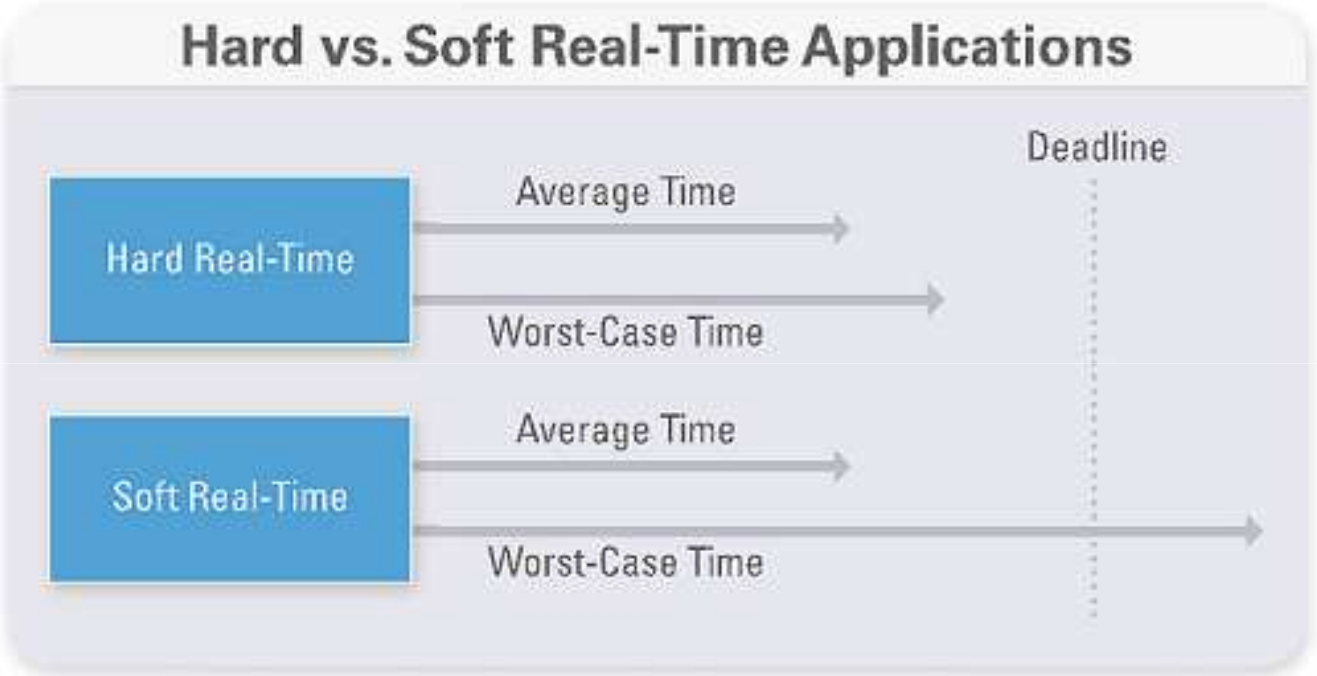
RT FIFOs vs. Queues

- Queues can handle string, variant, and other variable size data types, while RT FIFOs can not
- RT FIFOs are pre-determined in size, queues can grow as elements are added to them
- Queues use blocking calls when reading/writing to a shared resource, RT FIFOs do not
- RT FIFOs do not handle errors, but can produce and propagate them

Key Takeaway:

RT FIFOs are more deterministic for the above reasons

What is Determinism?



Determinism: An application (or critical piece of an application) that runs on a hard real-time operating system is referred to as deterministic if its timing can be guaranteed within a certain margin of error.

NATIONAL INSTRUMENTS

LabVIEW Real-Time Hardware Targets

LabVIEW Real-Time

CompactRIO

PXI

Desktop or Industrial PC

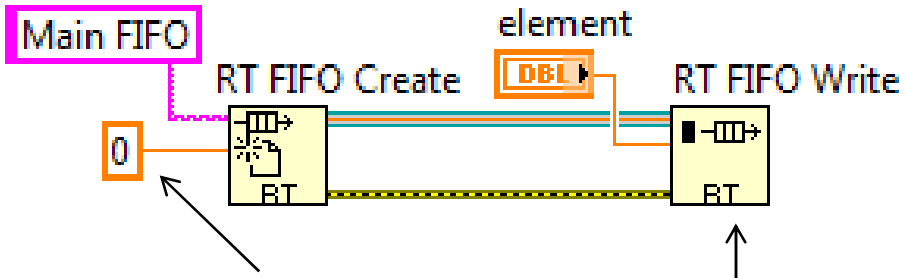
Vision Systems

Single-Board RIO

NATIONAL INSTRUMENTS

RT FIFOs

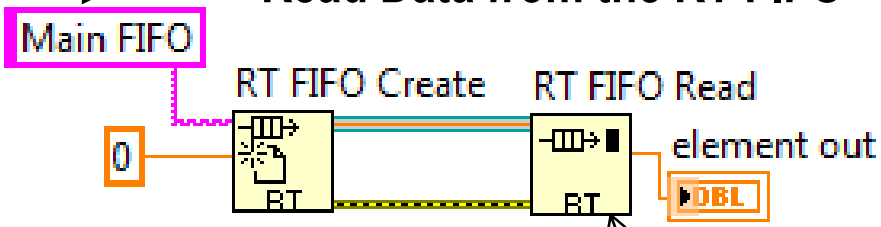
Write Data to the RT FIFO



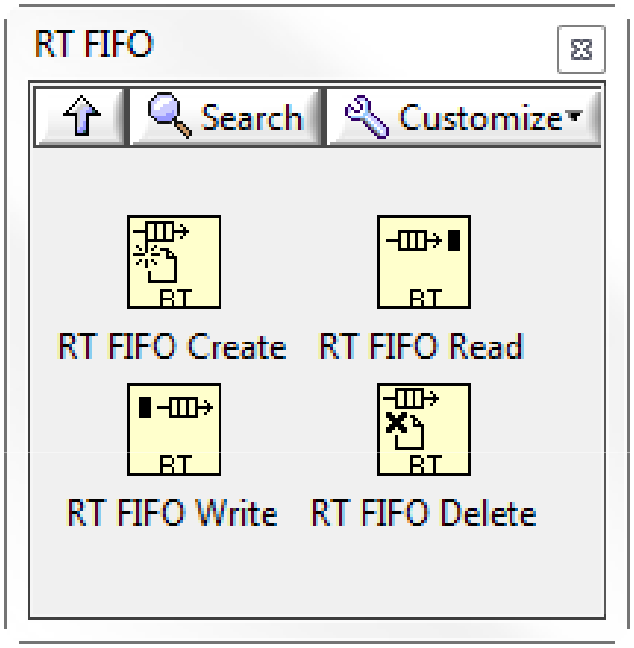
Select the data type the RT FIFO will hold

Reference to existing RT FIFO in memory

Read Data from the RT FIFO



Read/Write wait for data or time-out (defaults to 0)
Write can overwrite data on a timeout condition

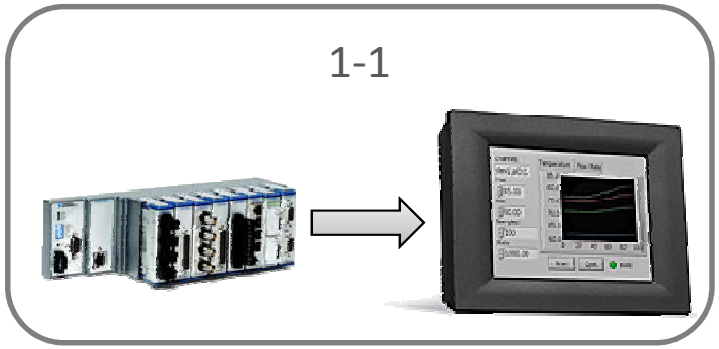


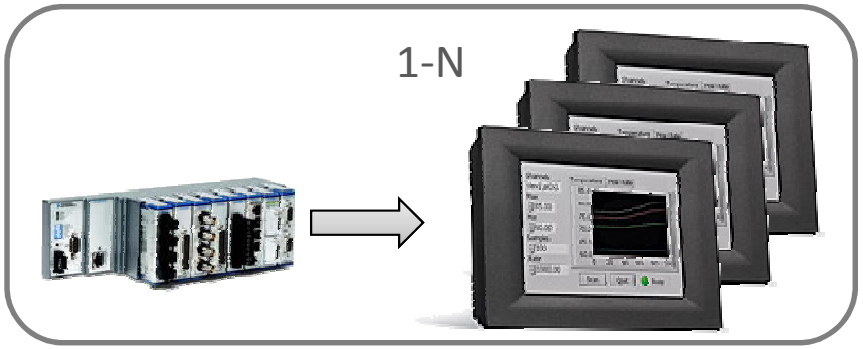
NATIONAL INSTRUMENTS

Common Network Transfer Policies

“Latest Value” or “Network Publishing”

- Making the current value of a data item available on the network to one or many clients
- Examples
  - I/O variables publishing to an HMI for monitoring
  - Logging temperature values on a remote PC
- Values persist until over written by a new value
- Lossy – client only cares about the latest value





Latest Value Communication

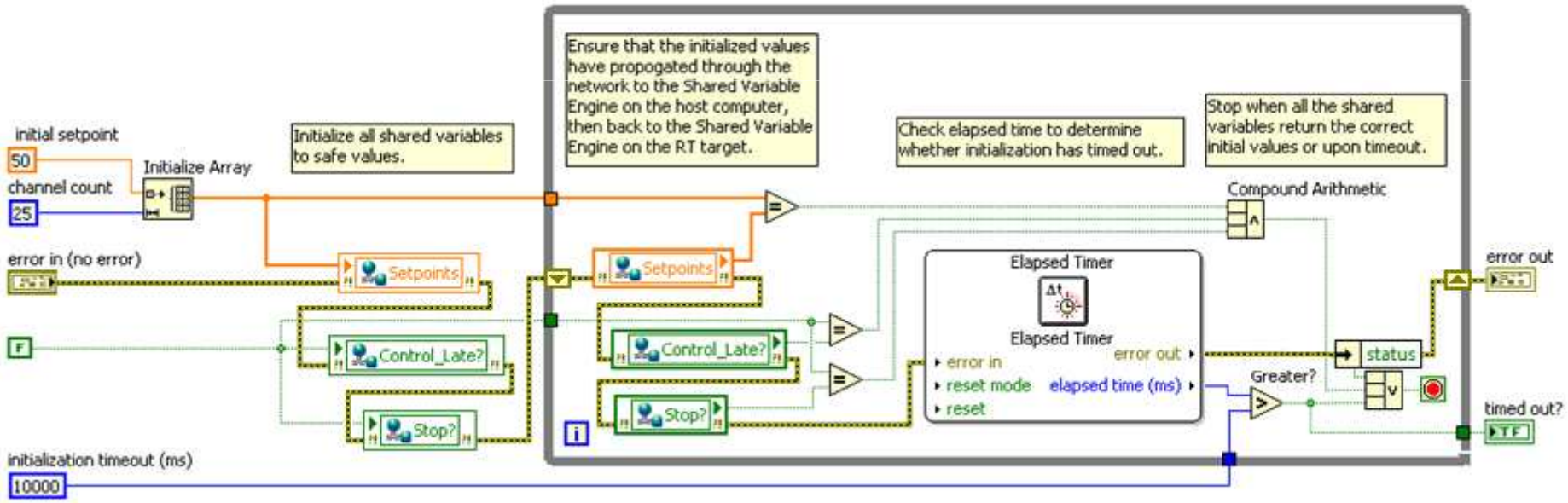
API	Type	Performance	Ease of Use	Supported Configurations	3rd Party APIs?
Shared Variable*	LabVIEW Feature			1:1, 1:N, N:1	• Measurement Studio • CVI

***Network buffering should be disabled**

Using Shared Variables Effectively

Programming Best Practices:

- Initialize shared variables
- Serialize shared variable execution
- Avoid reading stale shared variable data

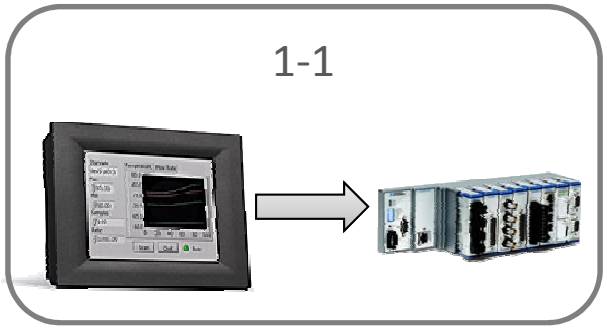


NATIONAL INSTRUMENTS

Common Network Transfer Policies

“Streaming”

- Sending a lossless stream of information
- Examples
  - Offloading waveform data from cRIO to remote PC for intensive processing
  - Sending waveform data over the network for remote storage
- Values don’t persist (reads are destructive)
- Lossless – client must receive all of the data
- High-throughput required (latency not important)



Streaming Lossless Data

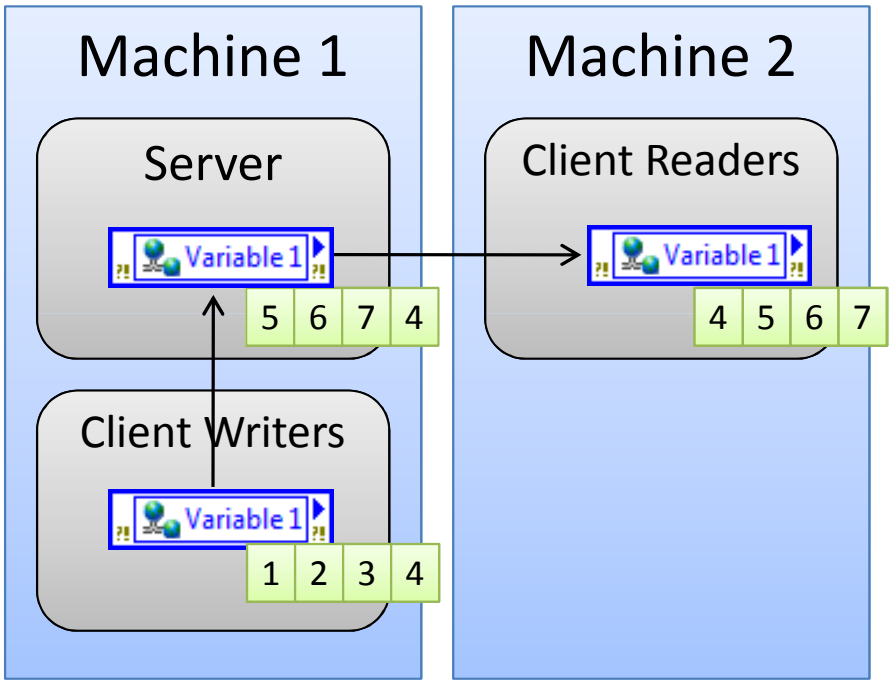
API	Type	Performance	Ease of Use	Supported Configurations	3rd Party APIs?
Network Streams NEW!	LabVIEW Feature			1:1	Not this year

What about the shared variable with buffering enabled?

NO!

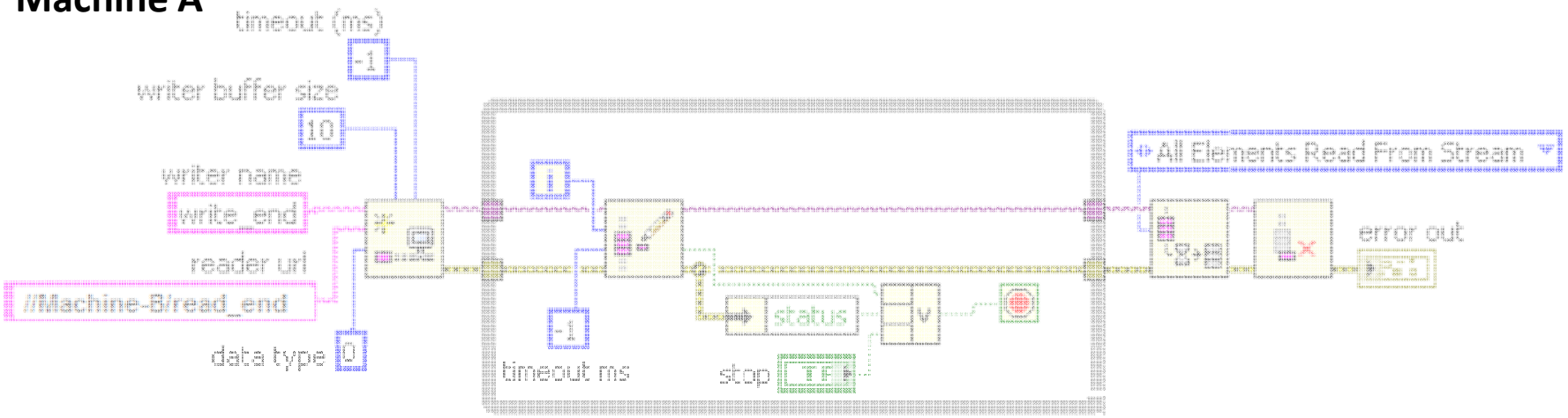
Pitfalls of Streaming with Variables

- Lack of flow control can result in data loss
- Data may be lost if the TCP/IP connection is dropped
- Data loss does not result in an error, only a warning

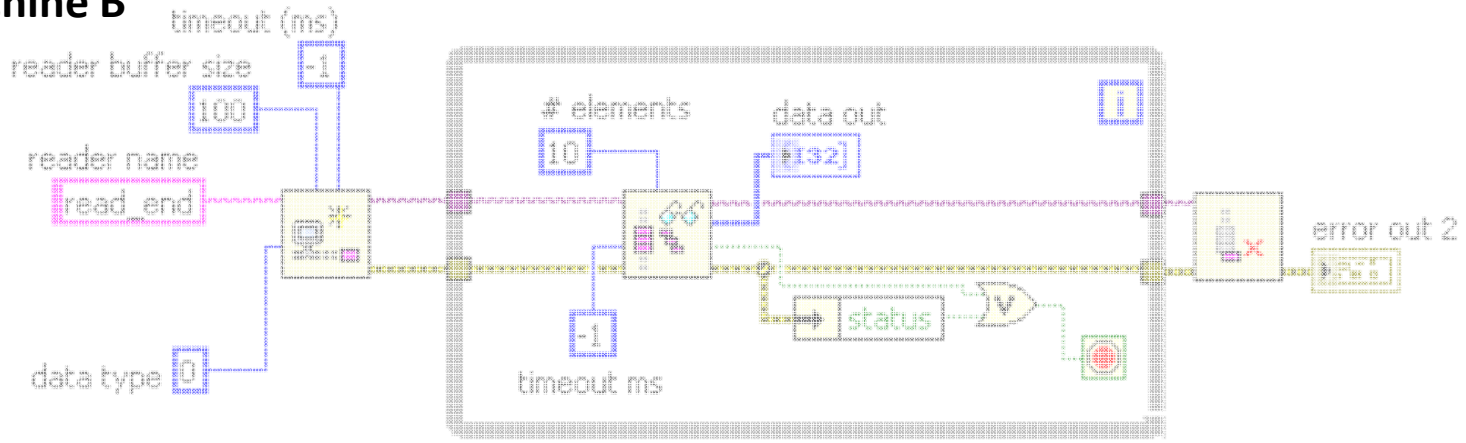


Network Streams **NEW!**

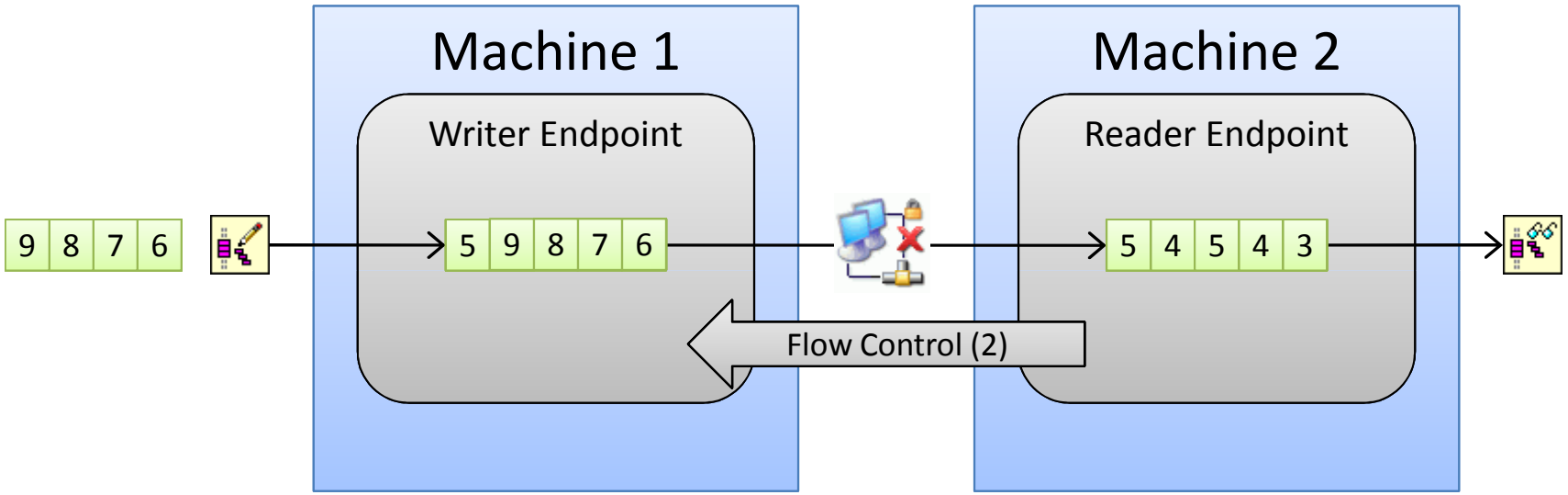
Machine A



Machine B



Network Streams in Action



Use Streams!

