NIDays09

WORLDWIDE GRAPHICAL SYSTEM DESIGN

CONFERENCE

NATIONAL INSTRUMENTS

Software Engineering for LabVIEW Applications

Presenter name

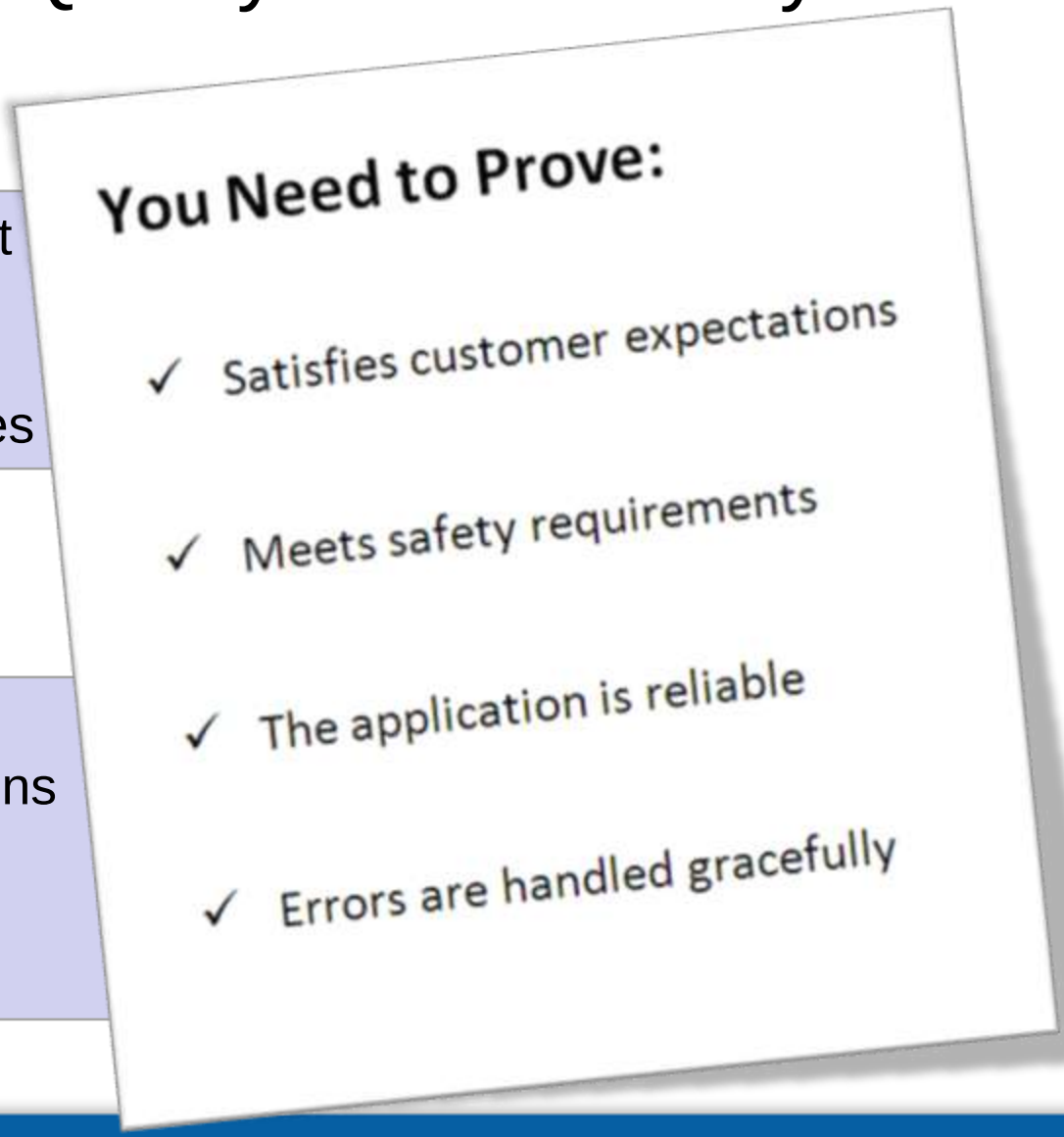
Ensuring Software Quality and Reliability

Goals

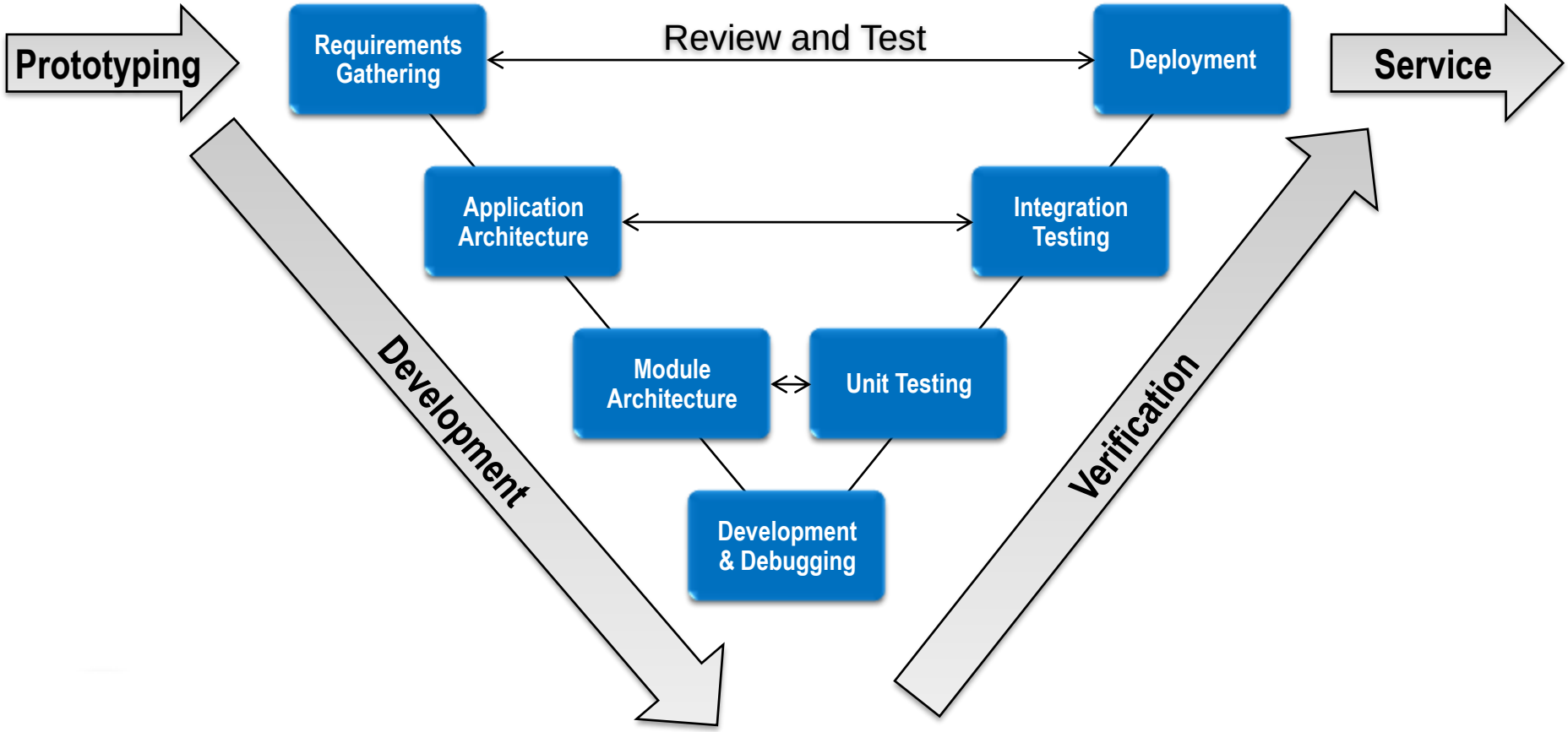
1. Deliver a working product
2. Prove it works right
3. Mitigate risk of failure
4. Avoid last-minute changes

Why?