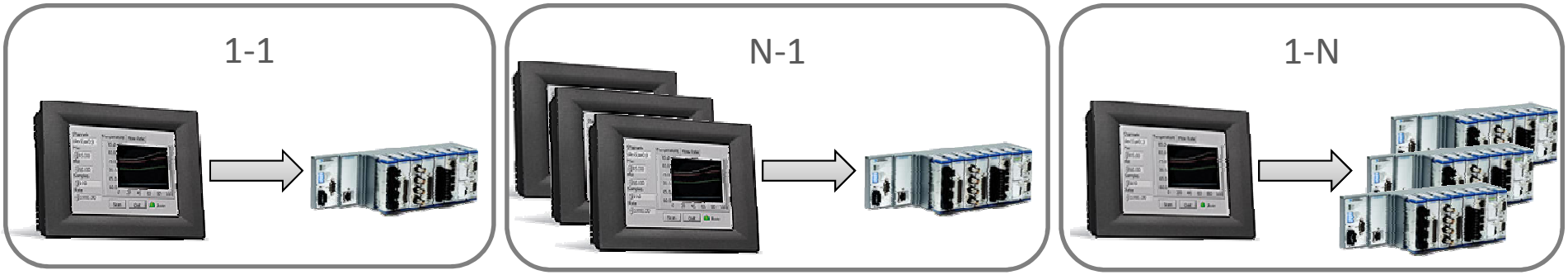
Demonstration

Inter-target Communication Using Network Streams

Common Network Transfer Policies

“Command” or “Message”

- Requesting an action from a worker
- Examples
  - Requesting an autonomous vehicle to move to a given position
  - Telling a process controller to begin its recipe
- Values don’t persist (reads are destructive)
- Lossless – client must receive every command
- Low latency – deliver the command as fast as possible

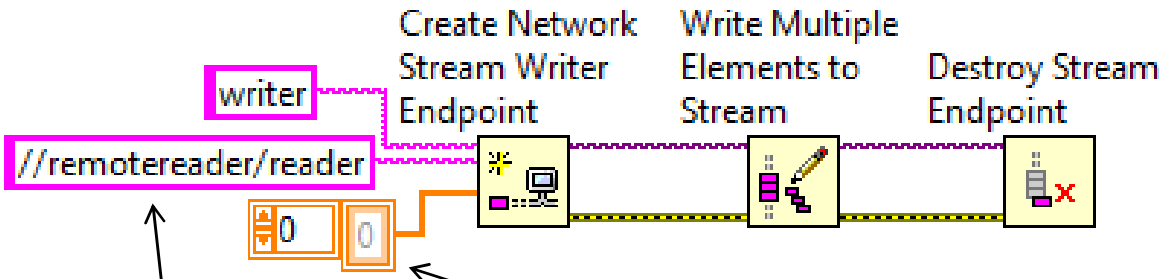


Network Command Mechanisms

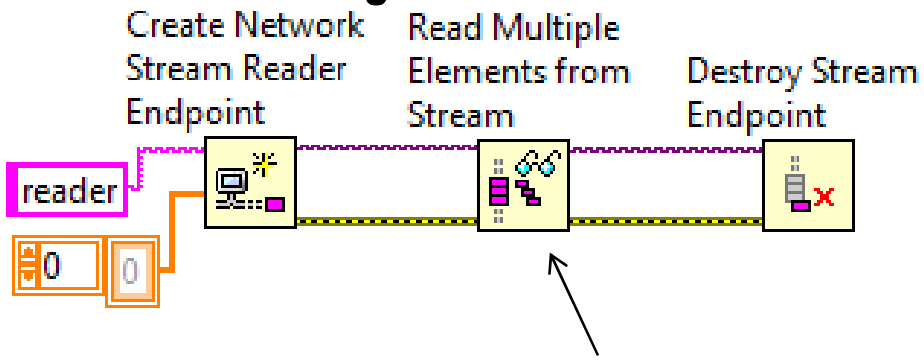
API	Type	Performance	Ease of Use	Supported Configurations	3rd Party APIs?
Network Streams	LabVIEW Feature			1:1	No

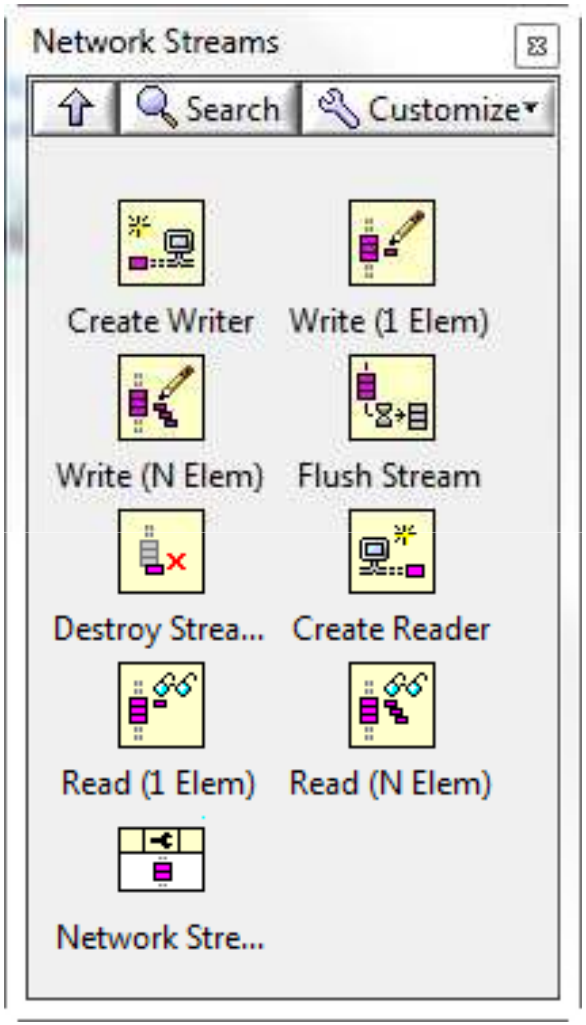
Network Streams

Writing Elements to the Stream



Reading Elements from Stream





NATIONAL INSTRUMENTS

Network Streams

- Lossless transfer, even in connection loss*
- Can be tuned for high-throughput (streaming) or low-latency (messaging)
- Unidirectional, P2P, LabVIEW only
- Not deterministic

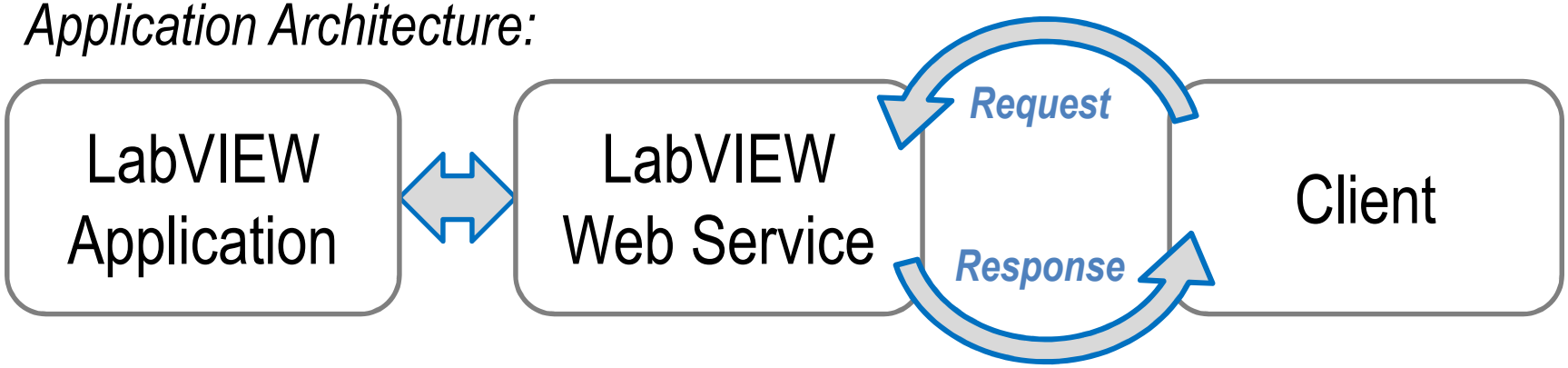
NATIONAL INSTRUMENTS

Demonstration

Introduction to Direct Memory Access

LabVIEW Web Services

Application Architecture:

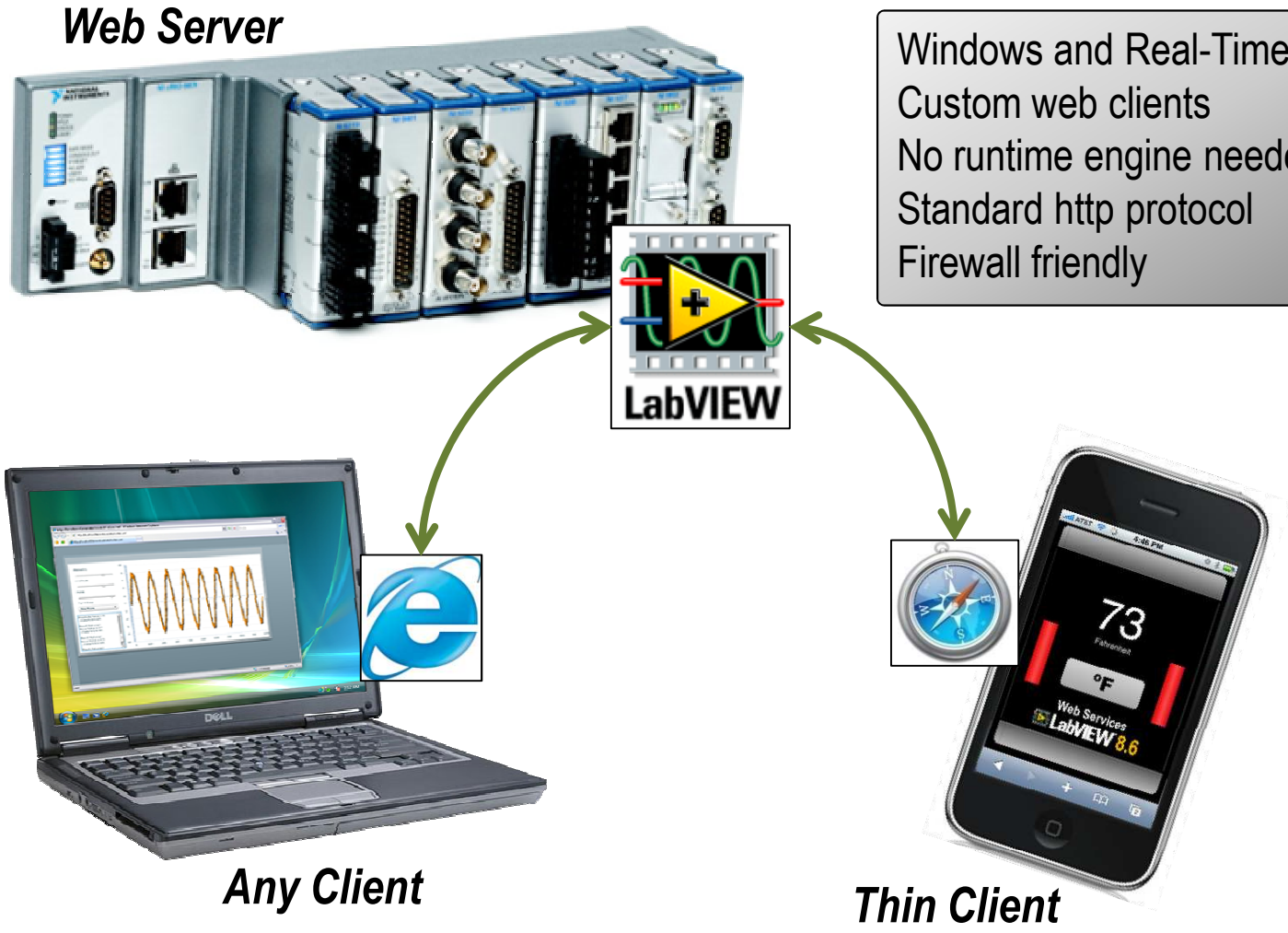


Sending Requests via URL:

Physical Location of Server
Name of Web Service
Mapping to a VI
Terminal Inputs (Optional)

NATIONAL INSTRUMENTS

Web Services in LabVIEW

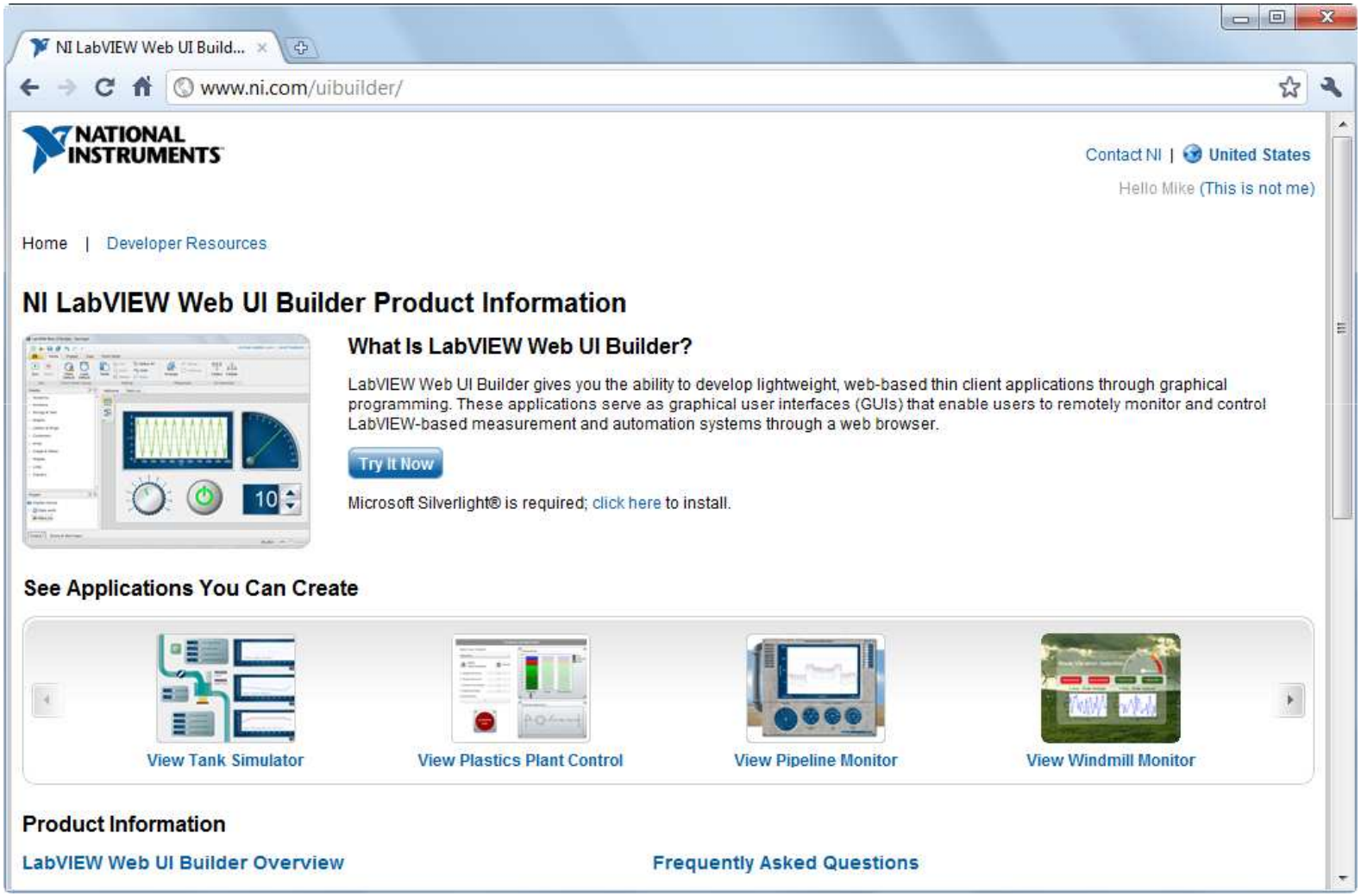


NATIONAL INSTRUMENTS

Demonstration

Basic Web Services

ni.com/uibuilder



NATIONAL INSTRUMENTS

Demonstration

Thin-Client Web Interfaces

Early Access Release Details

- Anyone can evaluate for free
  - Fully functional except for ‘Build and Deploy’
  - License for ‘Build and Deploy’ is $1,499 per user
  - License is sold as one-year software lease
- Not part of Developer Suite or Partner Lease

NATIONAL INSTRUMENTS

Inter-Target Communication Options

TCP/IP and UDP

Define low-level communication protocols to optimize throughput and latency

Shared Variables

Access latest value for a network published variable

Network Streams

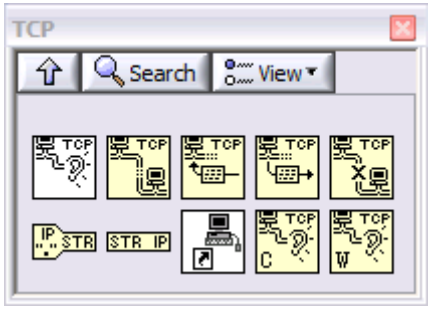
Point to Point streaming in LabVIEW with high throughput and minimal coding

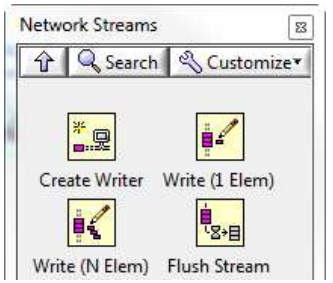
Web UI Builder

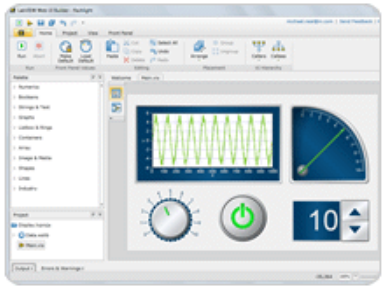
Create a thin client to communicate with a LabVIEW Web Service

DMAs

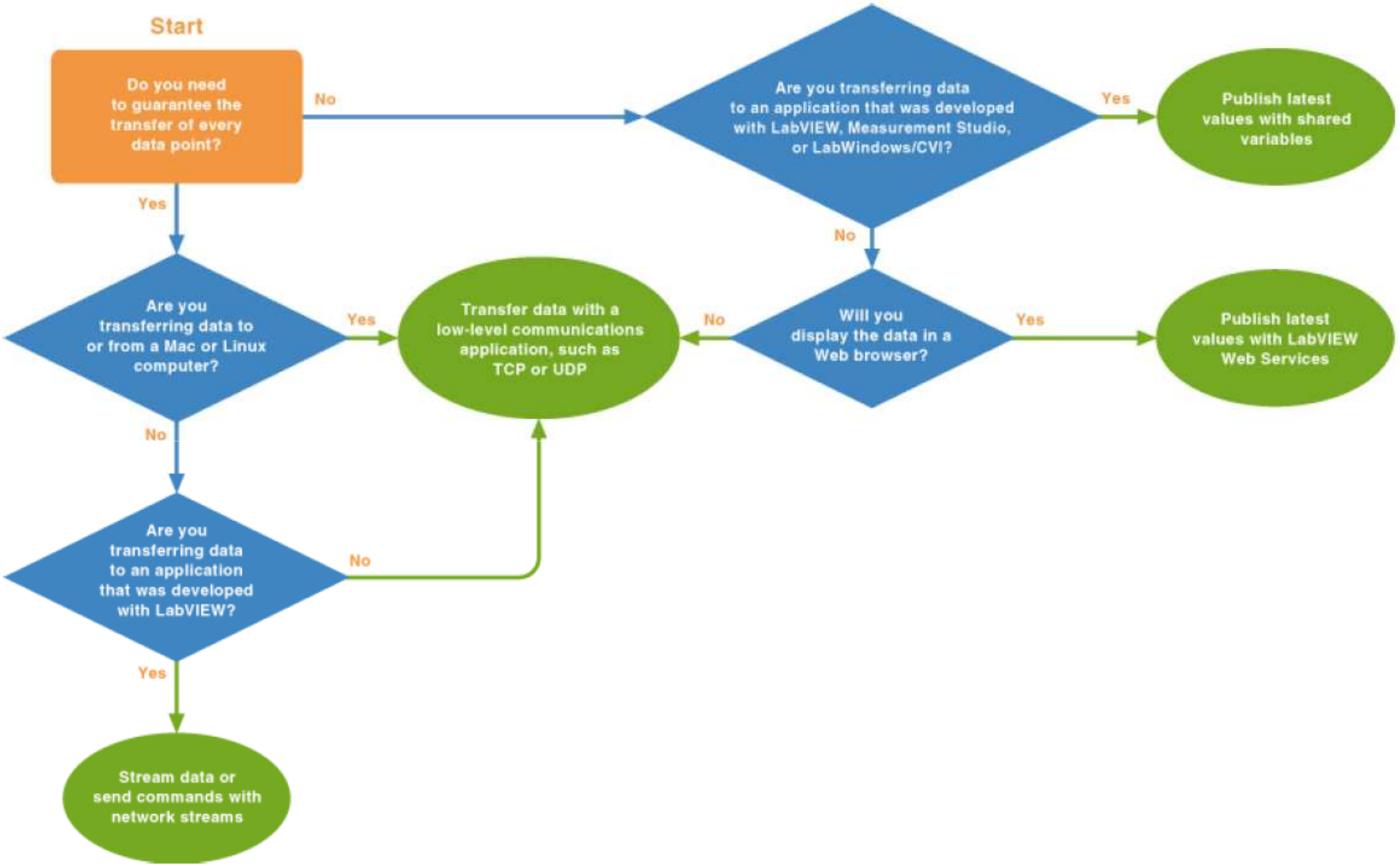
Direct memory access between to different components of a system







NATIONAL INSTRUMENTS



NATIONAL INSTRUMENTS

Section Five

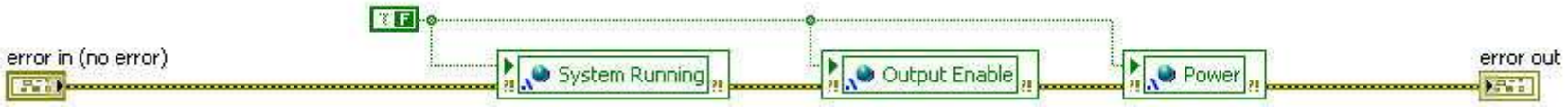
Advanced Error Handling Tools

NATIONAL INSTRUMENTS

Specific Error Handling Guidelines

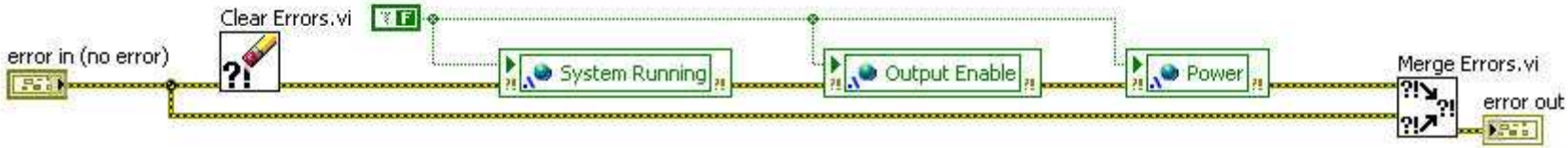
- Avoid allowing errors to affect subsequent code
  - Not all code behaves the same in the event of an error
  - Better to specifically decide which code to execute using case structures
- Example:
  - As part of shutting down your application, you need to turn off three values by writing false to three variables.

Specific Error Handling Example



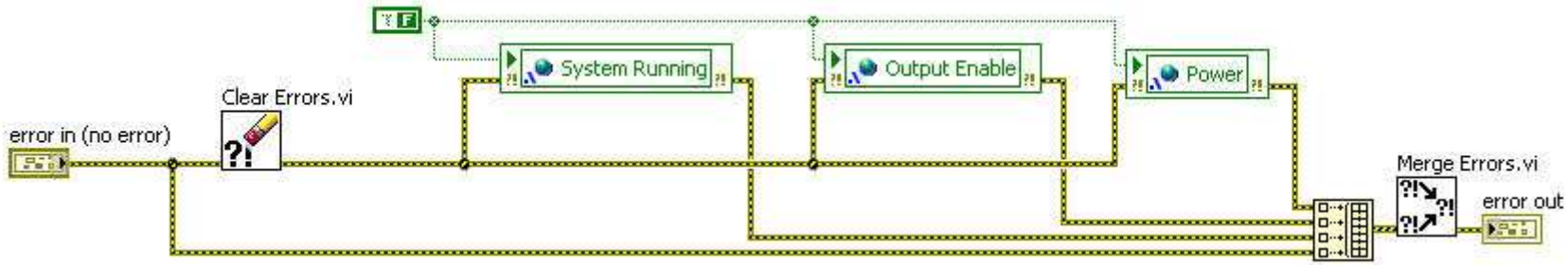
NATIONAL INSTRUMENTS

Specific Error Handling Example



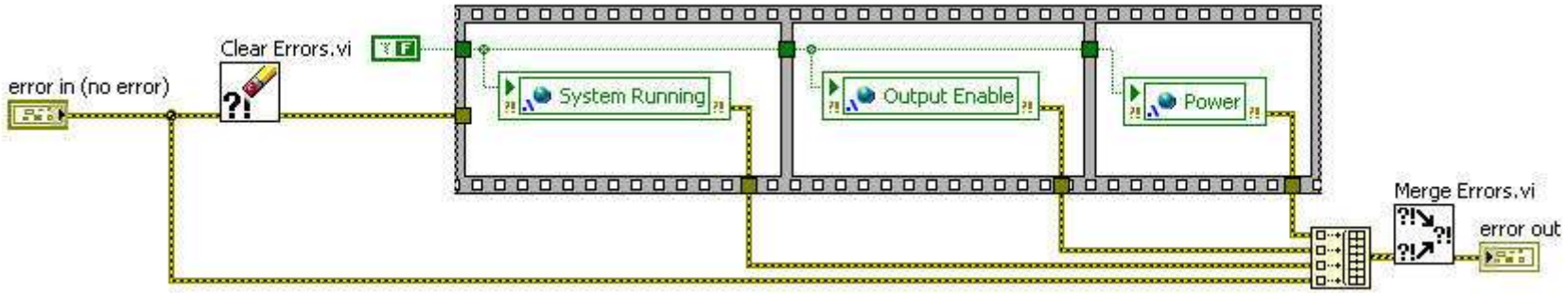
NATIONAL INSTRUMENTS

Specific Error Handling Example



NATIONAL INSTRUMENTS

Specific Error Handling Example



NATIONAL INSTRUMENTS

Error Classification and Description

- Organizes errors into classifications
- Examples
  - Warning
  - Critical Error
  - User Error
  - Communications Error

- Records additional information about error
- Examples
  - Timestamp
  - Number of Occurrences
  - Variable values

Error Communication

- Transfers an errors from their origin to a central location
- Desirable features:
  - Priority
  - Filtering (count occurrences)
  - Deterministic transfer for RT applications

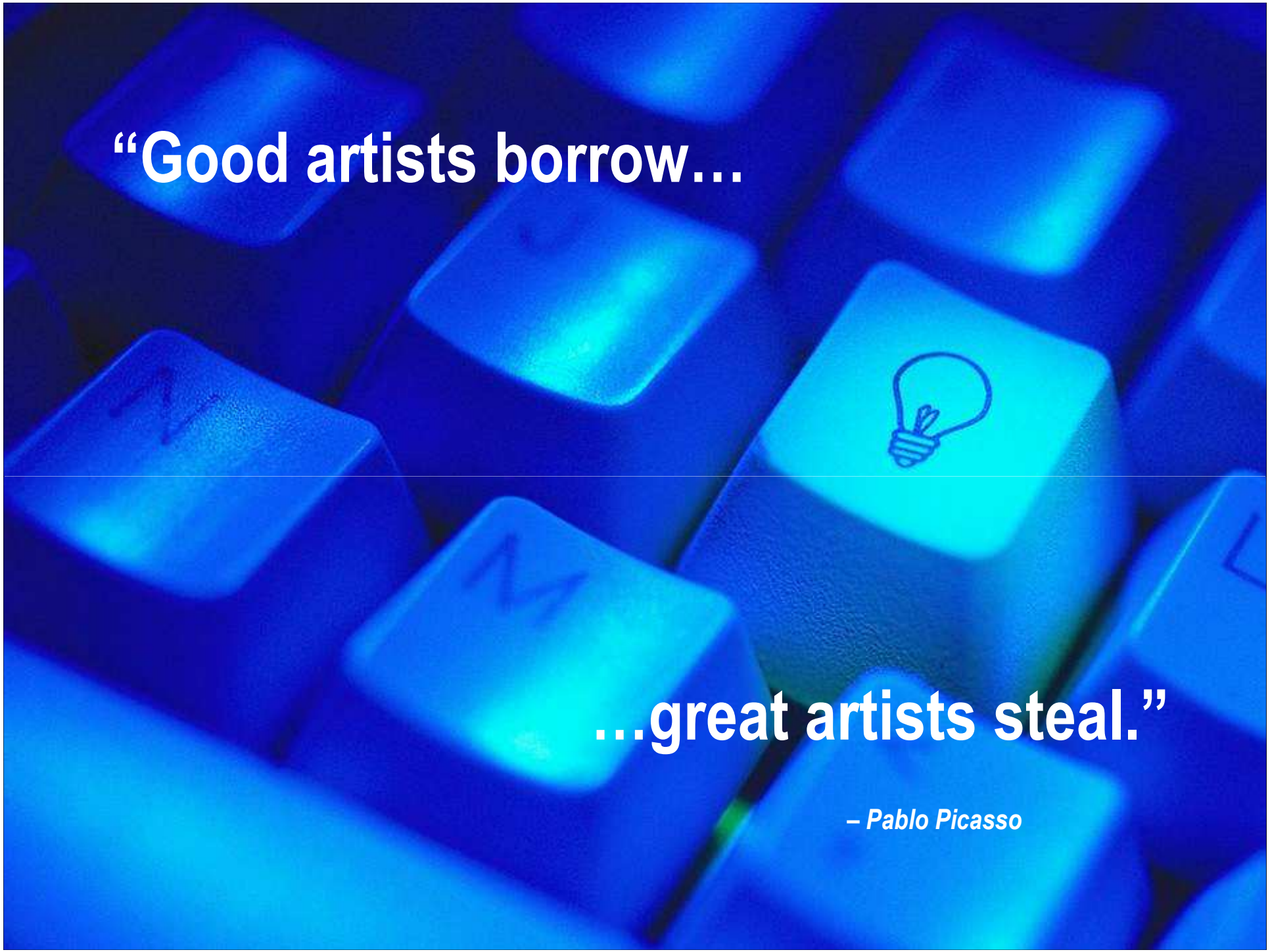
Error Communication Available Tools

- Custom Solutions
  - Typically use queues, AMC, or priority queues
- Structured Error Handler
  - Notification system provides priorities and filtering
- Scan Engine Faults
  - Provides priorities and filtering

Section Six

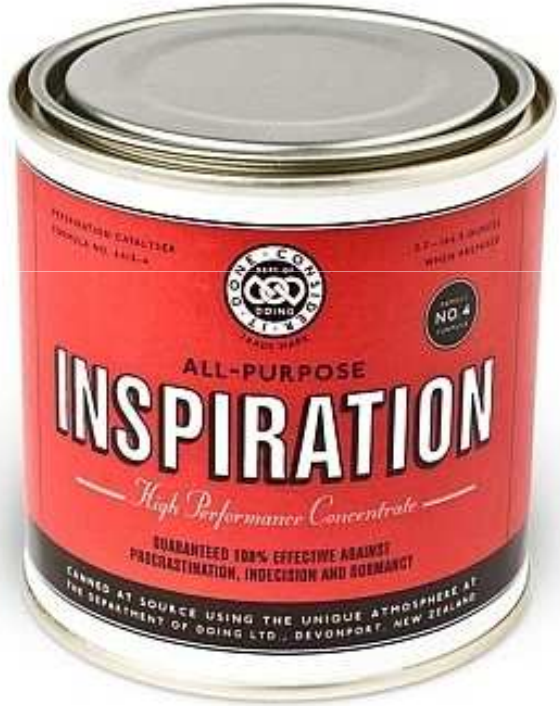
Advanced User Interface Design

NATIONAL INSTRUMENTS


“Good artists borrow…
…great artists steal.”
– Pablo Picasso

Sources of Inspiration

- Apple
- Microsoft Office
- Applications on your own computer
- Icon galleries
- Web design tutorials
- Your corporate design team
- Photography, Art
- Books on:
  - User interface design
  - User interaction design
  - Usability
  - Graphic design
  - Visualization of information



NATIONAL INSTRUMENTS

The Commandments of UI Design

- I. Help your user achieve their goal
- II. Don’t Be (too) Innovative
- III. Less Is More

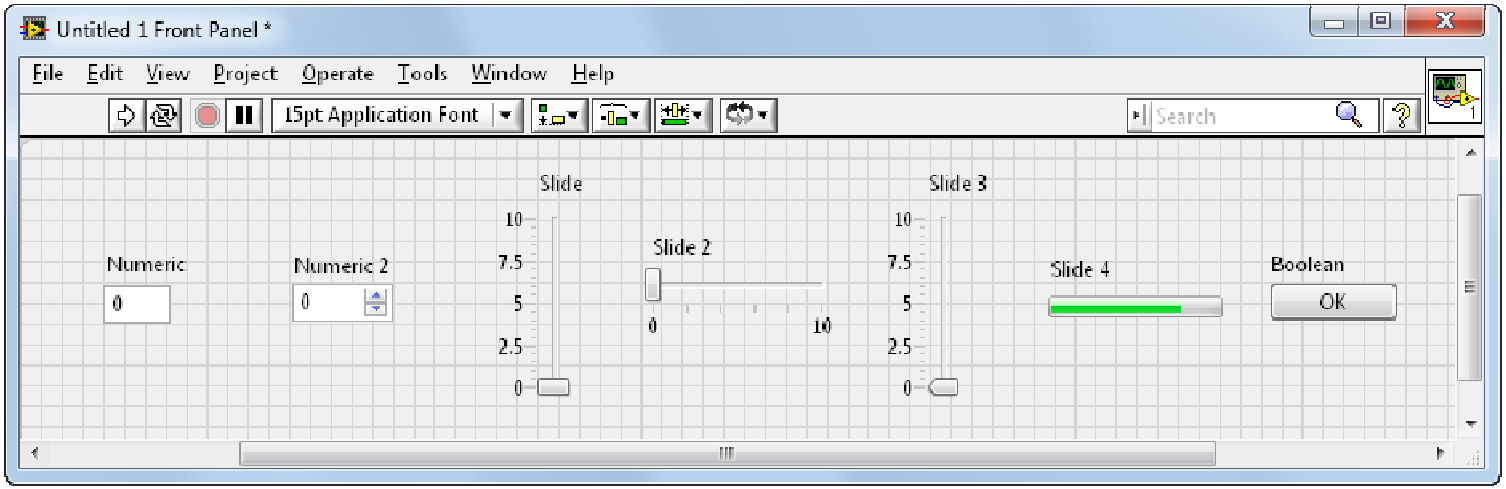
NATIONAL INSTRUMENTS

I. Think About Your User

- Do they know as much as you (they never do)?
- How will they interact with the application?
- Why are they using your software?
  - The software (and UI) should support their goal

II. Don’t Be Innovative

- Use familiar elements
- Don’t change expected functionality
- Polish – don’t reinvent



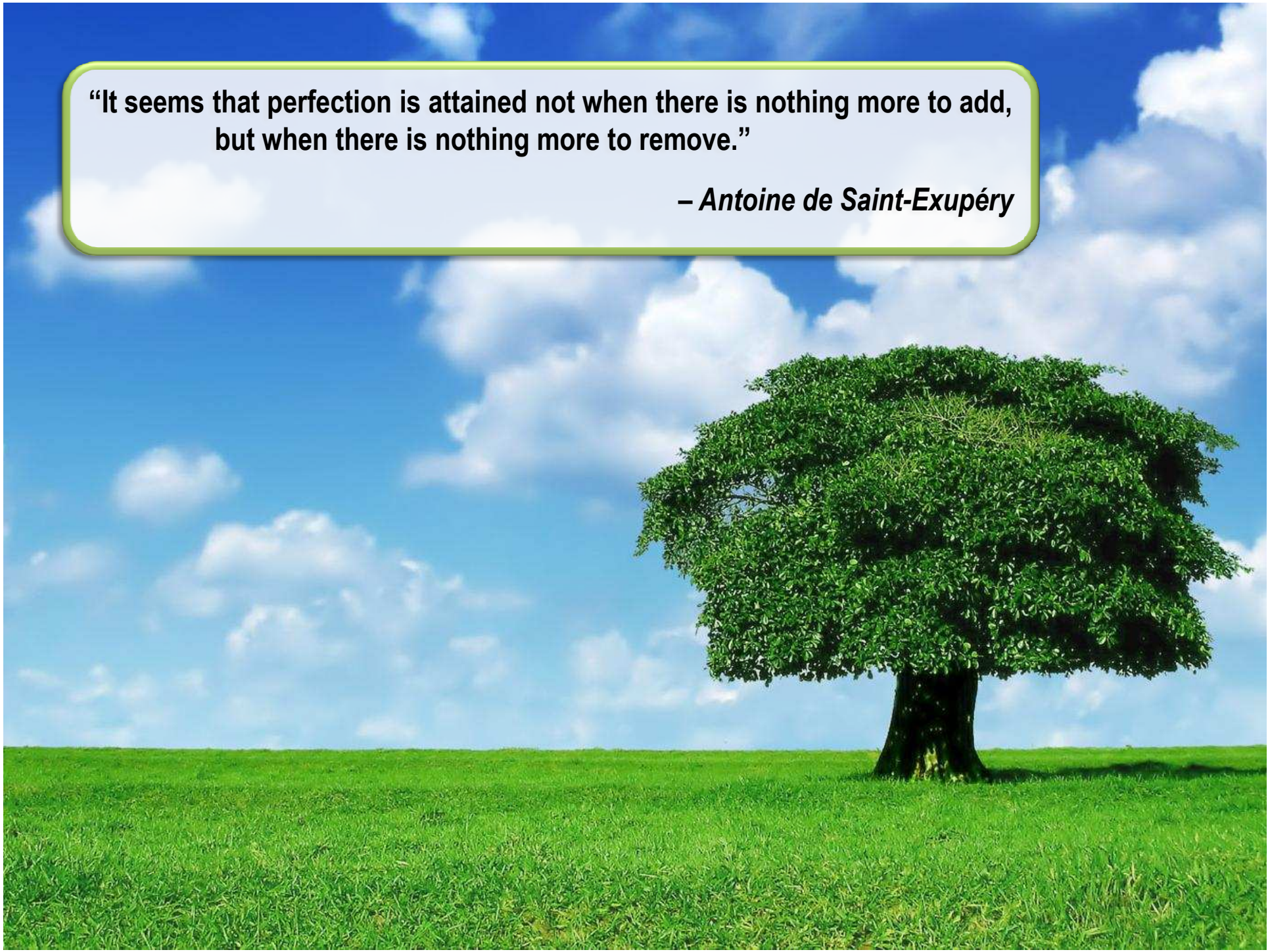
System controls mimic common operating system controls and are familiar to most users