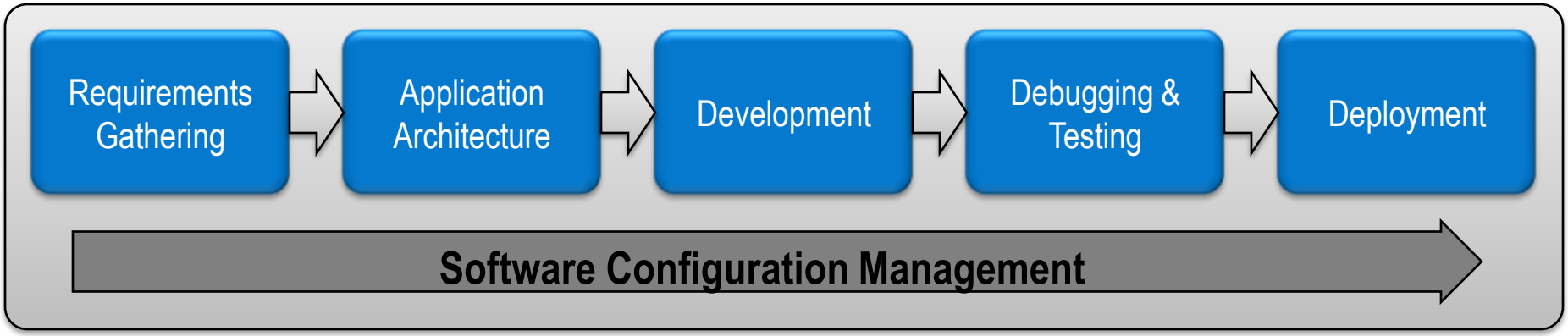
1. More complex software
2. Mission-critical applications
3. Team size is growing
4. Increased scrutiny
5. Decreased time



Software Engineering V-Model



The Software Engineering Process



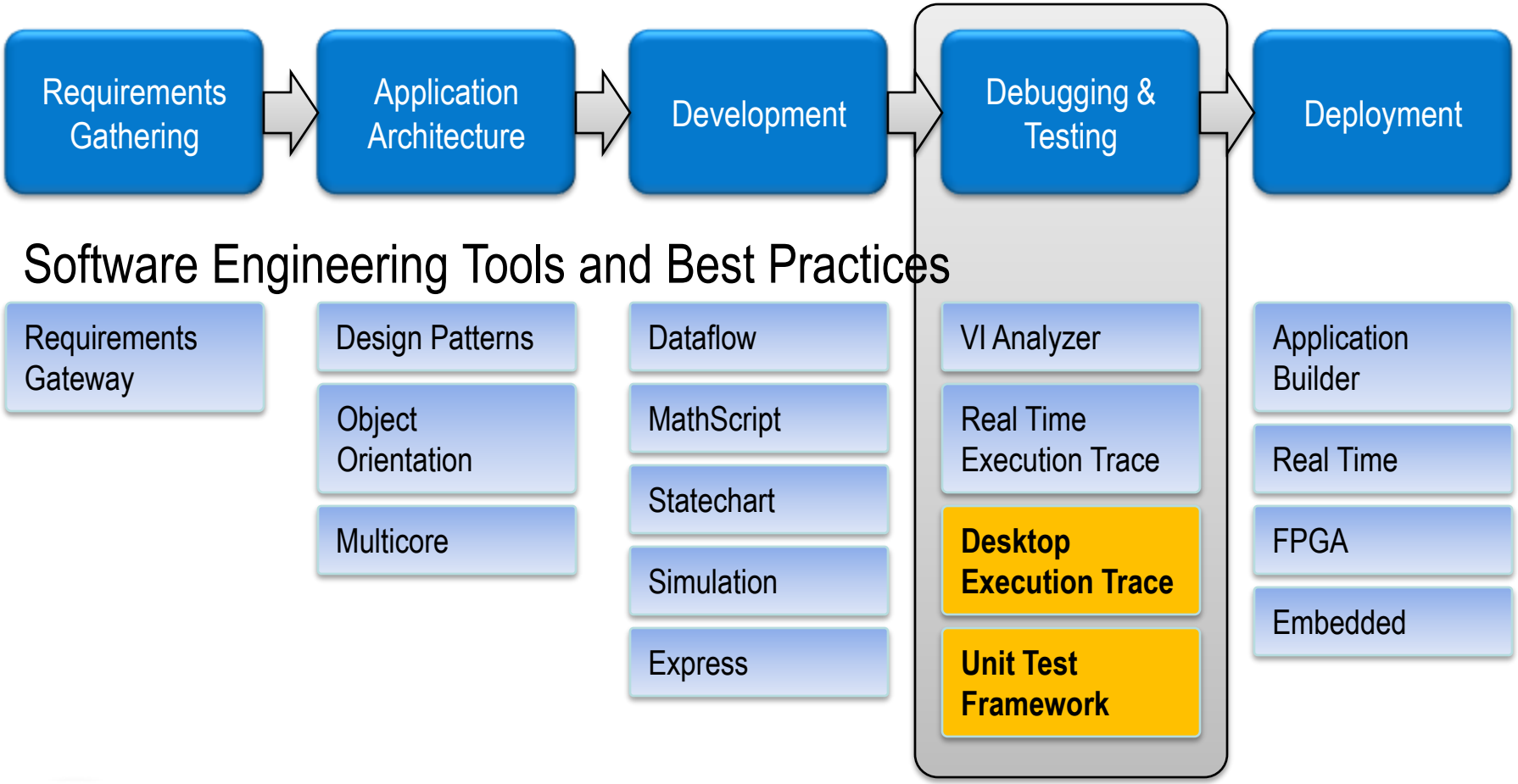
- Process is independent of programming language
- Demonstrate a particular process for certification
- Automate this process for **LabVIEW** with toolkits and add-ons
- SCM is applied throughout process

The Software Engineering Process

Software Engineering Tools and Best Practices

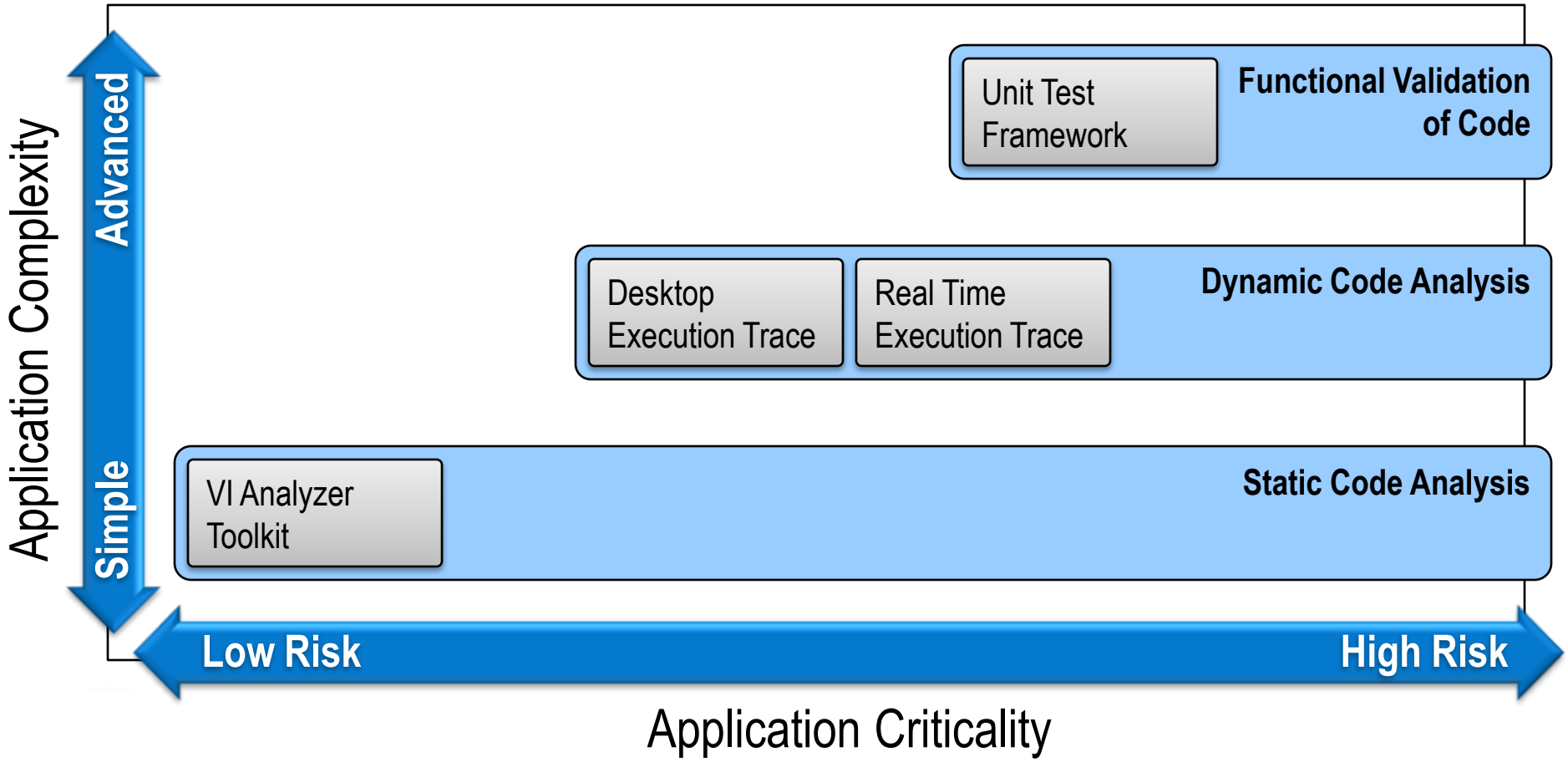
Requirements Gateway	Design Patterns	Dataflow	VI Analyzer	Application Builder
	Object Orientation	MathScript	Real Time Execution Trace	Real Time
	Multicore	Statechart	Desktop Execution Trace	FPGA
		Simulation	Unit Test Framework	Embedded
		Express		