NATIONAL INSTRUMENTS

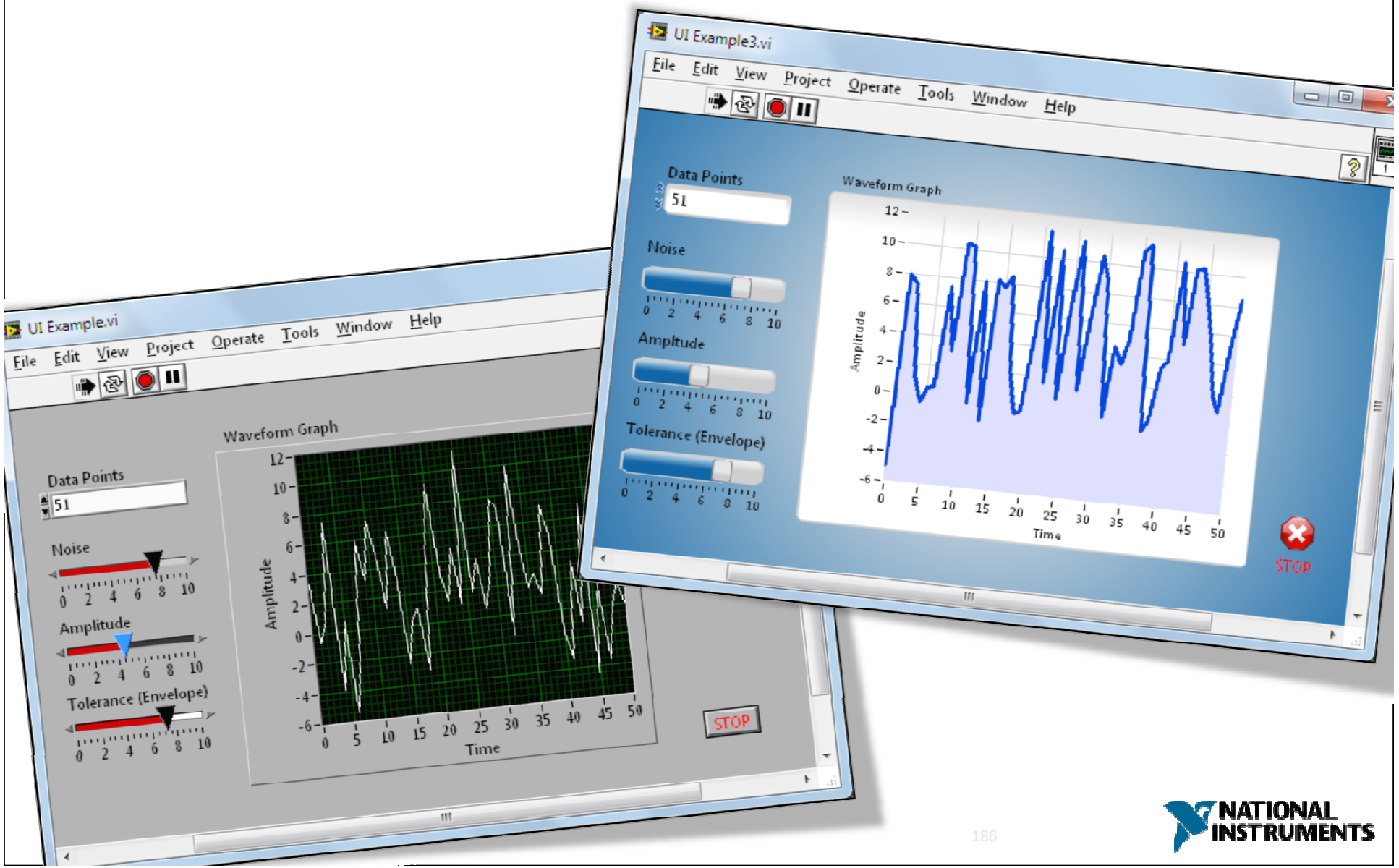
III. Less Is More

- Too much at once is distracting; err toward minimalism
- Stick to requirements
  - Don't do what isn't necessary
  - You'll get done sooner
  - It'll cost you less to own it
- Focus should be on what's important
  - Example: Guide the user with color…

NATIONAL INSTRUMENTS


“It seems that perfection is attained not when there is nothing more to add,
but when there is nothing more to remove.”
– Antoine de Saint-Exupéry

Giving your UI a Custom Look



NATIONAL INSTRUMENTS

UI Customization Techniques

Front Panel Images and Decorations

- Make default controls transparent
- Add an image via menu **Edit » Paste**

Custom Controls

- Use to customize cosmetics of controls and indicators
- Access via front panel right-click context menu **Advanced » Customize**

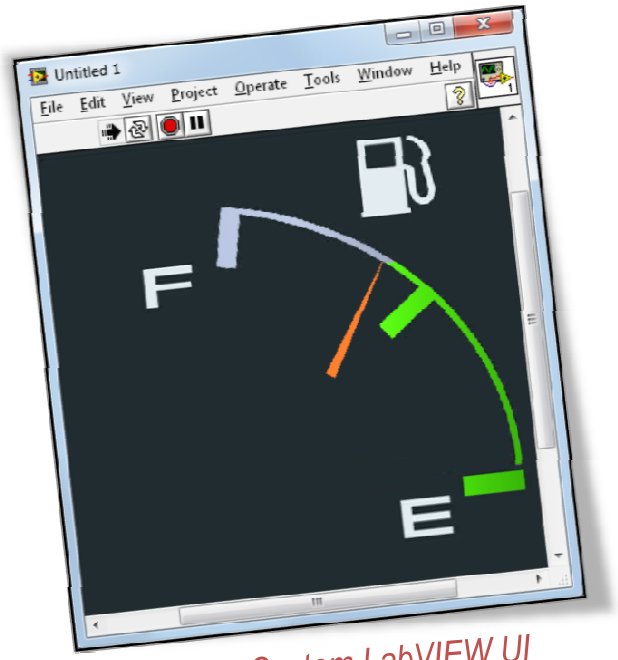
XControls

- Use to customize the functionality or cosmetics of controls and indicators
- Access via menu **File » New**

Adding an Image to the Front Panel

- As simple as **Copy and Paste**!
- Use an image editor to erase, add transparency
- Populate native LabVIEW controls

Start: PowerPoint Clipart

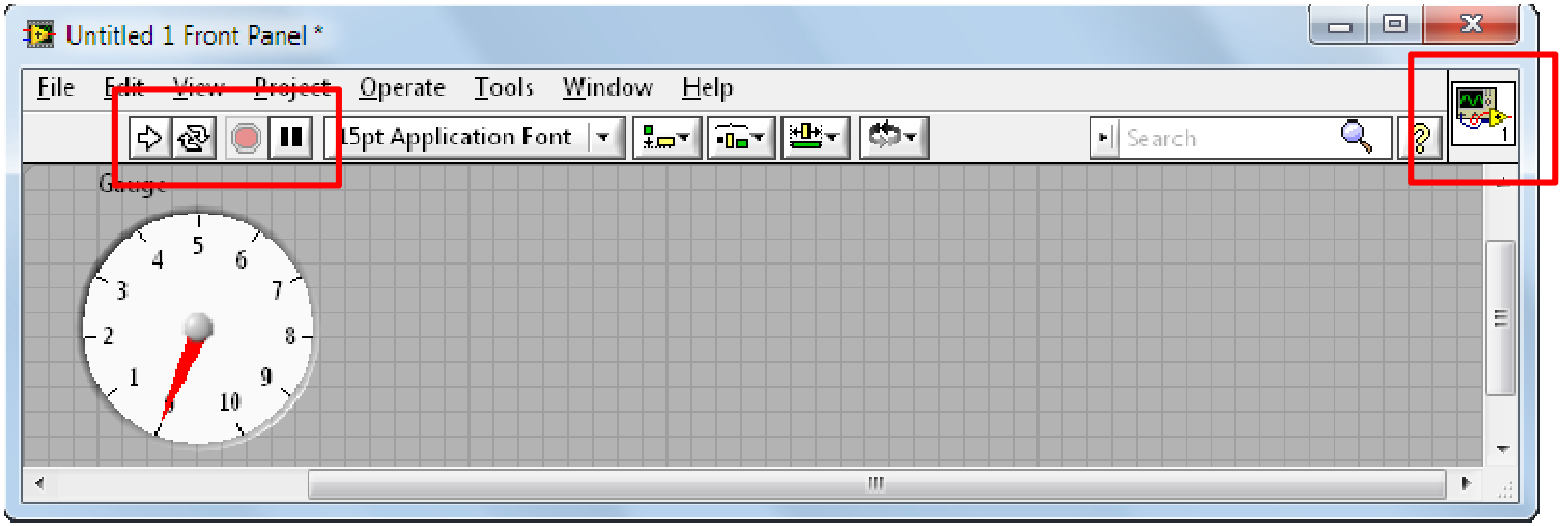


Finish: Custom LabVIEW UI

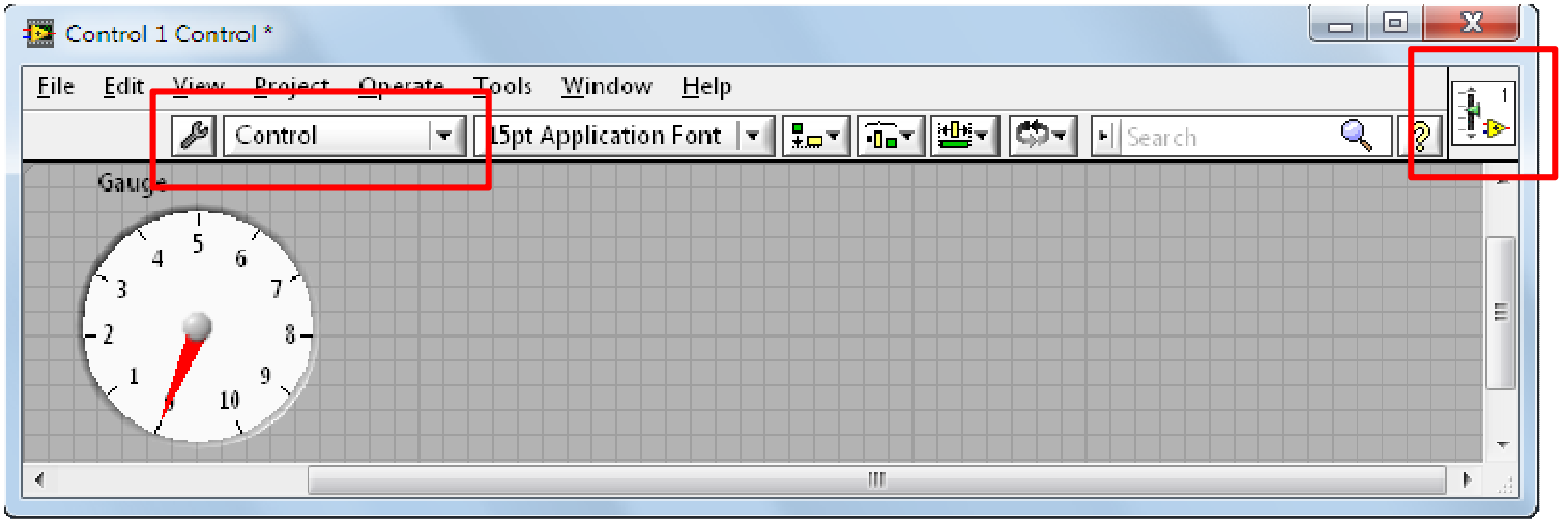
NATIONAL INSTRUMENTS

Differences in the Control Editor

Normal Front Panel



Control Editor Window



NATIONAL INSTRUMENTS

When you should use Controls

- To create cosmetically different, reusable controls
- **Example:** resizing a Stop button to make it easier to click

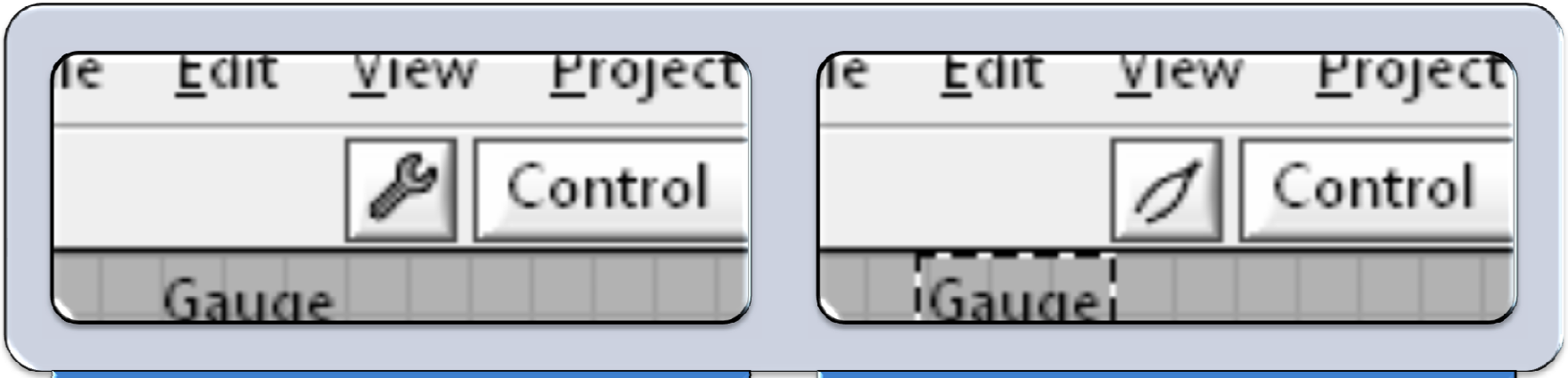
When you should use Type Defs

- To automatically update data type in custom controls
- **Example:** reusing an Enum whose values may change

When you should use Strict Type Defs

- To create reusable controls that are identical copies
- **Example:** propagating changes to Gauge size and color

Edit Mode vs. Customize Mode



Edit Mode

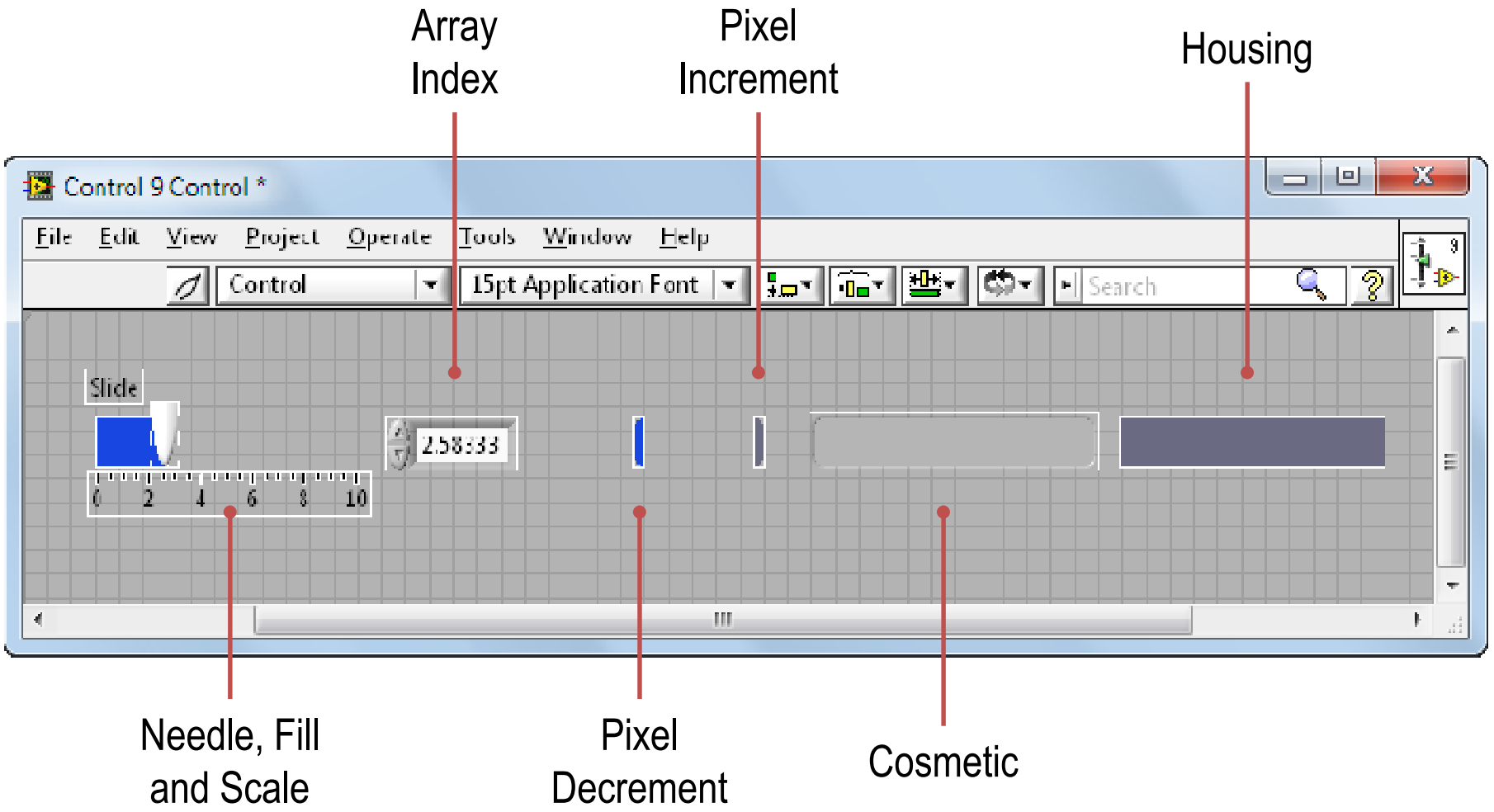
- Change size or color of a control or indicator
- Access a right-click shortcut menu

Customize Mode

- Make extensive changes to controls or indicators
- Change individual parts of a control or indicator

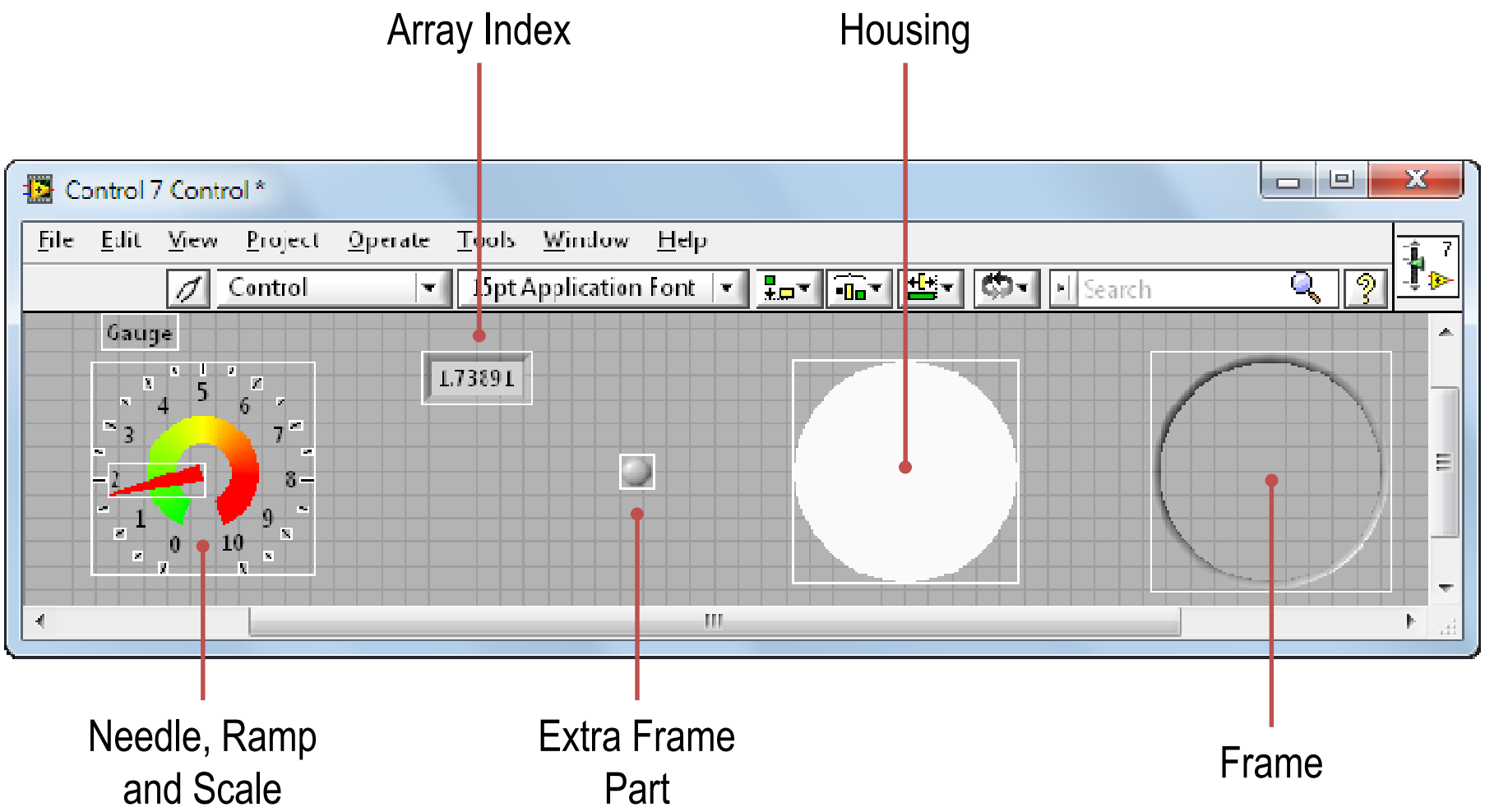
NATIONAL INSTRUMENTS

Components of a Control (Slide)



NATIONAL INSTRUMENTS

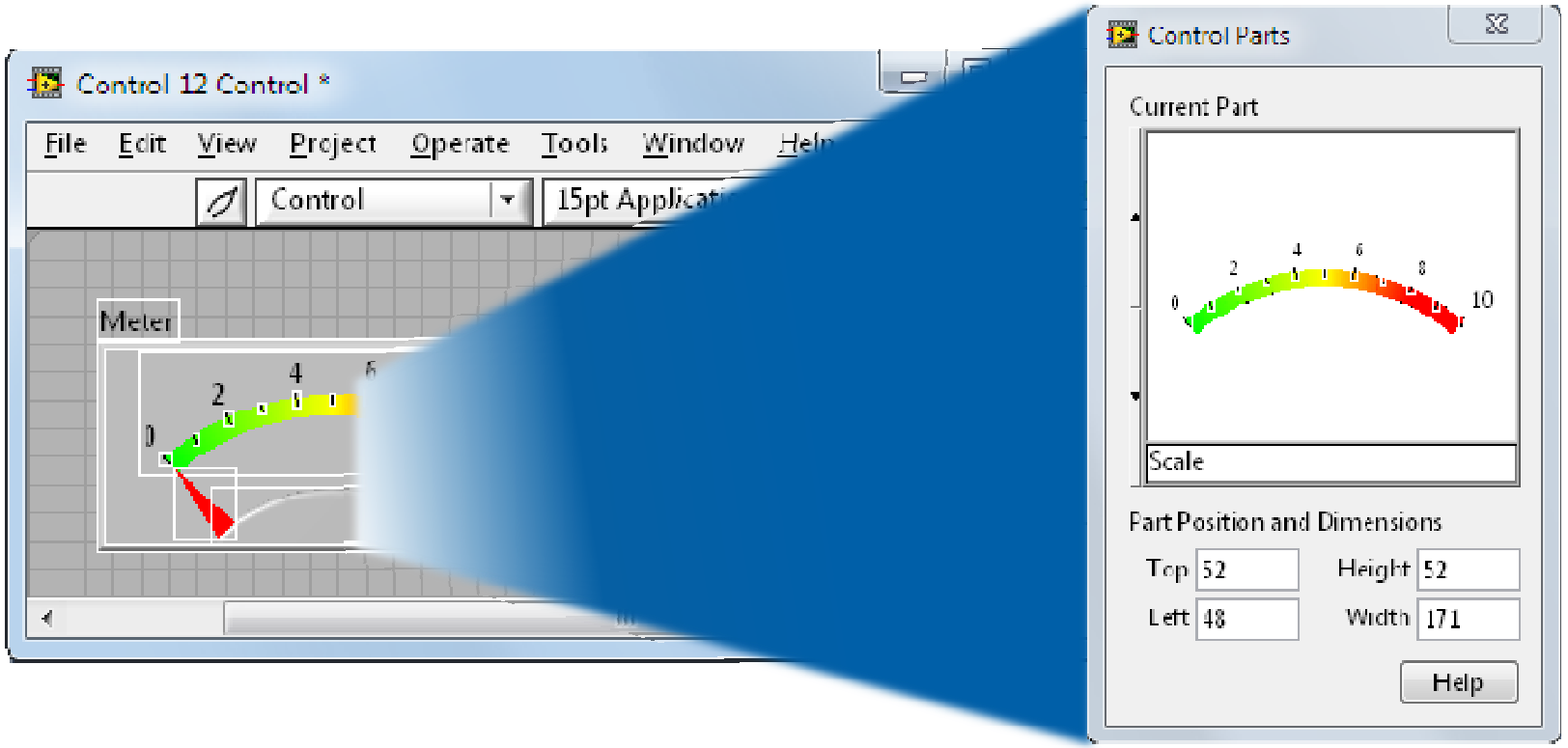
Components of a Control (Gauge)



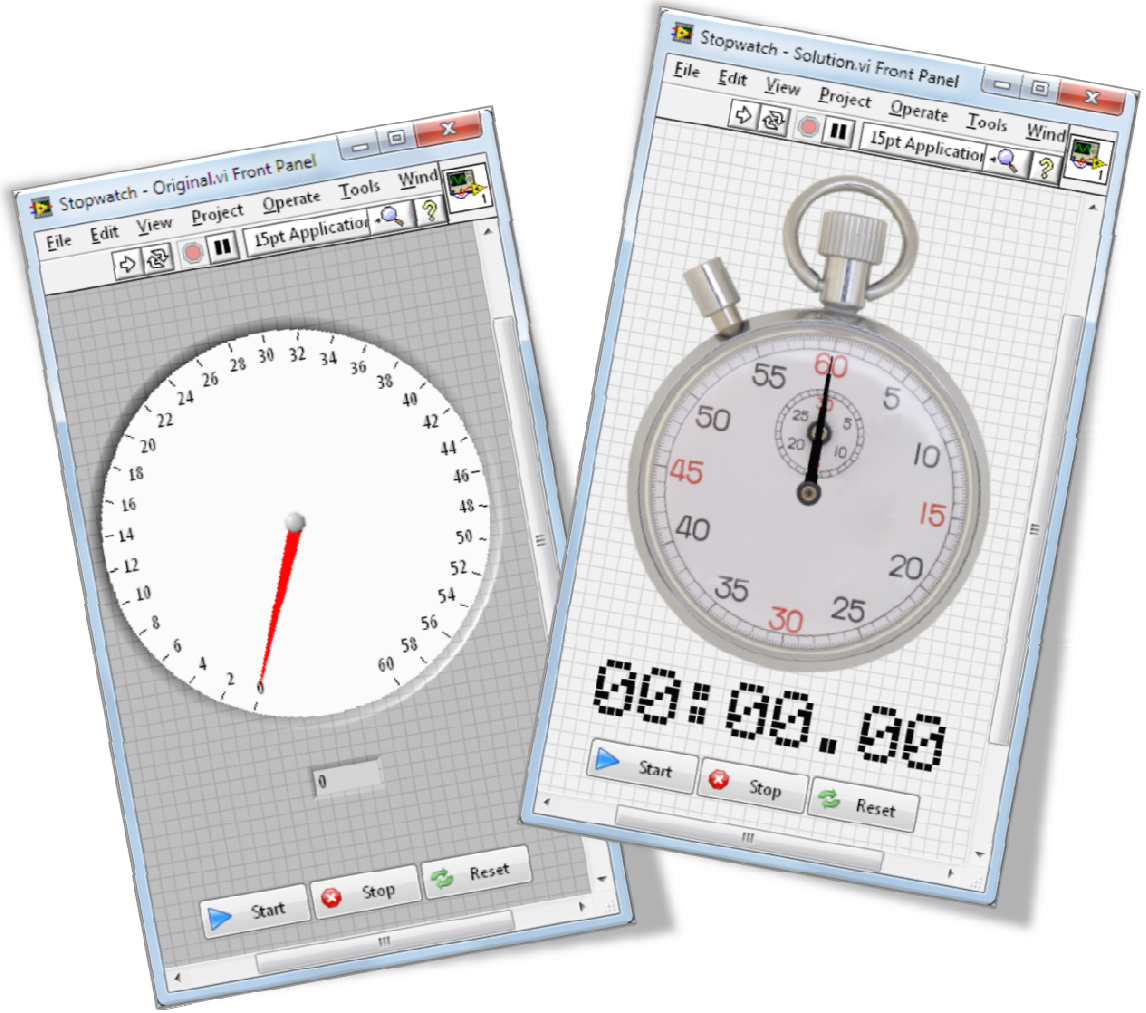
NATIONAL INSTRUMENTS

Viewing Control Components

- View individual control components using menu **Window » Show Parts Window**



NATIONAL INSTRUMENTS



Adding a Decoration and Custom Imagery to a Gauge Control

DEMO

NATIONAL INSTRUMENTS

When you should use XControls

- To create reusable components with dynamic behavior
- To encapsulate extended functionality or cosmetics

When you should not use XControls

- To accomplish pure cosmetic changes
- When working on a single-shot application

Use Strict Type Defs instead when…

- You do not need dynamic run-time and edit-time behavior

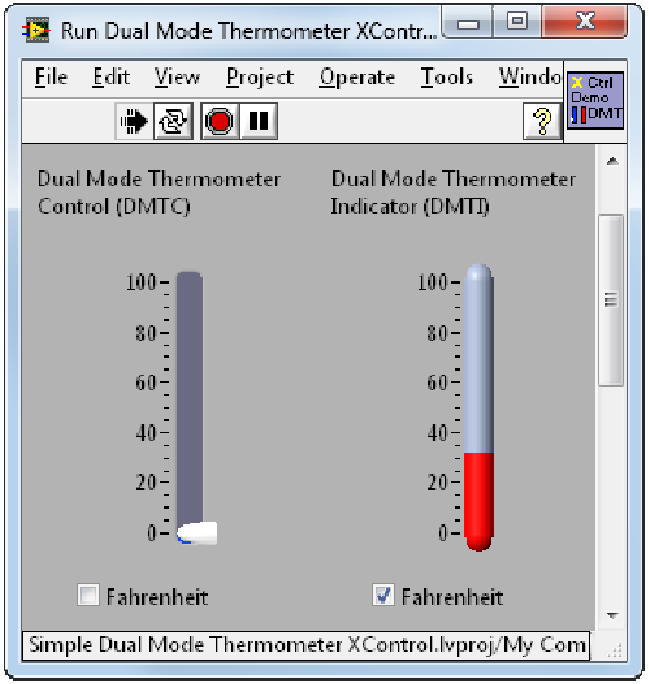
NATIONAL INSTRUMENTS

An XControl Example…

Design a thermometer control that can represent a single numeric input in either Celsius or Fahrenheit.

Discussion:

What makes this a good application for an XControl?



Note: This example XControl is pre-built as part of the default LabVIEW Examples

NATIONAL INSTRUMENTS

Example XControl Requirements

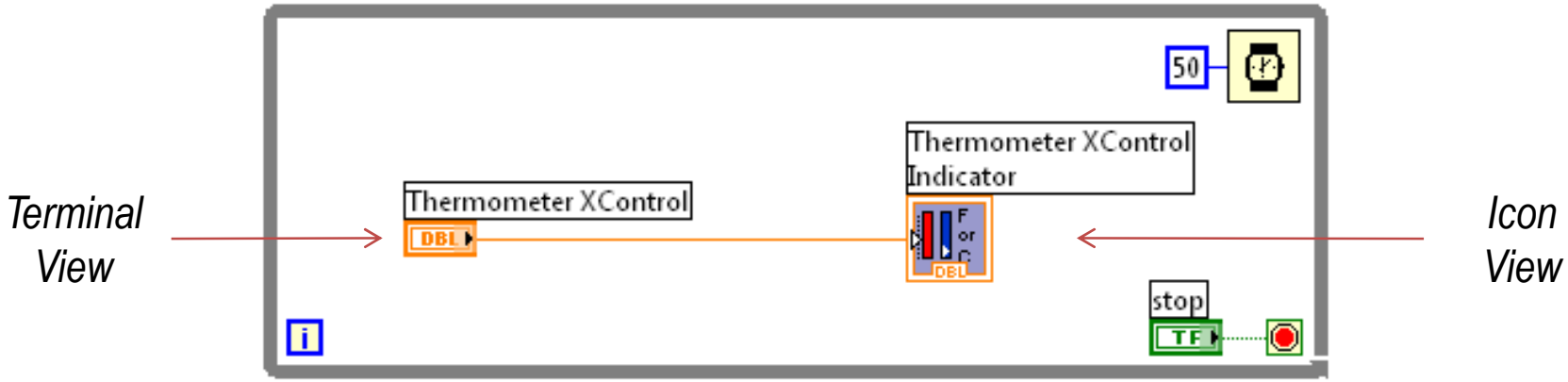
1. Thermometer must keep track of which temperature unit it represents

2. Thermometer must convert temperature between Fahrenheit and Celsius

3. Thermometer must represent a single numeric value and look (visually) like a thermometer.

Using XControls

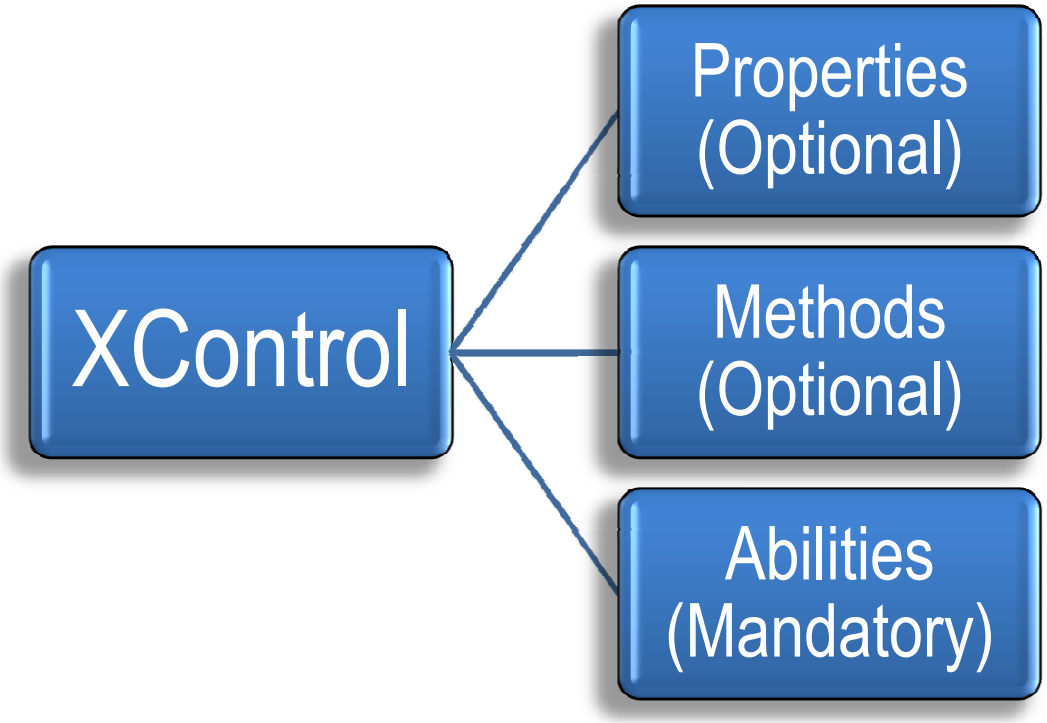
- Manage XControls via the Project Explorer
- XControls appear as regular terminals
- You start with shell code