The Software Engineering Process



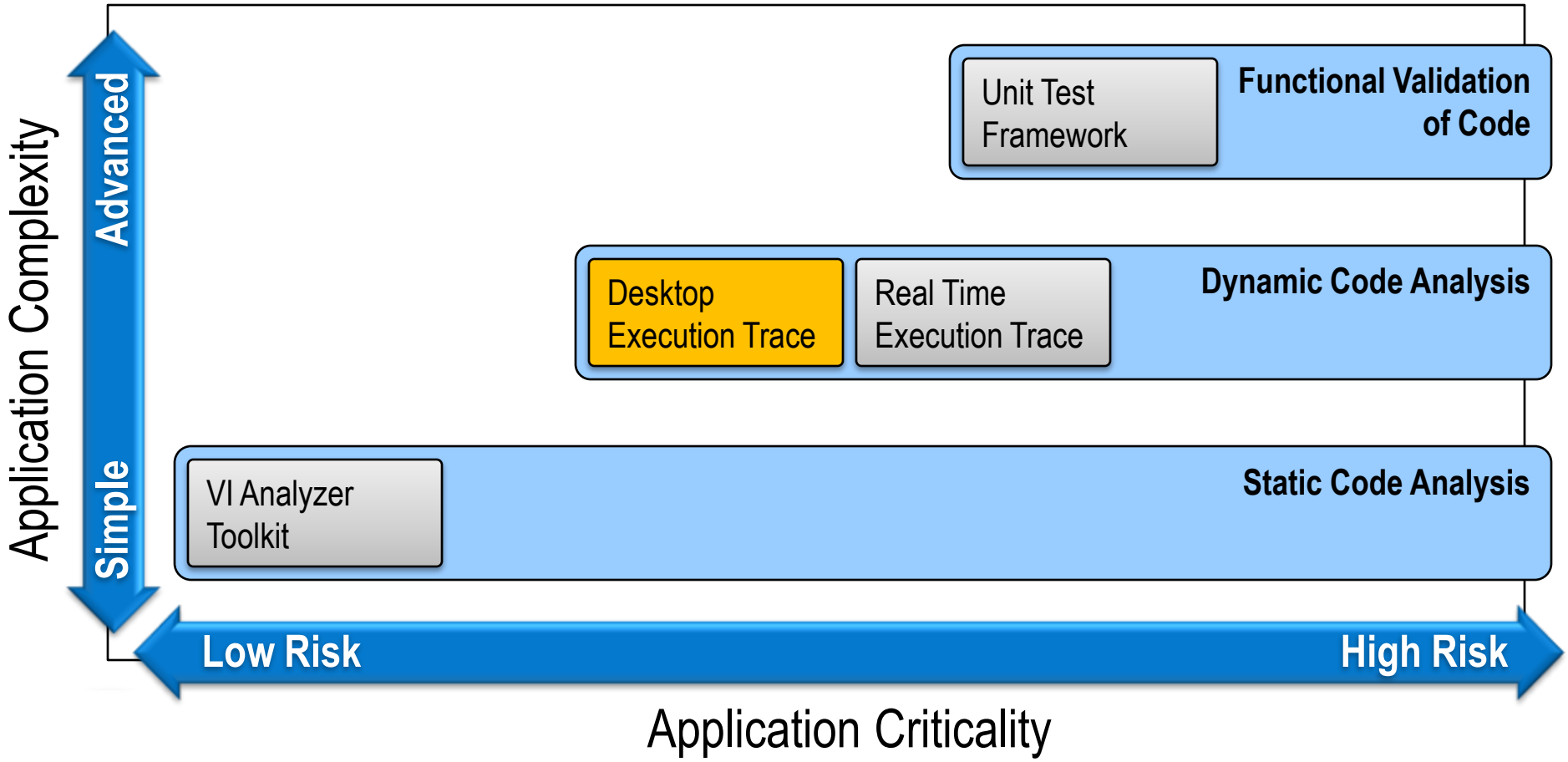
Tools for Debugging and Testing

Debugging & Testing



Tools for Debugging and Testing

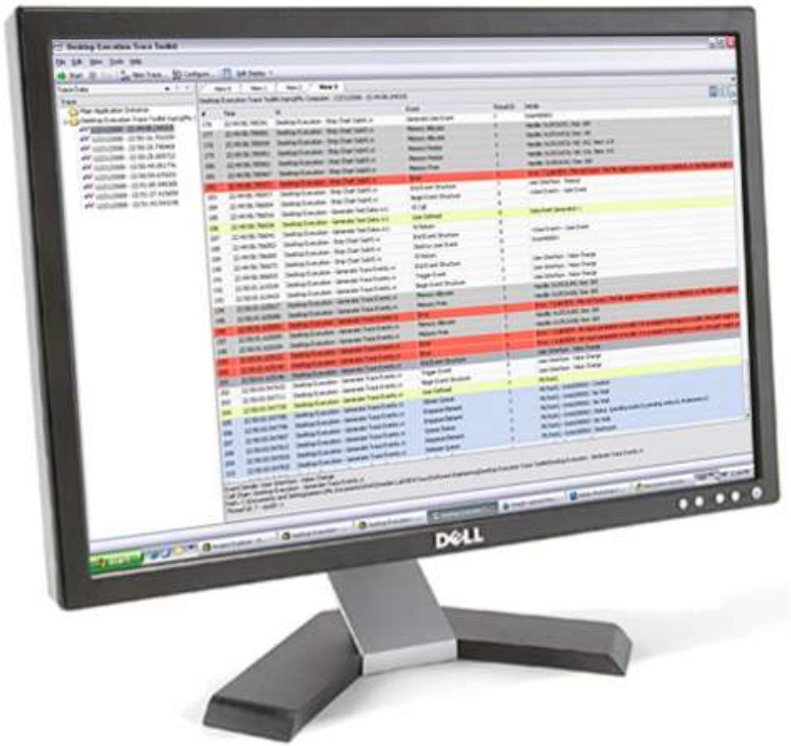
Debugging & Testing



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?

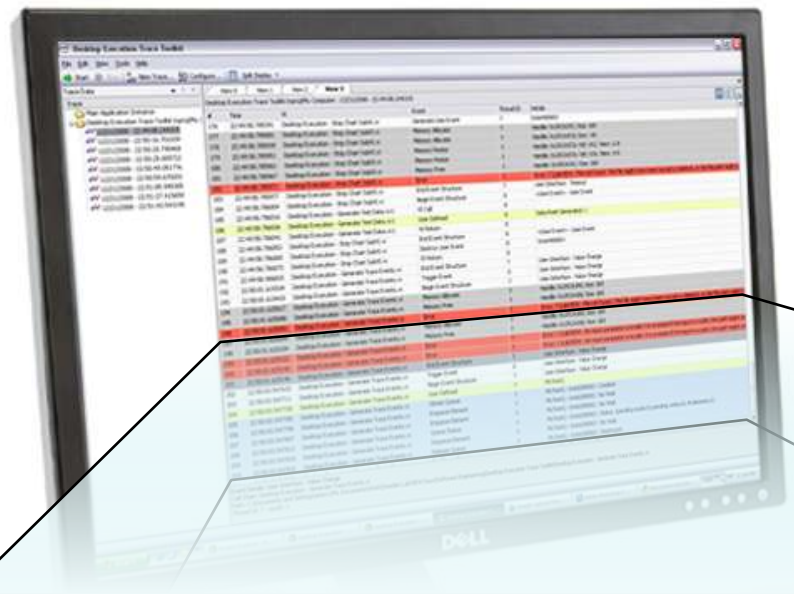
Desktop Execution Trace Toolkit



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
- Reference Leaks
- Thread ID
- Unhandled Errors