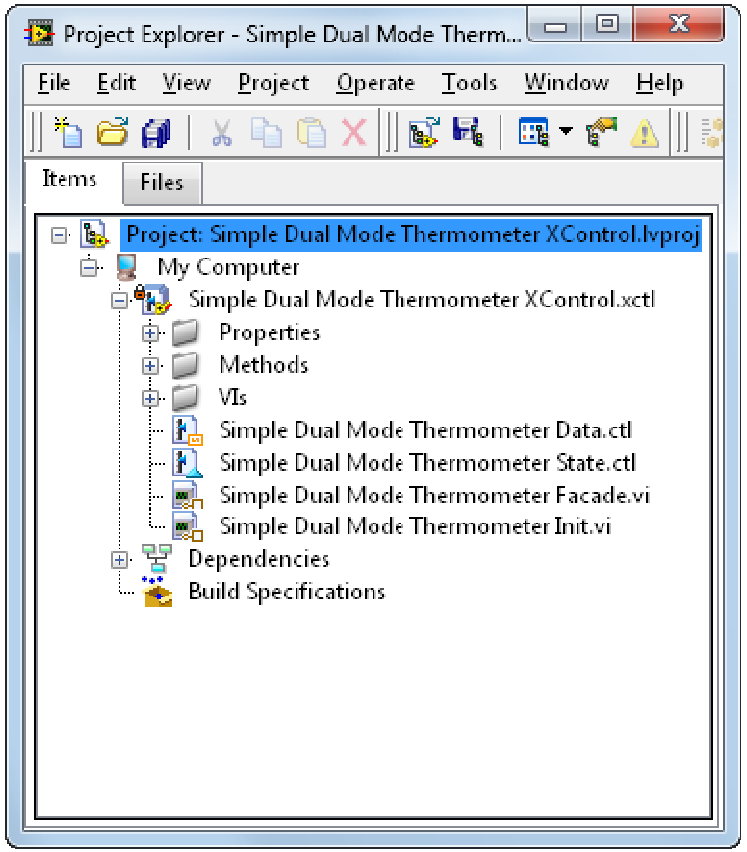


Remember – you can go a long way without resorting to XControls

NATIONAL INSTRUMENTS

Structure of an XControl

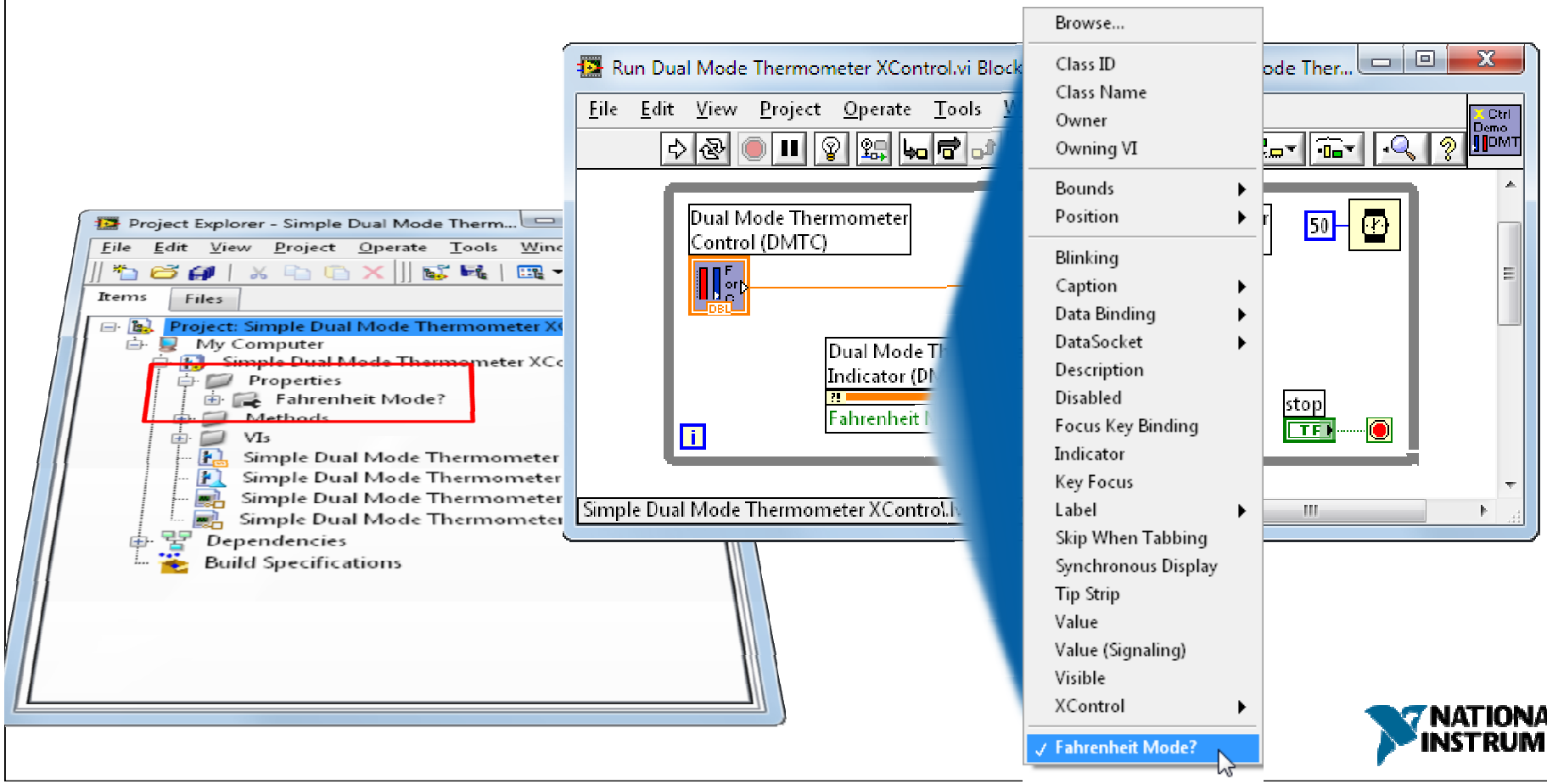




NATIONAL INSTRUMENTS

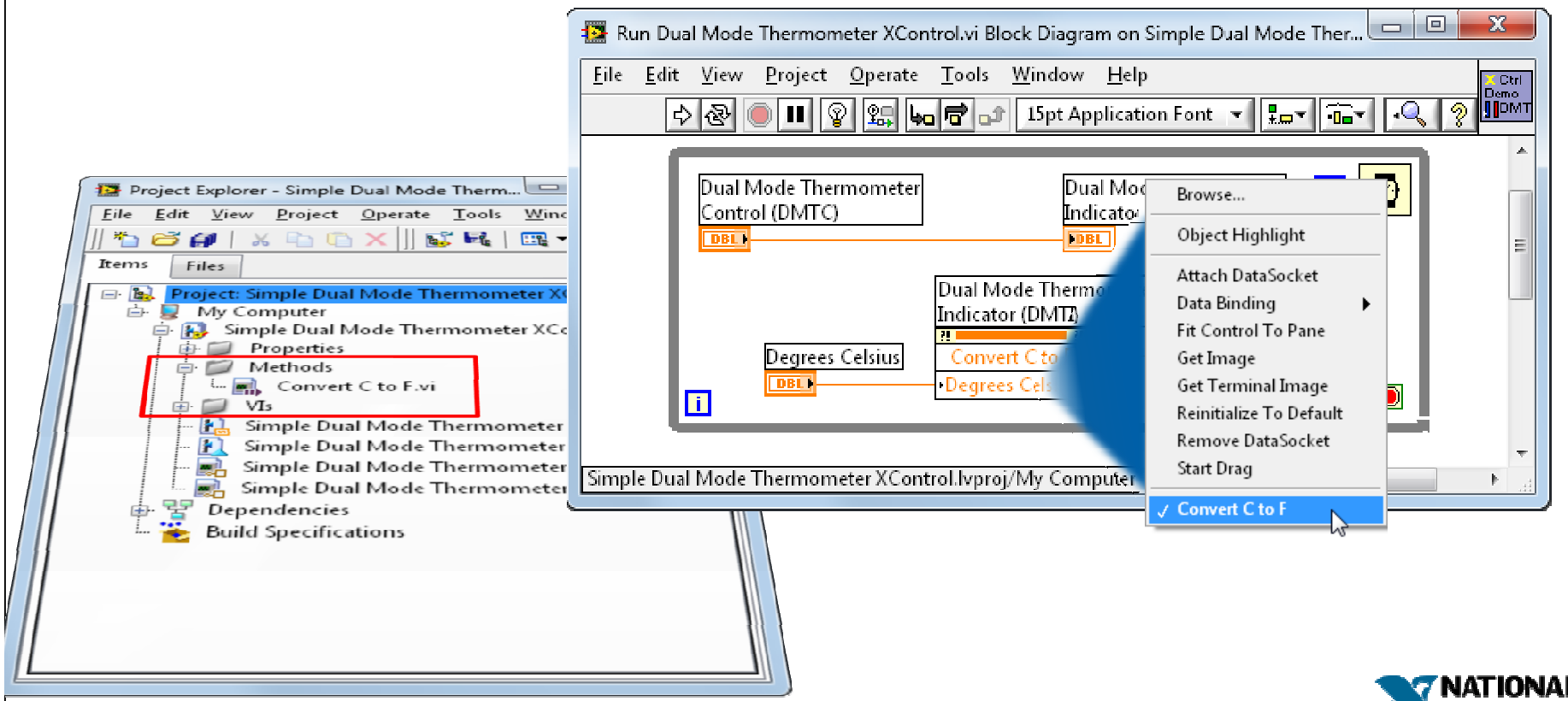
XControl Properties

- Allow the user to configure the XControl programmatically via Property Node



XControl Methods

- Allow the user to engage functionality of the XControl programmatically via Invoke Node

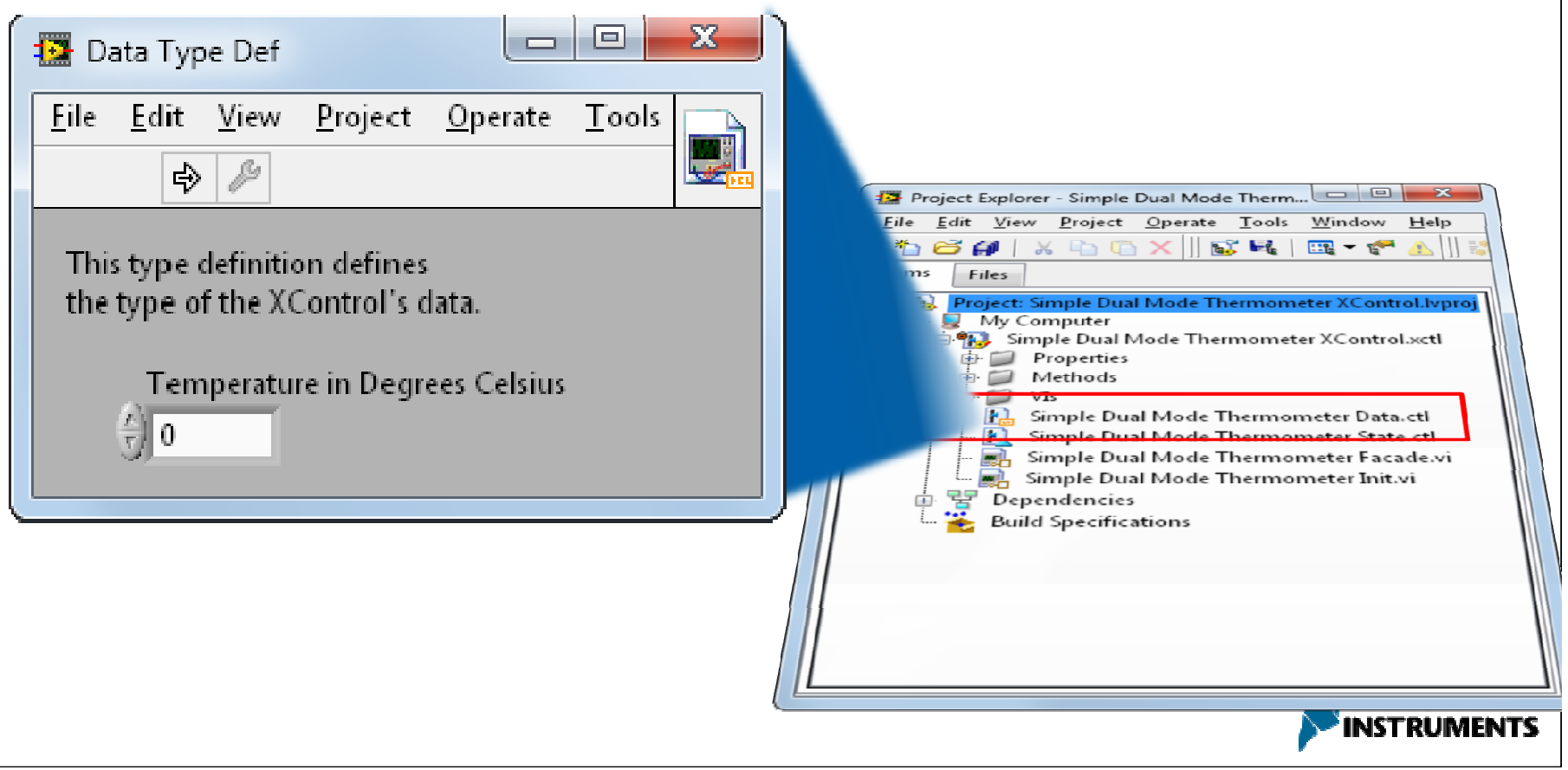


XControl Abilities

- Required components for proper function
- Represented by VIs or Controls
- Four mandatory abilities:
  - Data
  - State
  - Façade
  - Init
- Additional optional abilities may exist

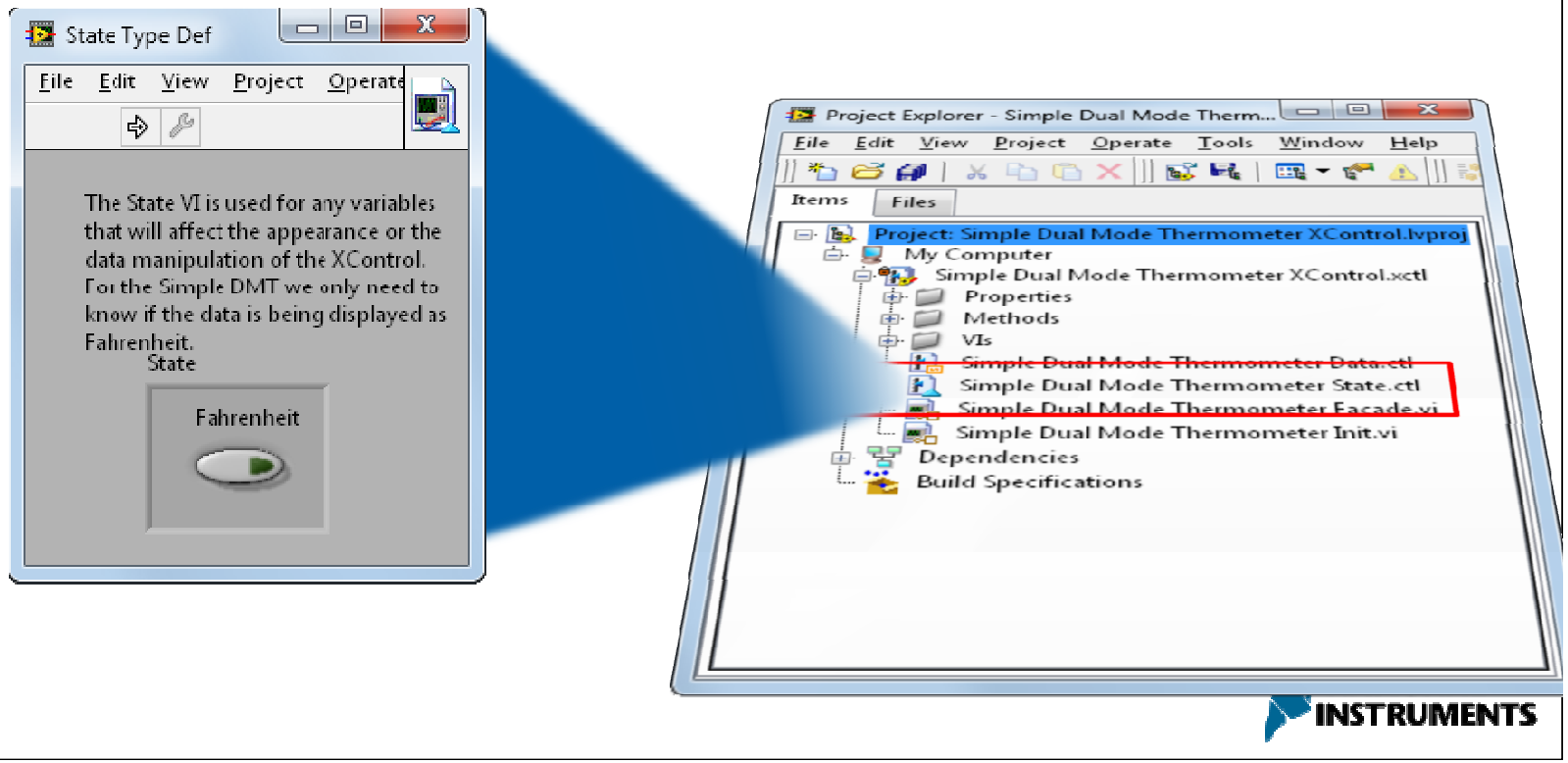
XControl Data Ability

- Type Definition Control
- Specifies the data type of the XControl



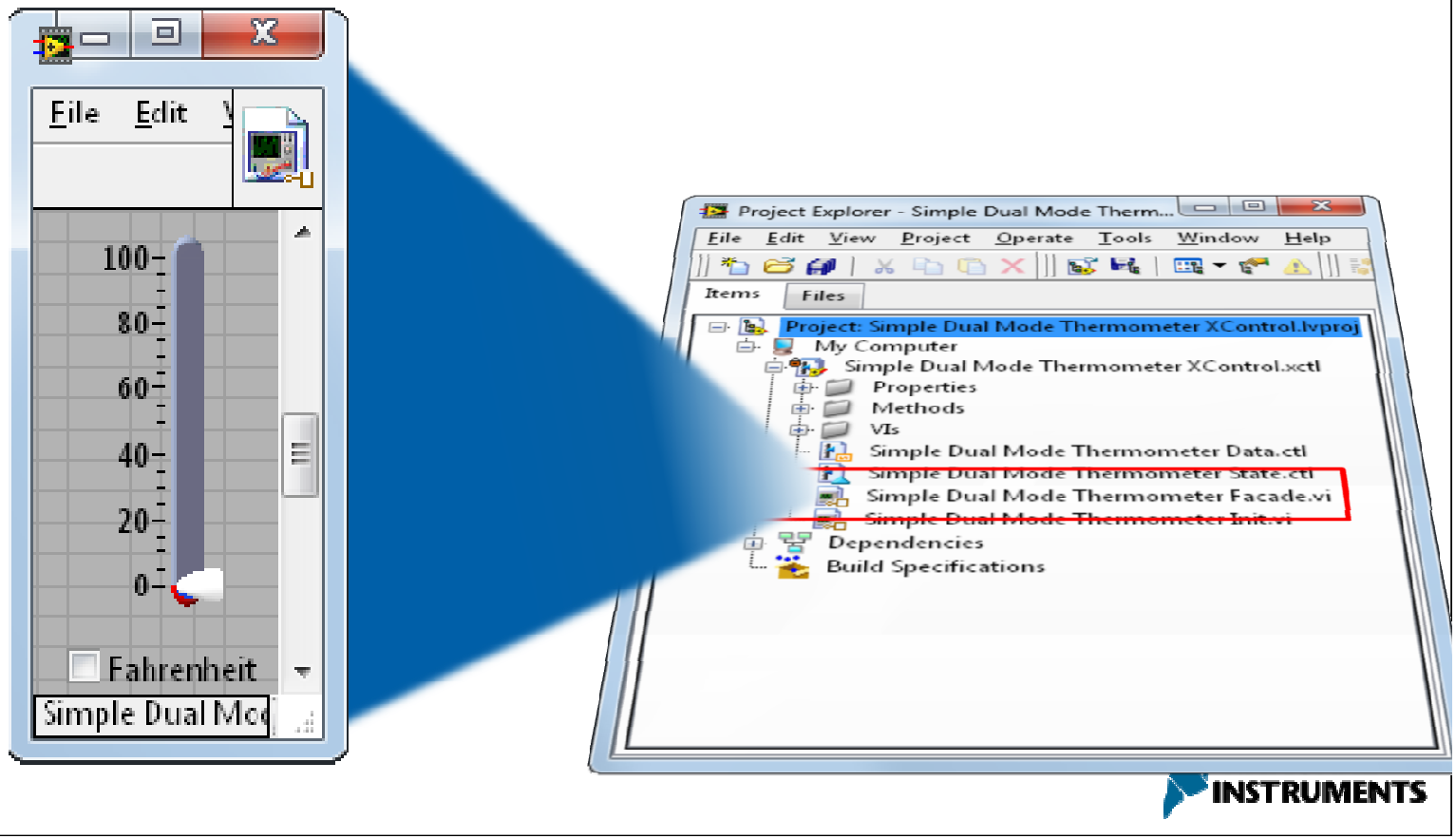
XControl State Ability

- Type Definition Control
- Specifies information other than data type that affect the appearance of the XControl



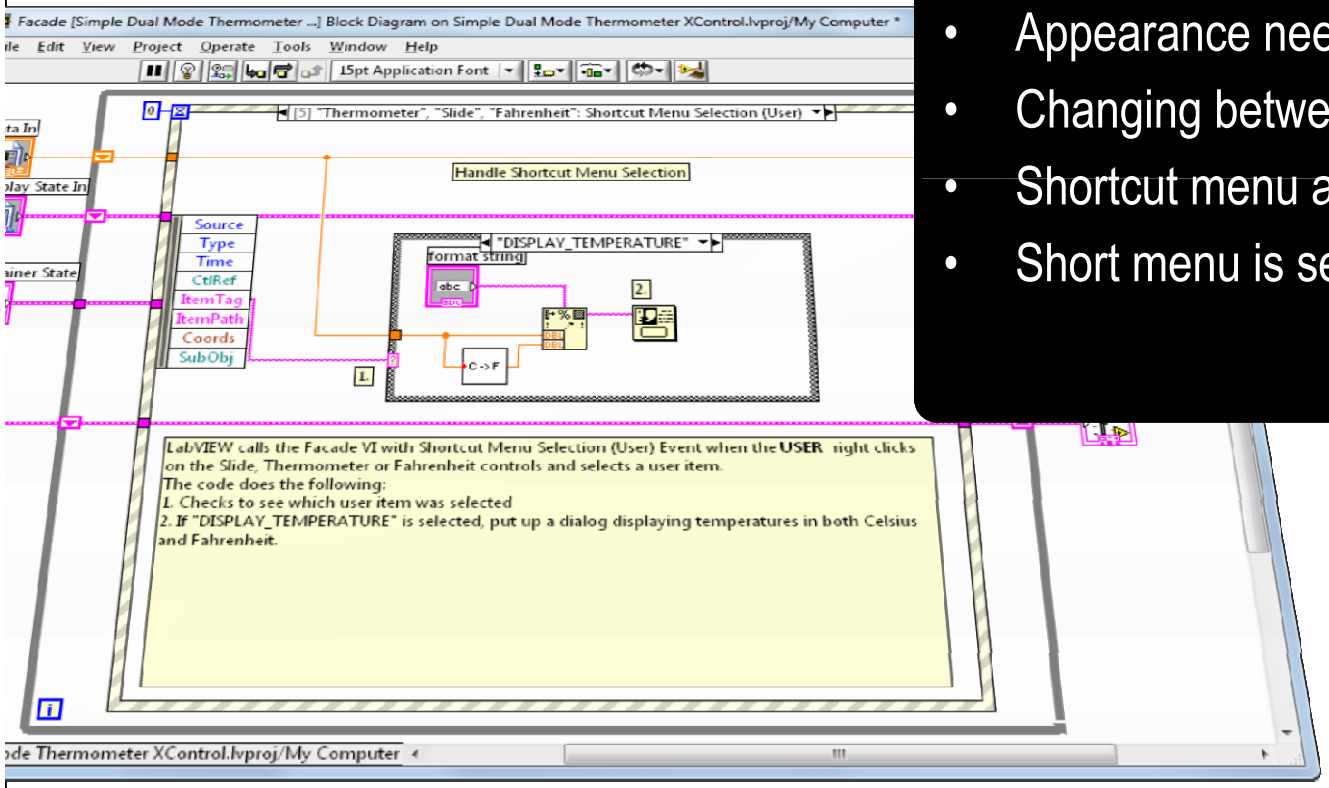
XControl Façade Ability

- Defines the appearance of the XControl
- Invoked after changes to Properties / Methods



XControl Façade Ability

- Also invoked when:



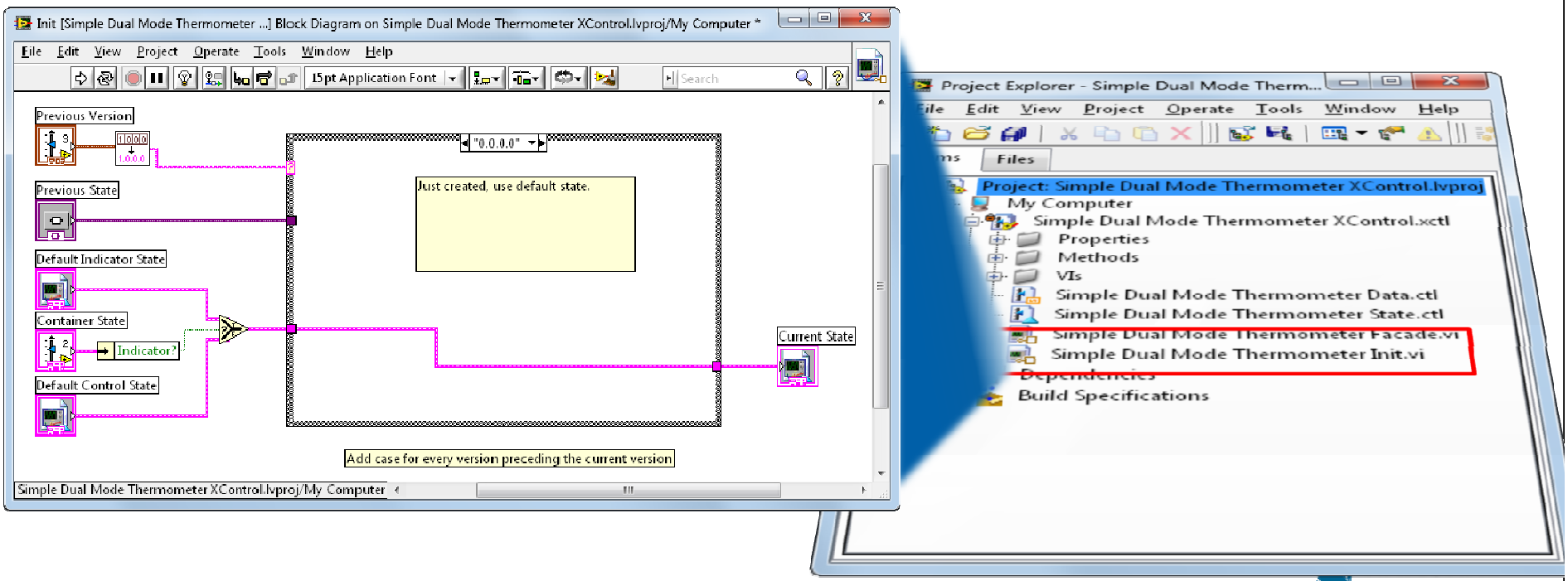
- Data is written to the control
- Appearance needs updating
- Changing between Control and Indicator
- Shortcut menu activates
- Short menu is selected

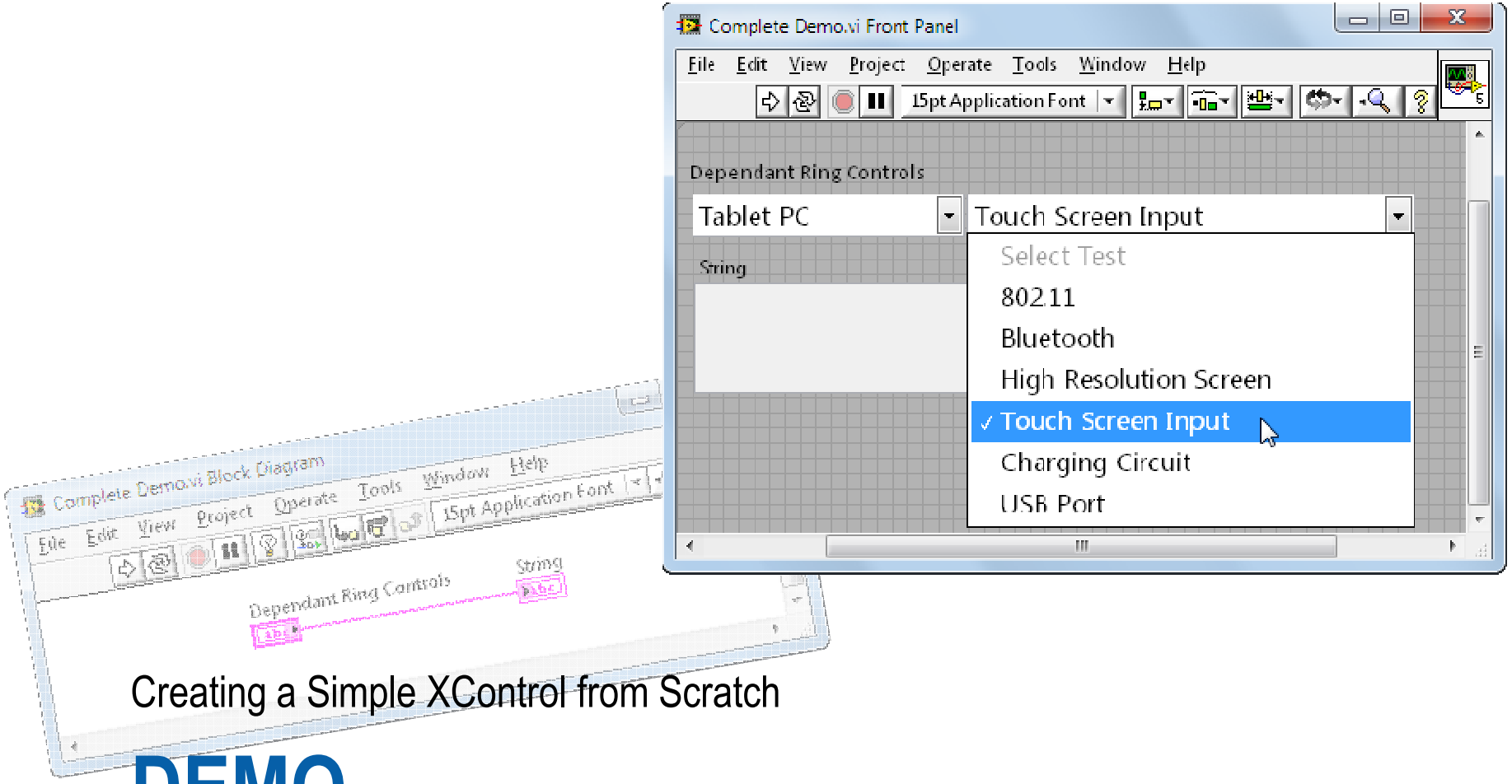
…and more

NATIONAL INSTRUMENTS

XControl Init Ability

- Called upon first placement or load into memory
- Initializes display state before being displayed
- Handles control versioning



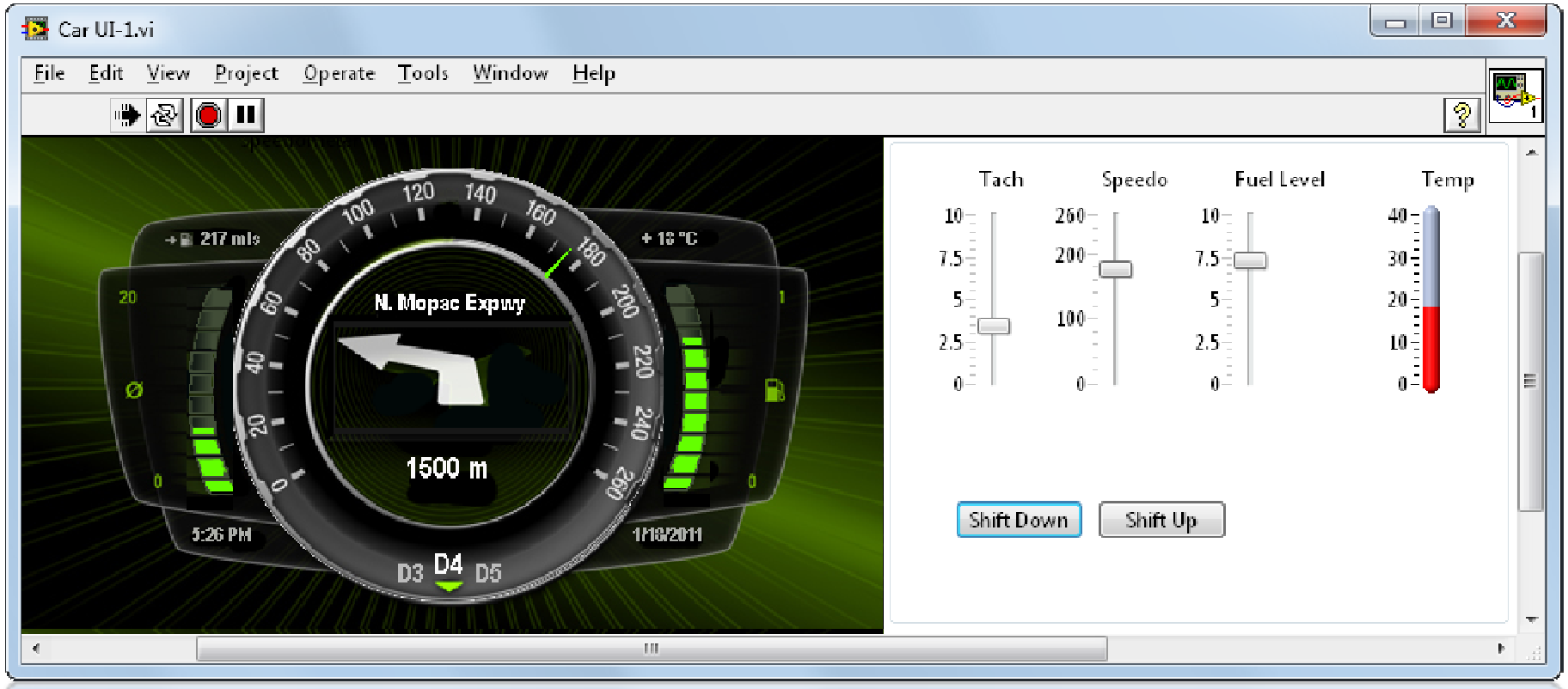


Creating a Simple XControl from Scratch

DEMO

NATIONAL INSTRUMENTS

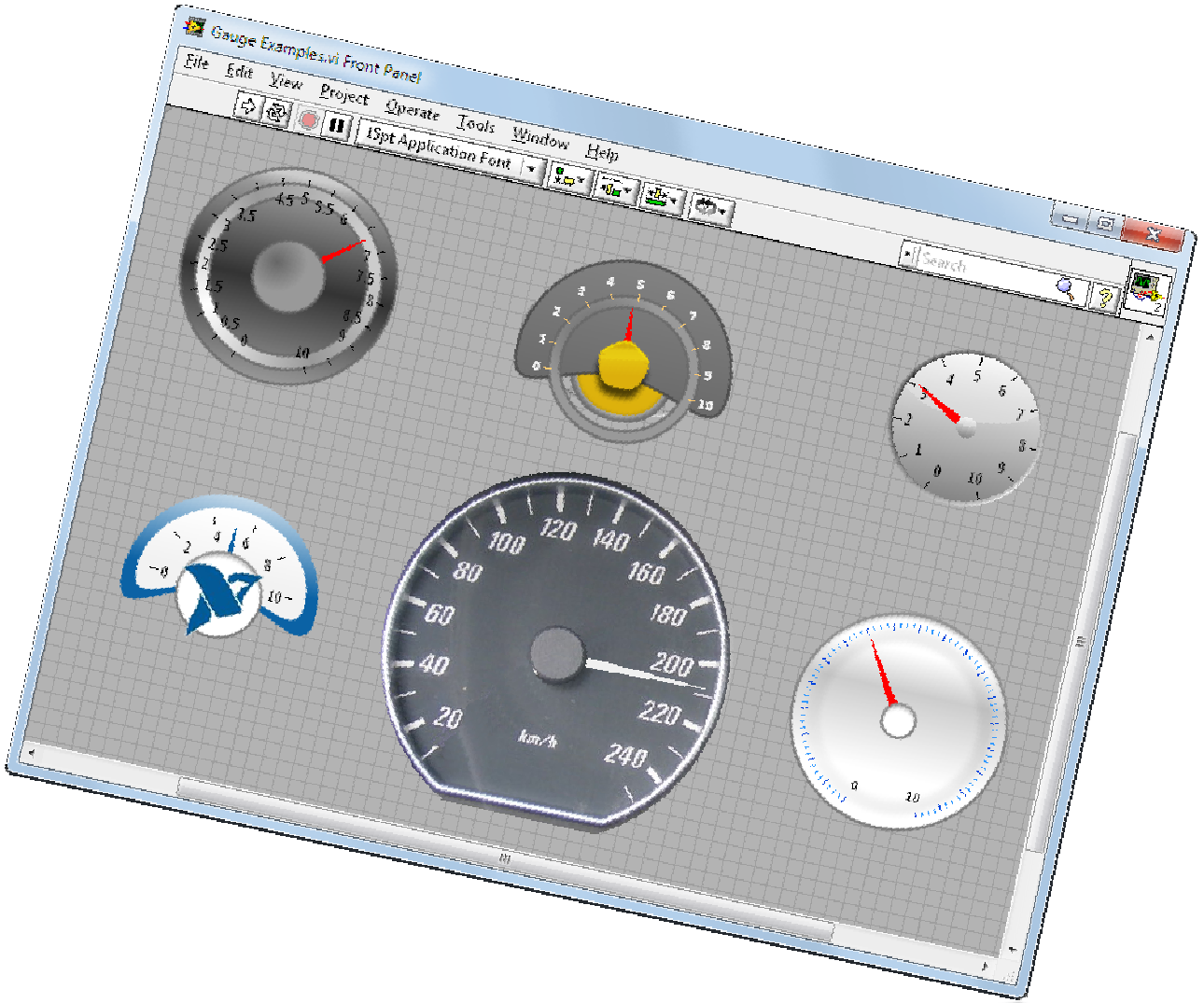
Gallery of Examples



Credit: Simon Hogg
NI Community UI Interest Group

NATIONAL INSTRUMENTS

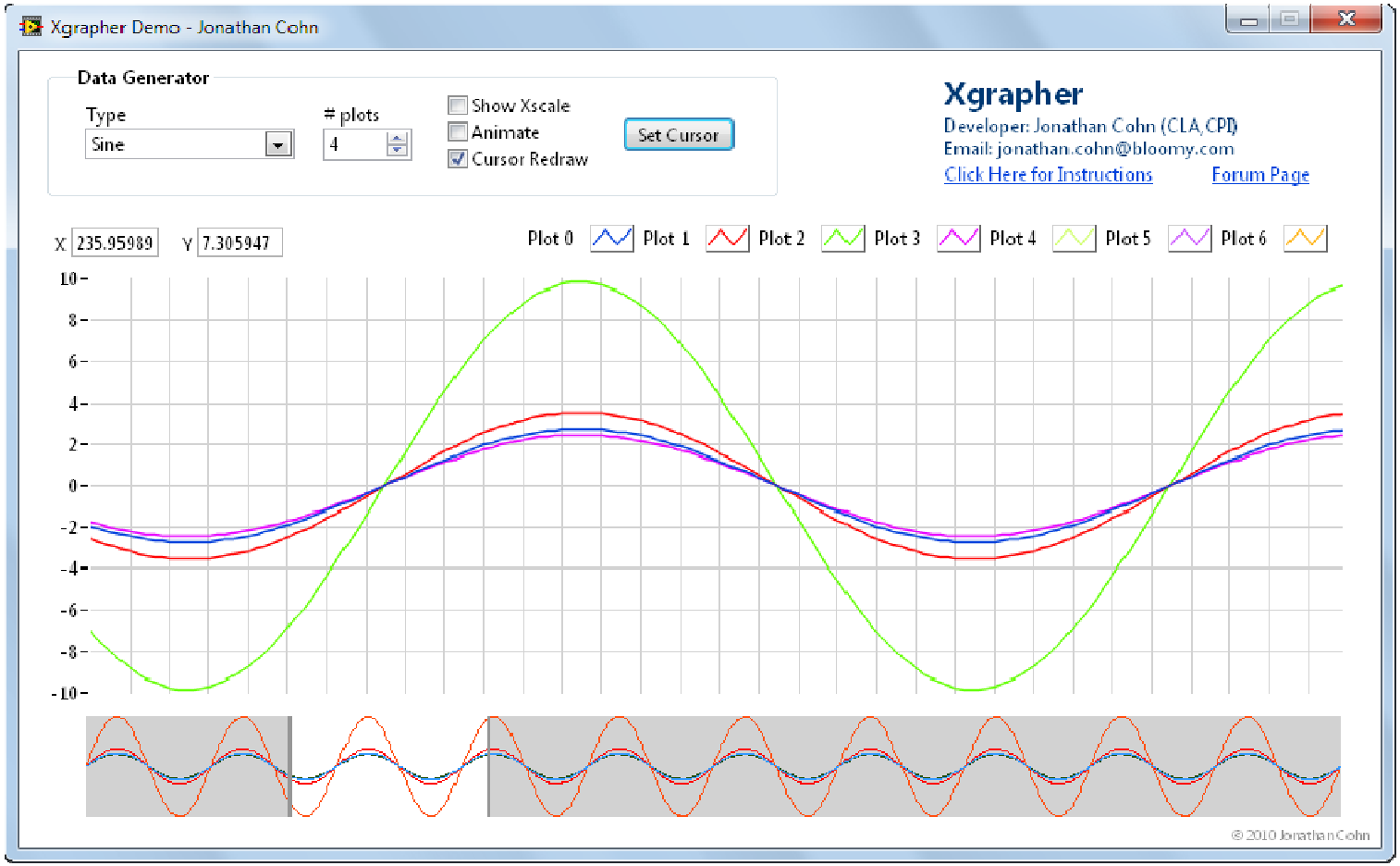
Gallery of Examples



Credit: Simon Hogg
NI Community UI Interest Group

NATIONAL INSTRUMENTS

Gallery of Examples



Credit: Jonathan Cohn – Bloomy Controls
LabVIEW Example Code Contest 2010 UI Controls Category Winner

NATIONAL INSTRUMENTS

Organization of Complex UIs

Tab Control

- Use when controls don't all have to be visible at once
- Limit to the number of tabs you can gracefully use
- Loads all controls into memory at once

Category and Content View using SubPanels

- Use to dynamically decide which controls to display
- No limit to the number you can gracefully interact with
- You control when the VI is loaded or release from memory

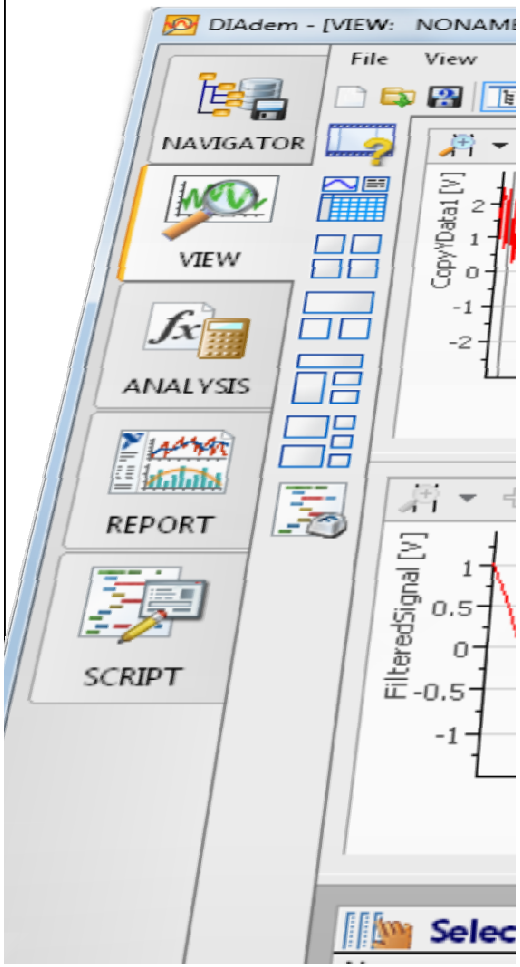
Using a Tab Control

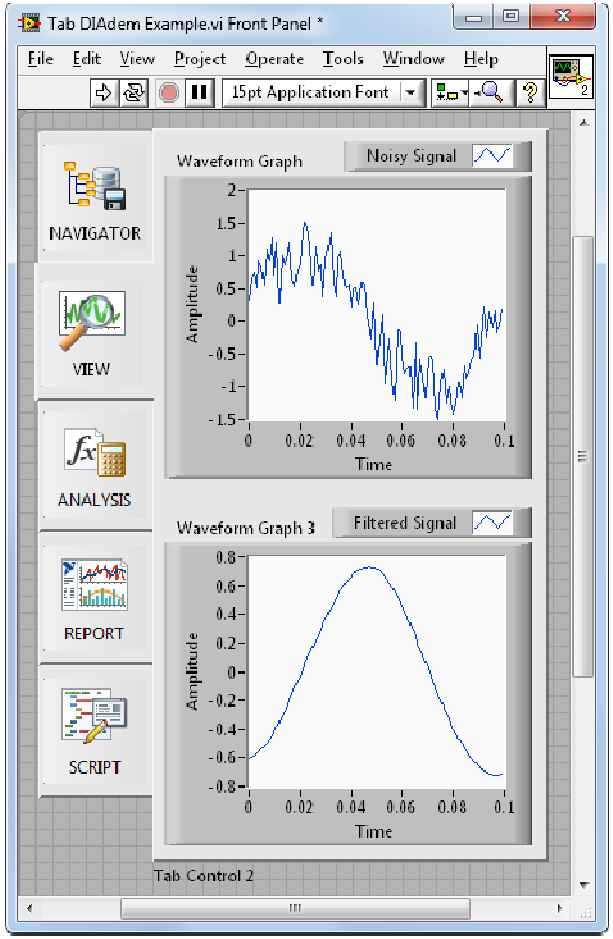
1. Add to front panel
2. Customize tab control
3. Populate with controls