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 of LabVIEW VIs

DEMO

Trace Production Systems Remotely

LabVIEW Desktop Execution Trace Toolkit

Run-Time Execution Information

Network

VIs and Debuggable Executables

LabVIEW Real-Time Execution Trace Toolkit

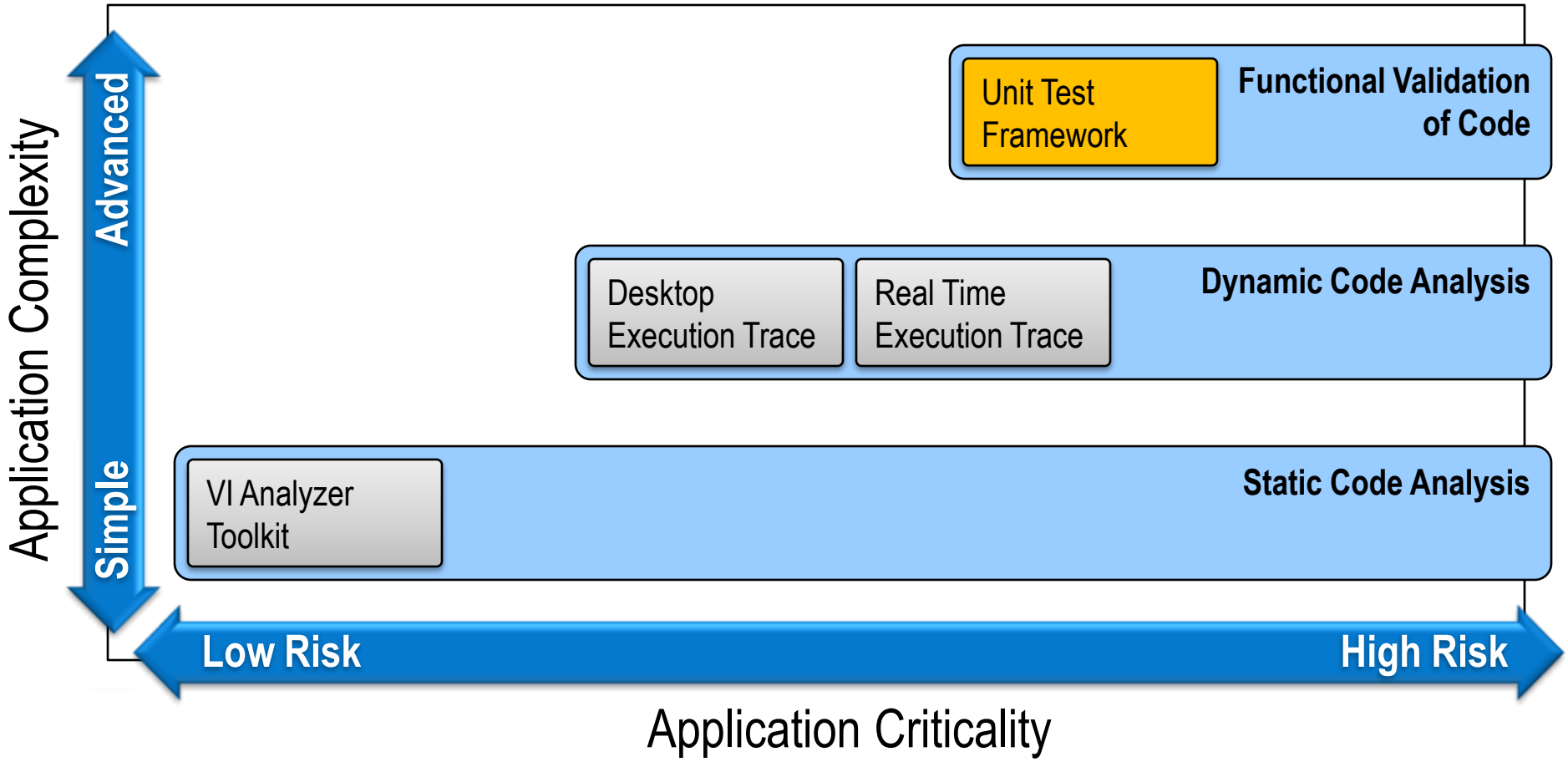
Run-Time Execution Information

Network

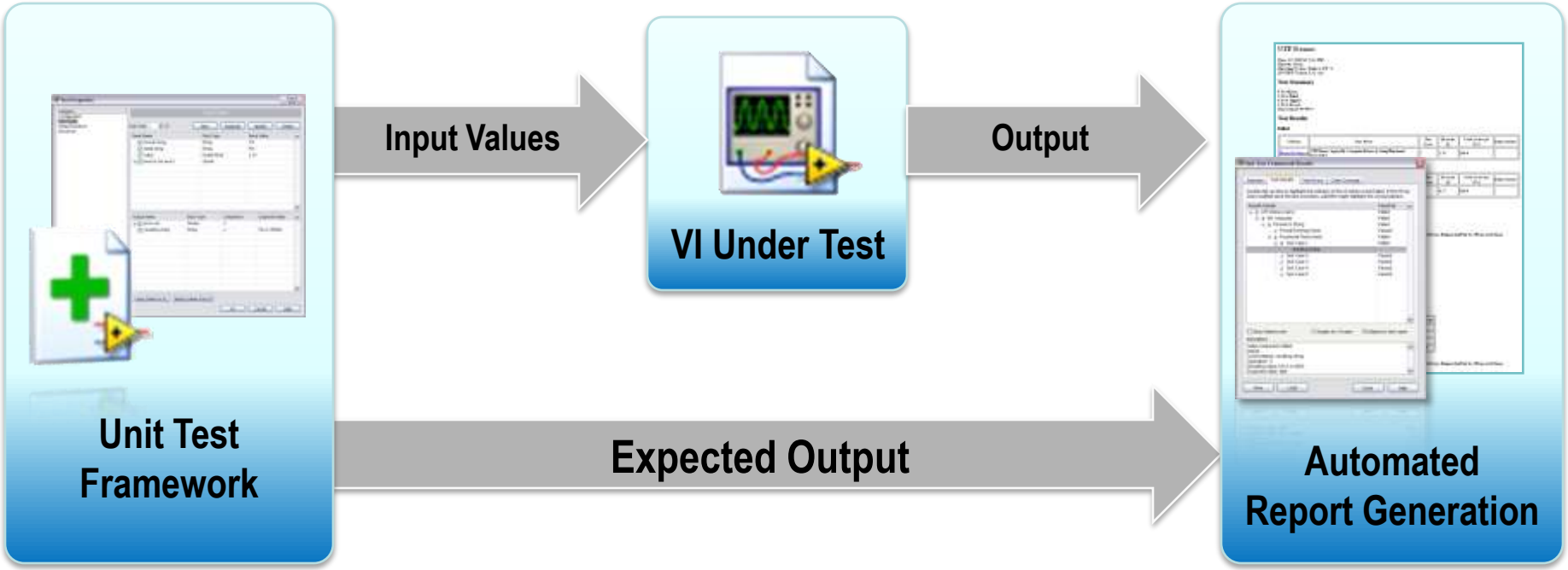
Deployed Real-Time Applications

Tools for Debugging and Testing

Debugging & Testing

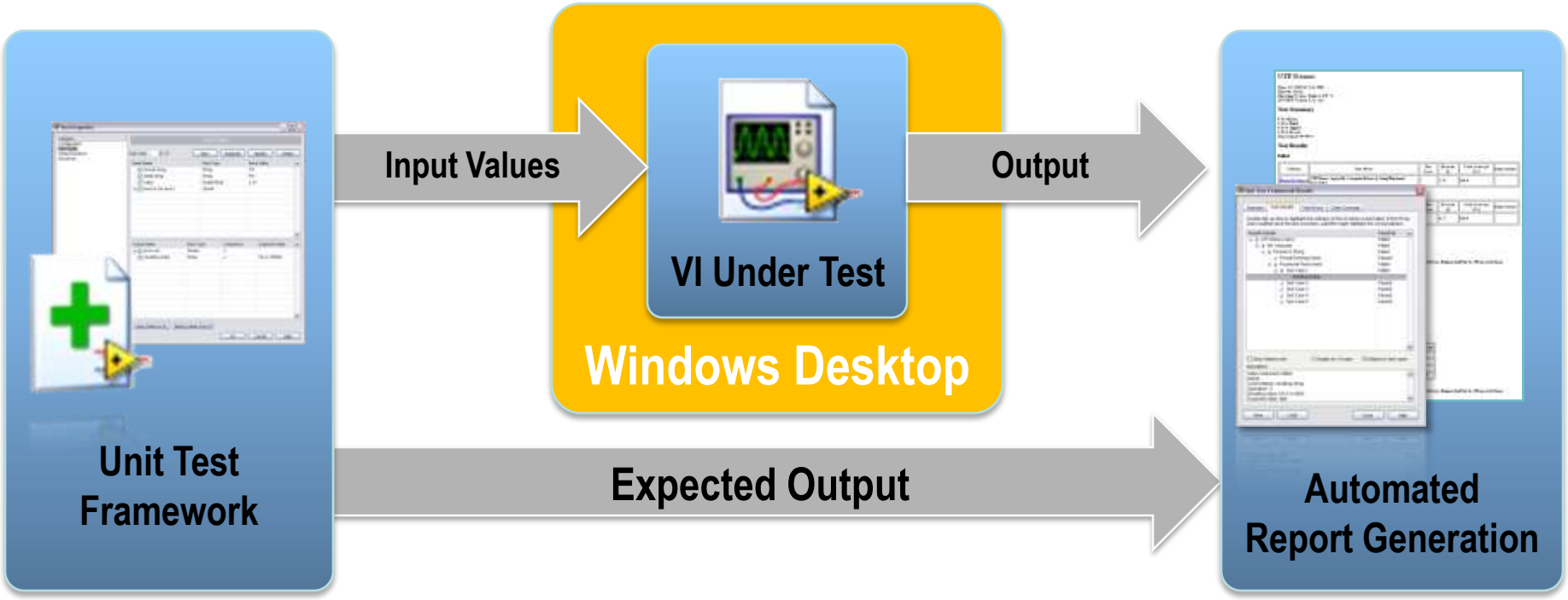


LabVIEW Unit Test Framework



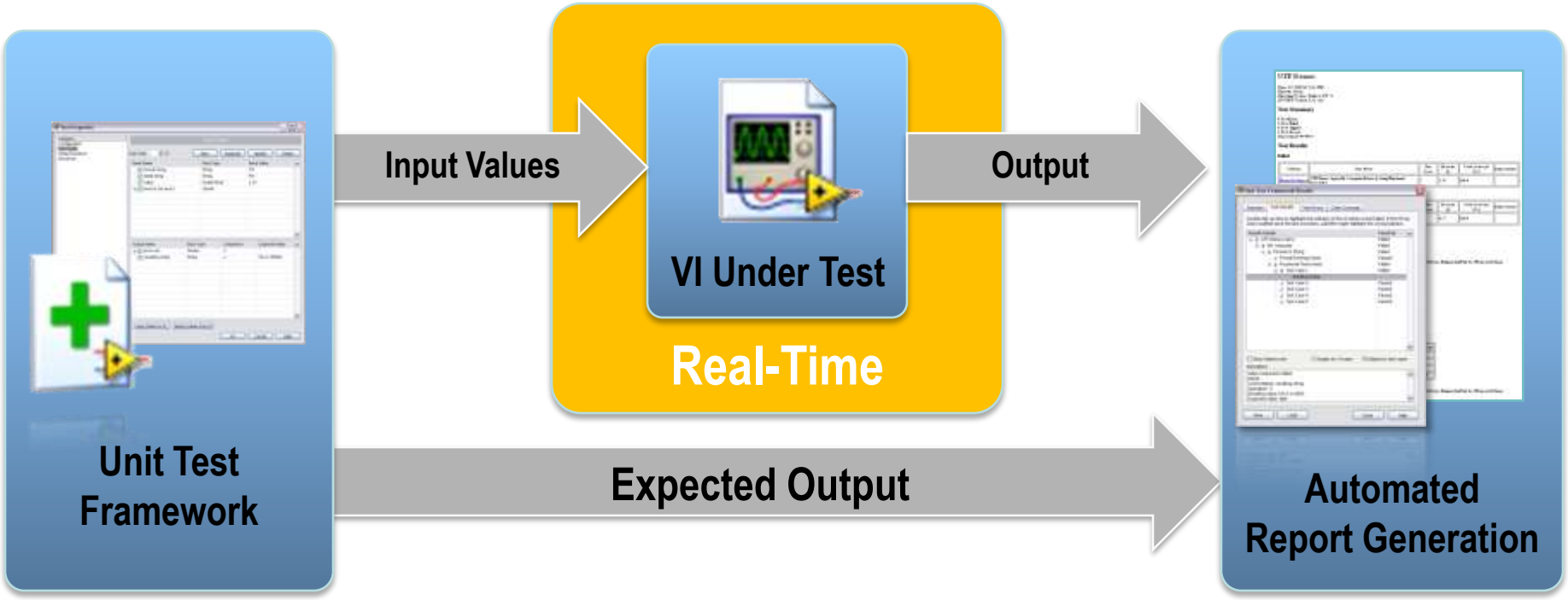
Test vector = Input value(s) + Expected output(s)

LabVIEW Unit Test Framework



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

LabVIEW Unit Test Framework

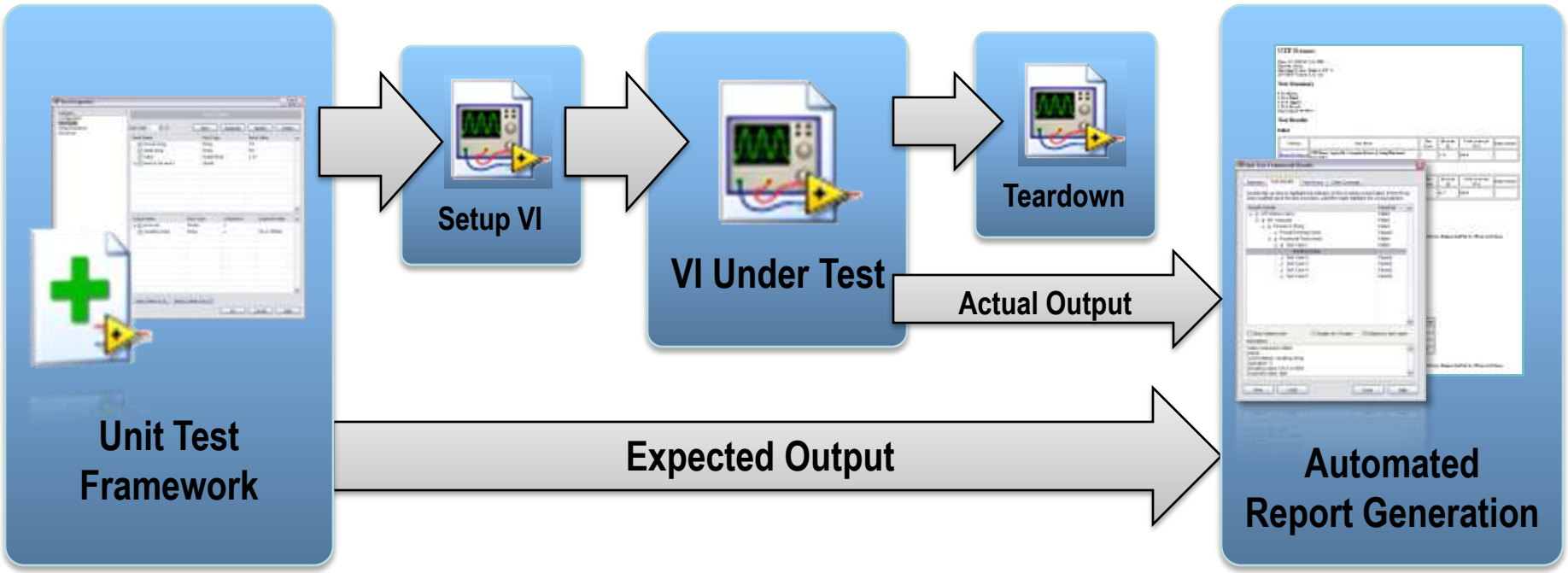


Test vector = Input value(s) + Expected output(s)