- More flexible than most people realize
  - Transparent tabs
  - Vertical tabs
  - Tabs with images

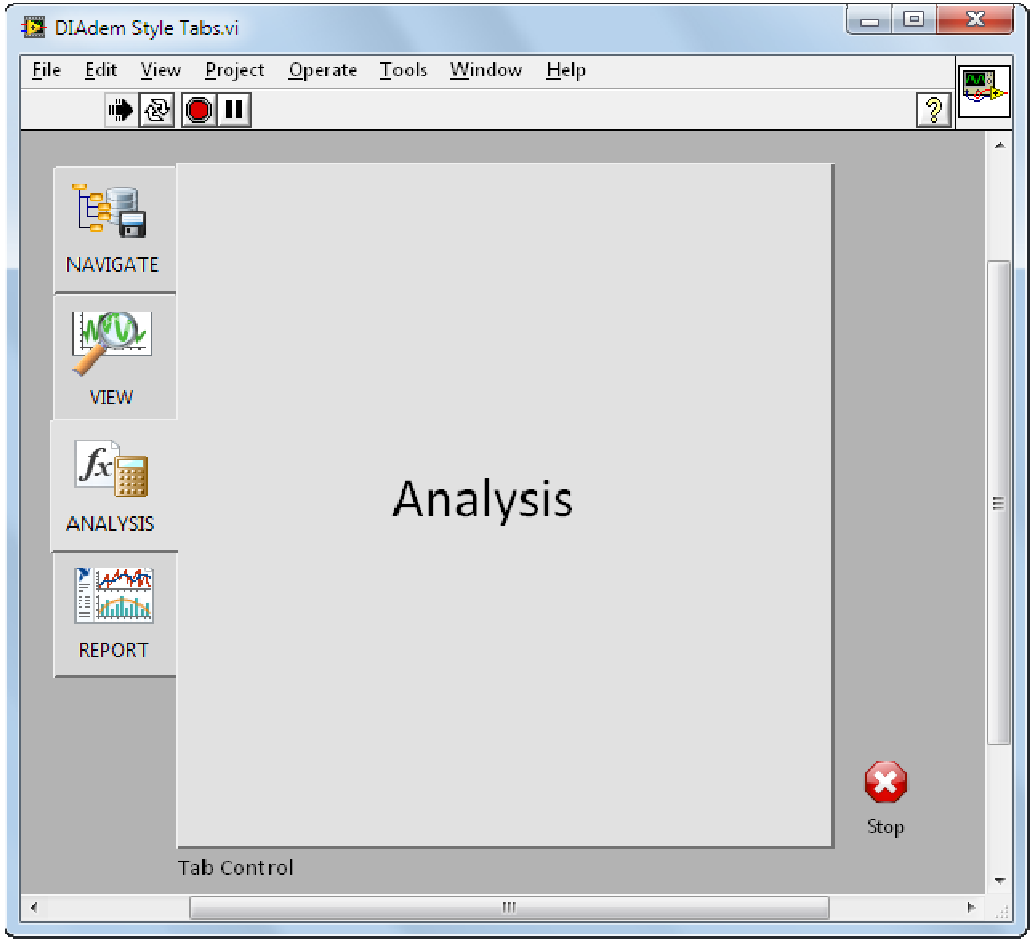
Example Tab Control Application

Using LabVIEW to mimic NI DIAdem, software that outshines Excel for data postprocessing





NATIONAL INSTRUMENTS



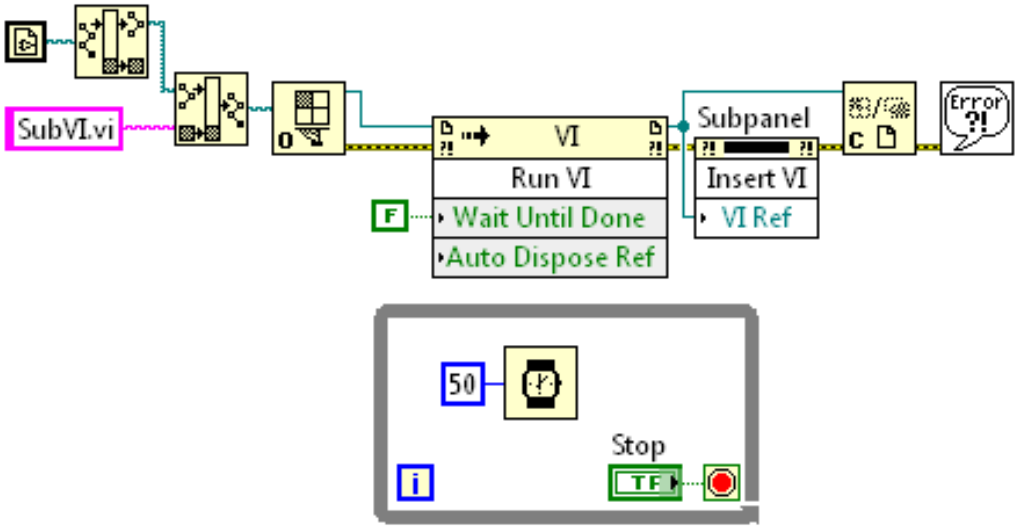
Disguising a Tab Control with Transparent or Image Tabs

DEMO

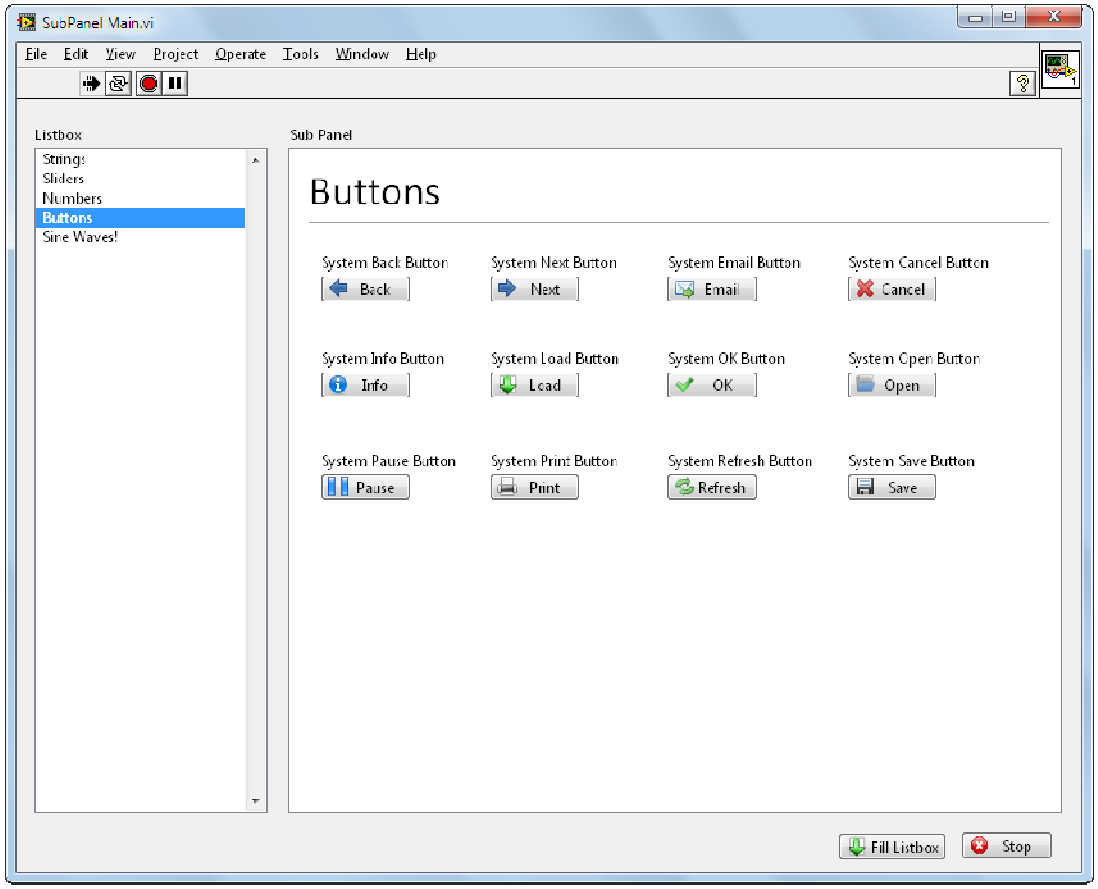
NATIONAL INSTRUMENTS

Using a SubPanel

1. Determine higher level VI screen real estate
2. Develop size appropriate, modular SubVIs
3. Dynamically Run SubVI
4. Dynamically insert SubVI into subpanel



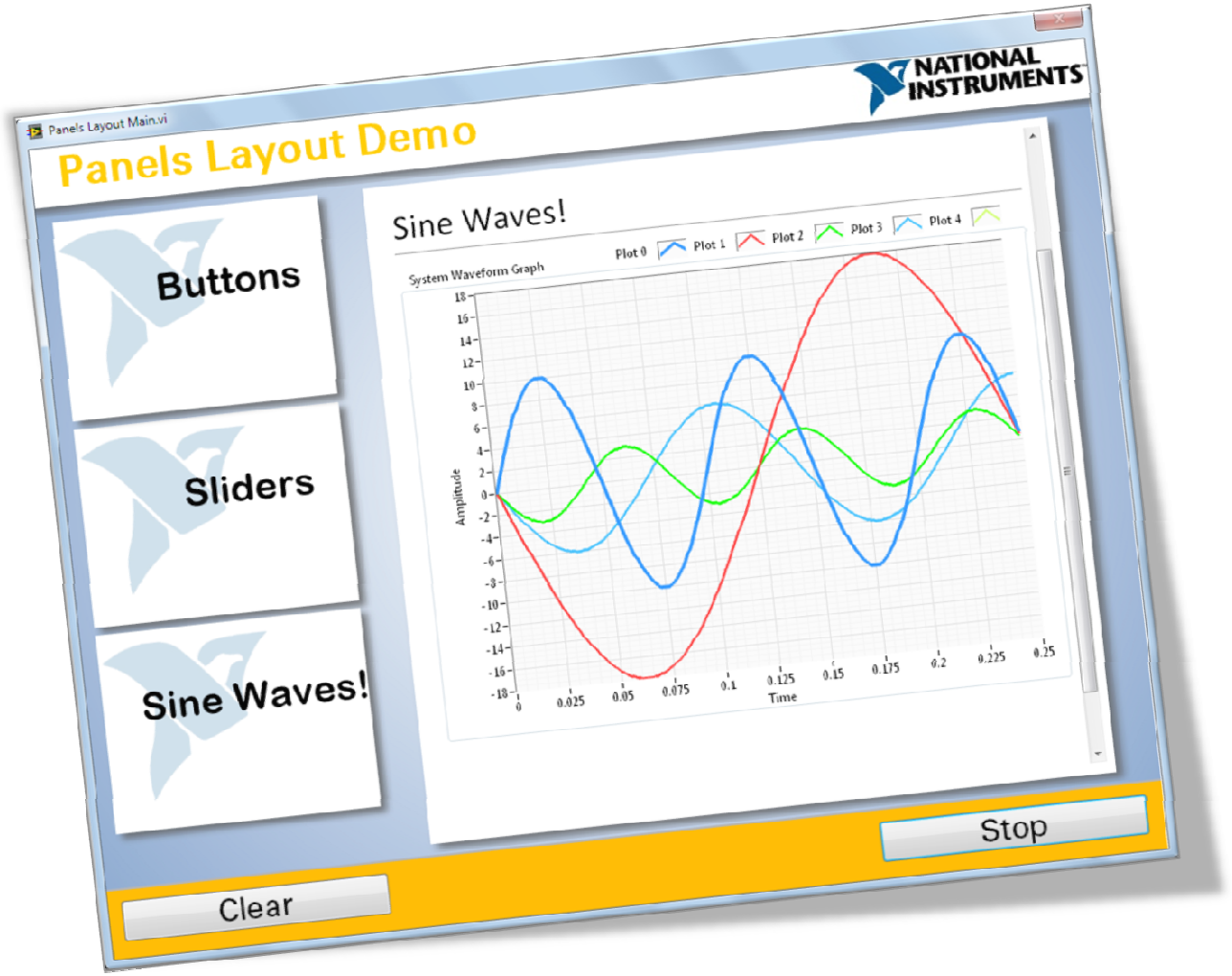
NATIONAL INSTRUMENTS



Displaying Subpanels based upon User Selections

DEMO

Gallery of Examples



Credit: Simon Hogg
NI Community UI Interest Group

NATIONAL INSTRUMENTS

Indicating Progress of Slow Operations

- Splash screens are effective when applications have a long load time
  - Provide development and support information
- Busy cursors
- Progress bars

For tips on using busy cursors, view Part I of this presentation available on the NI Community UI Interest Group site

NATIONAL INSTRUMENTS

Using an Event Structure to Create a Splash Screen

DEMO

NATIONAL INSTRUMENTS

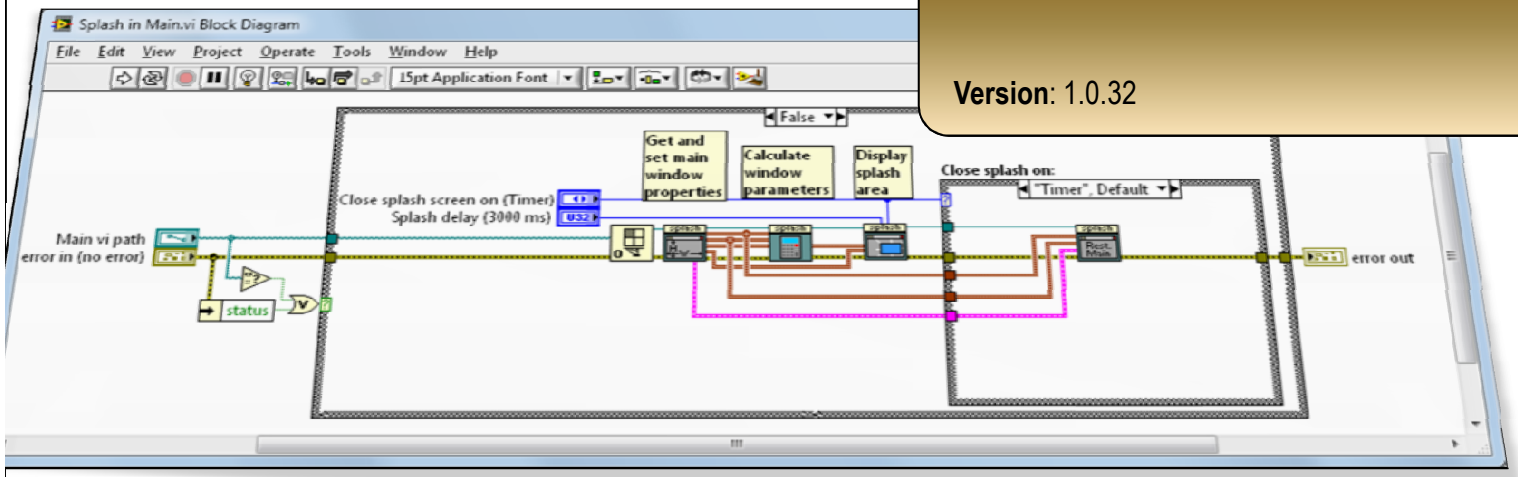
Gallery of Examples

For Support, Please Visit: http://www.ni.com/

Sine Wave Tester

Version: 1.0.32

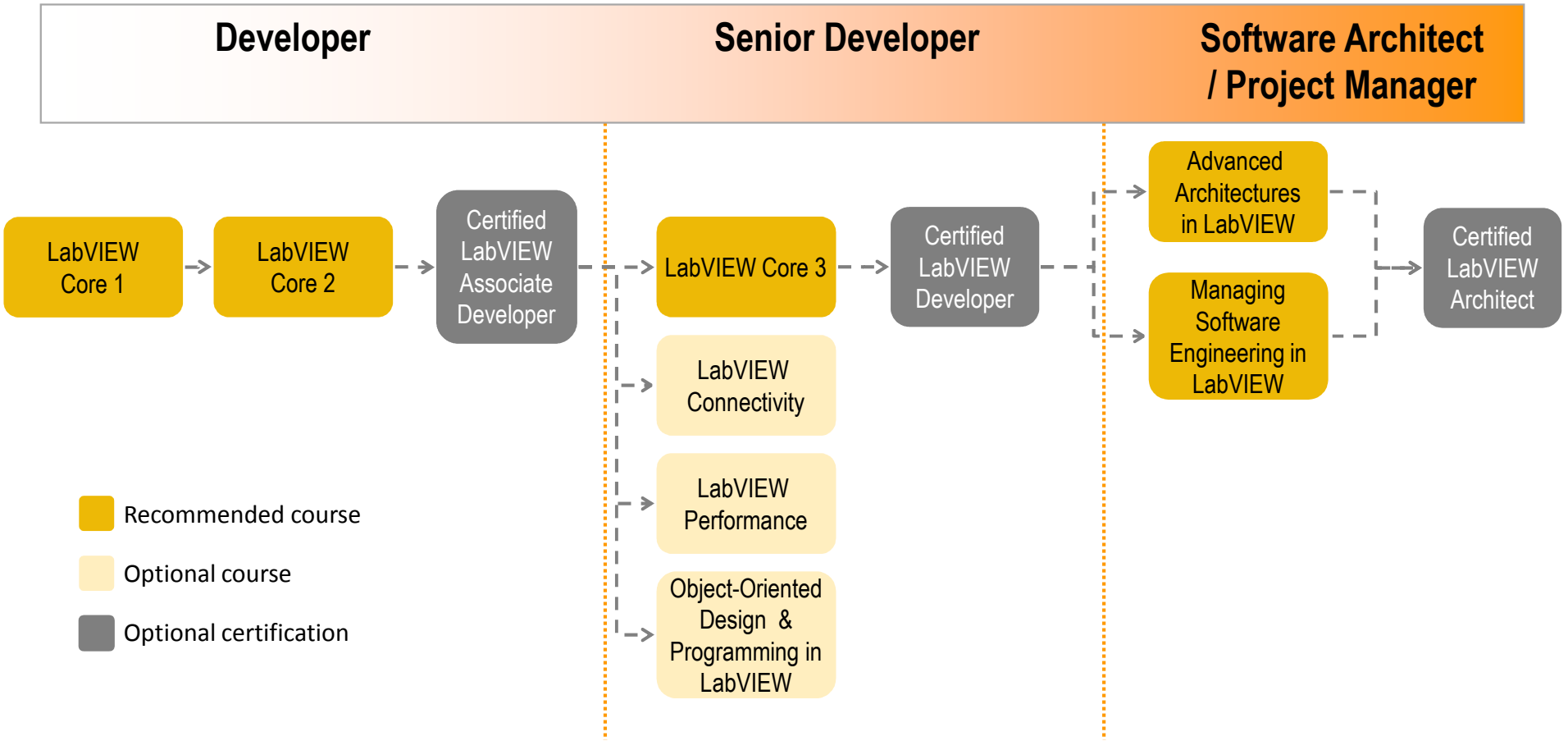
Author: Derrick Snyder



Framework Credit: Mark Ridgley, Logic PD
NI Developer Community

NATIONAL INSTRUMENTS

NI LabVIEW Certifications Align with Training



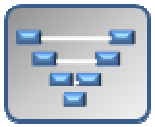
"Certification is an absolute must for anyone serious about calling himself a LabVIEW expert... At our organization, we require that every LabVIEW developer be on a professional path to become a Certified LabVIEW Architect."

- President, JKI Software, Inc.

Software Engineering Best-Practices

ni.com/largeapps


Software Engineering Tools


Development Practices


LargeApp Community