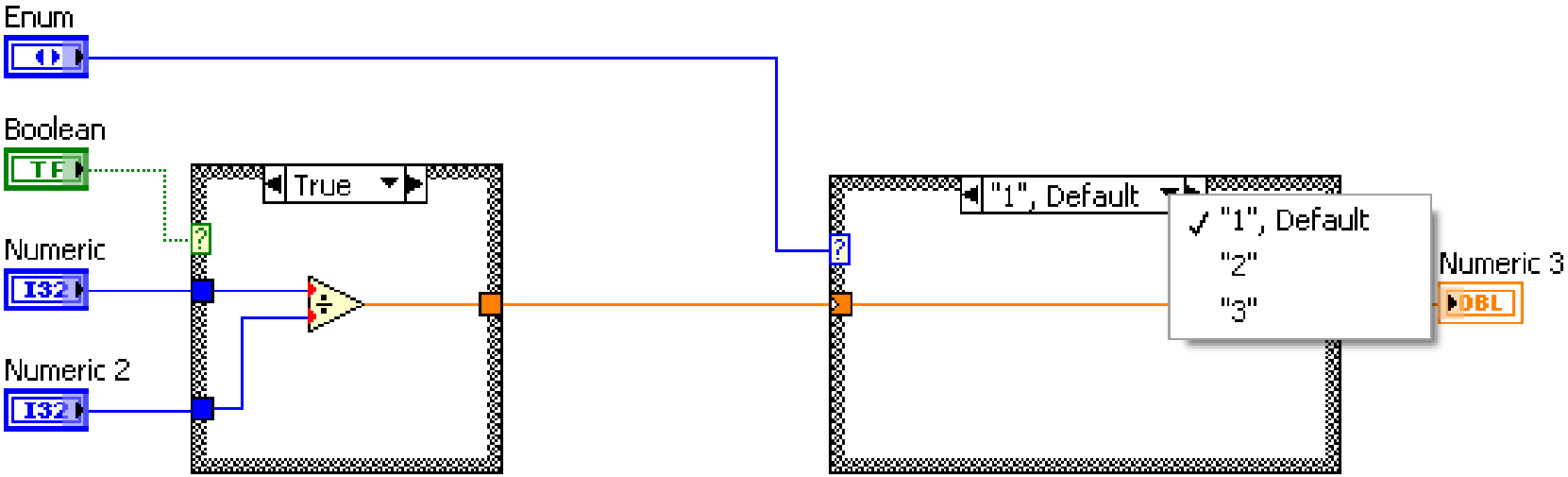
Creating Test Vectors with the LabVIEW Unit Test Framework

DEMO

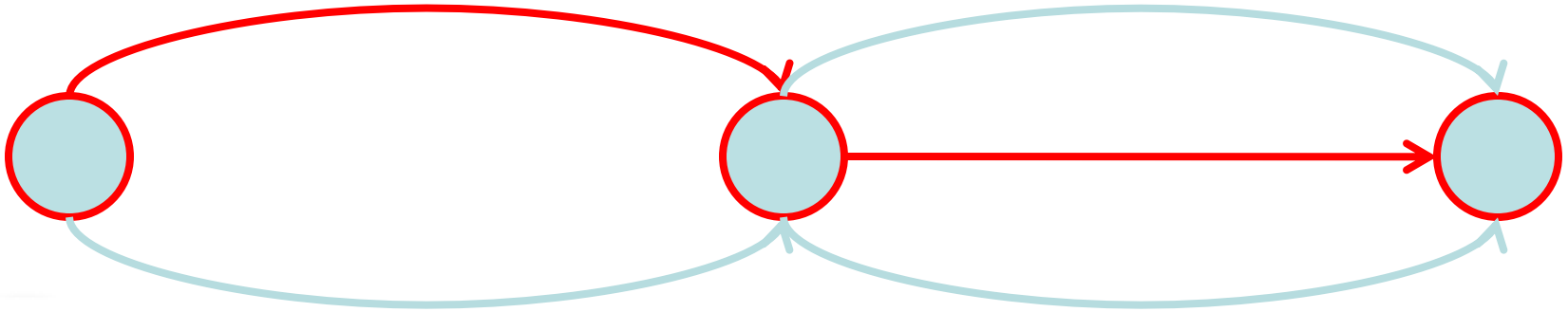
LabVIEW Unit Test Framework



Code Coverage Example



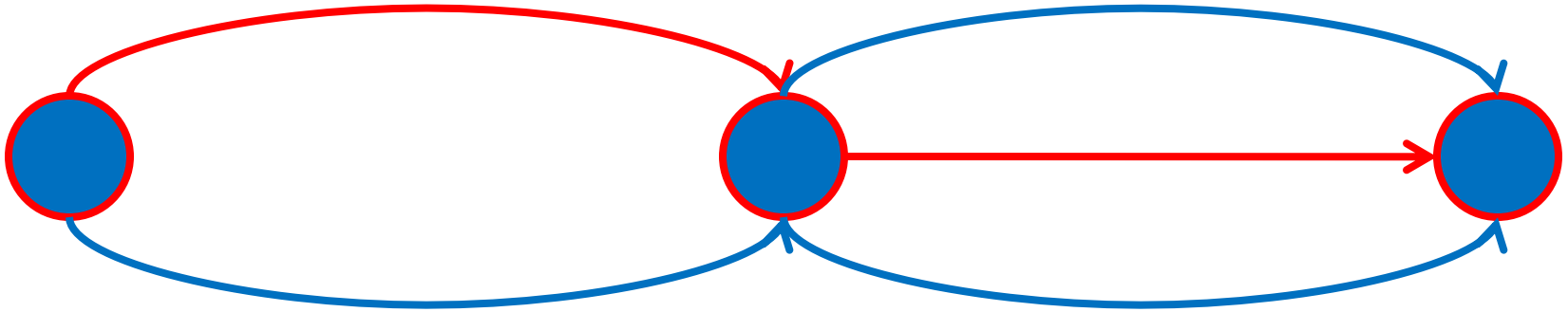
2 Edges and 1 Path. 6 Possible Routes. 50% Code Coverage



Code Coverage Example

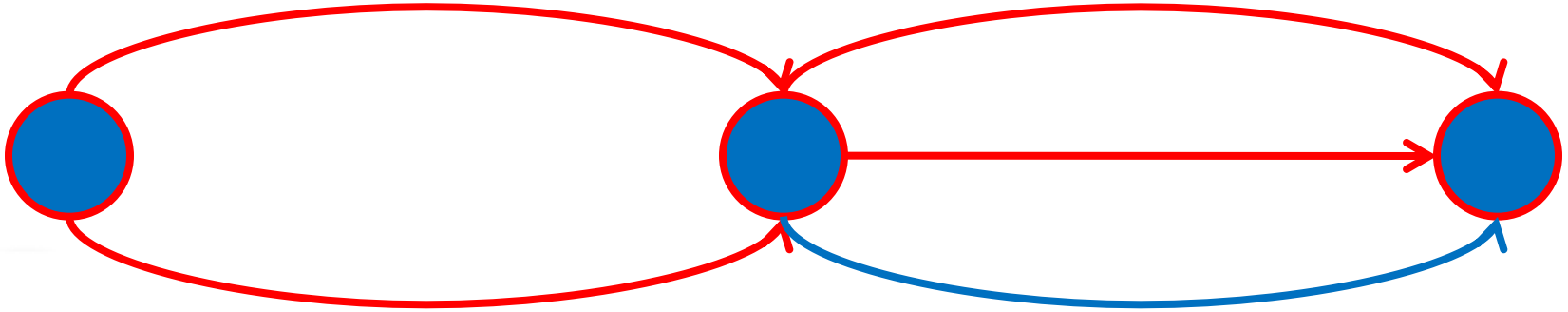
1ST Test Vector:

2 Edges and 1 Path. 6 Possible Routes. (2 + 1) / 6 = 50% Code Coverage



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

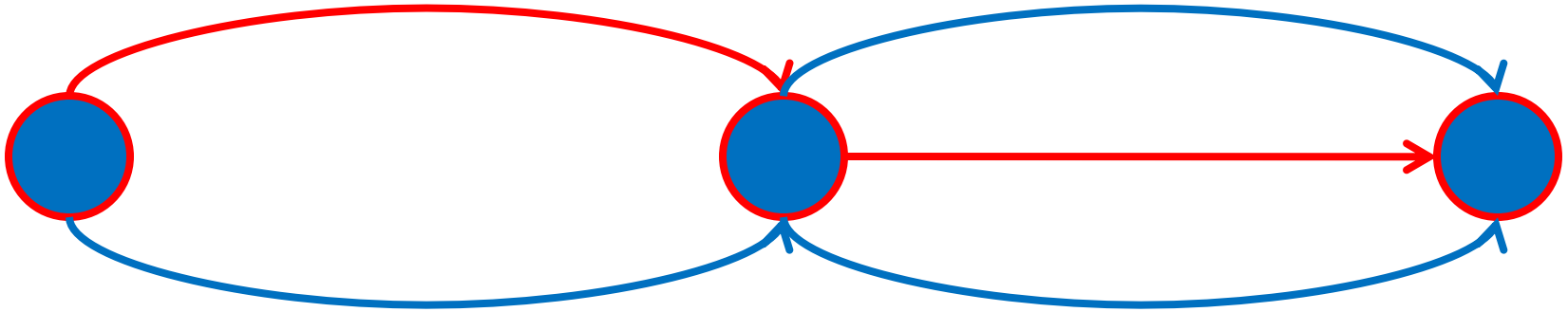
4 Edges and 1 Path. 6 Possible Routes. (4 + 1) / 6 = 83.33% Code Coverage



Code Coverage Example

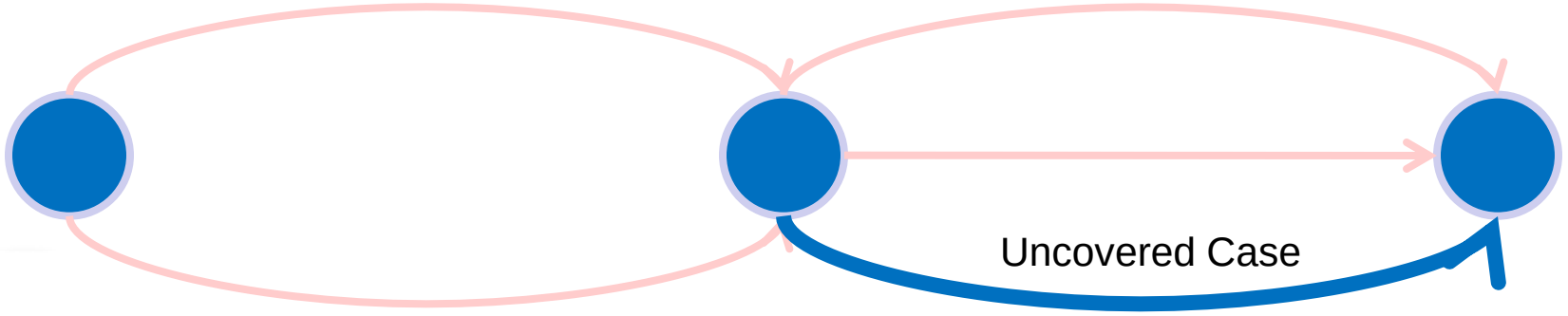
1ST Test Vector:

2 Edges and 1 Path. 6 Possible Routes. (2 + 1) / 6 = 50% Code Coverage



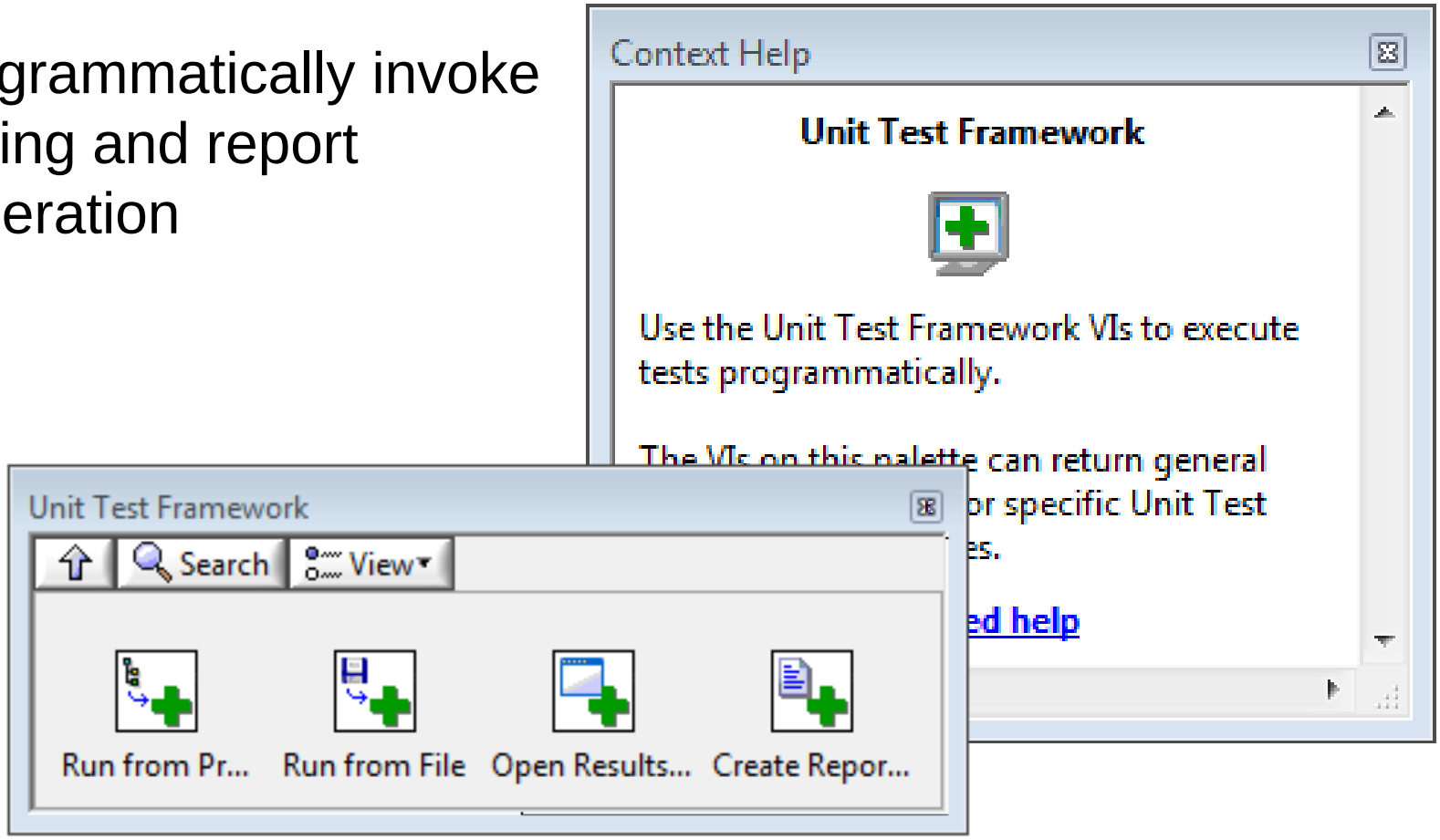
2nd Test Vector (aggregates covered code from 1st pass)

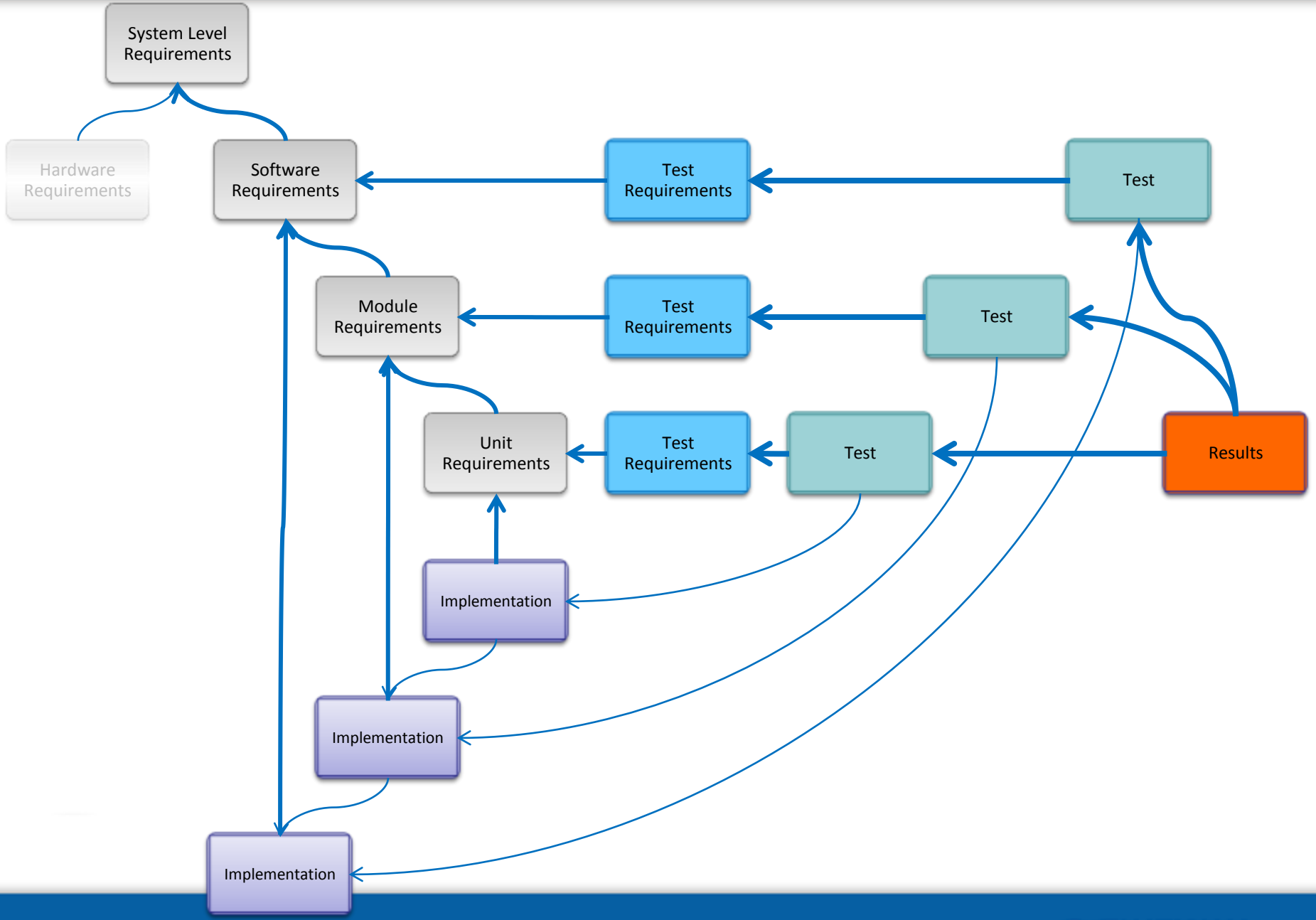
4 Edges and 1 Path. 6 Possible Routes. (4 + 1) / 6 = 83.33% Code Coverage



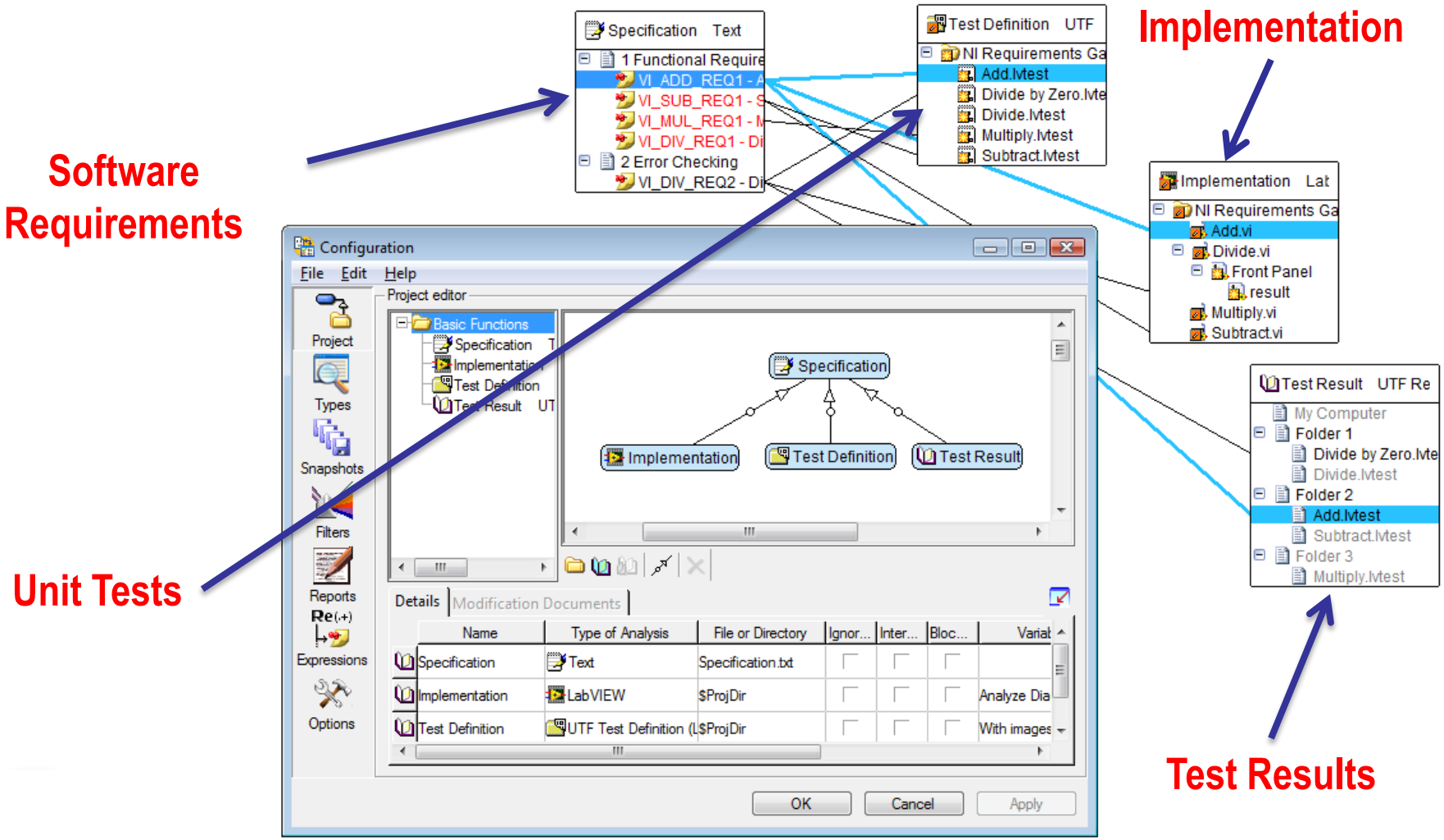
Programmatic Unit Testing

- Programmatically invoke testing and report generation

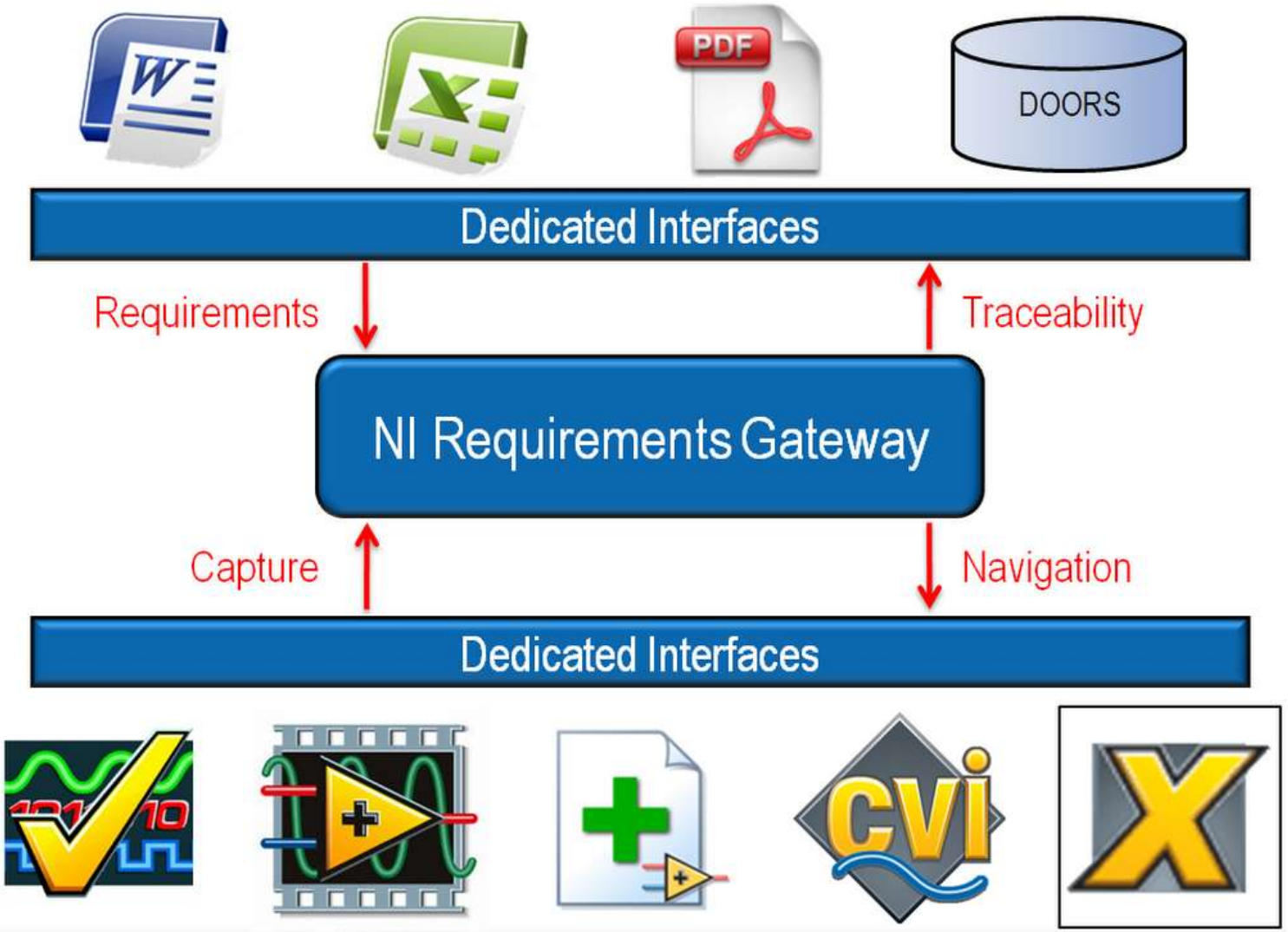




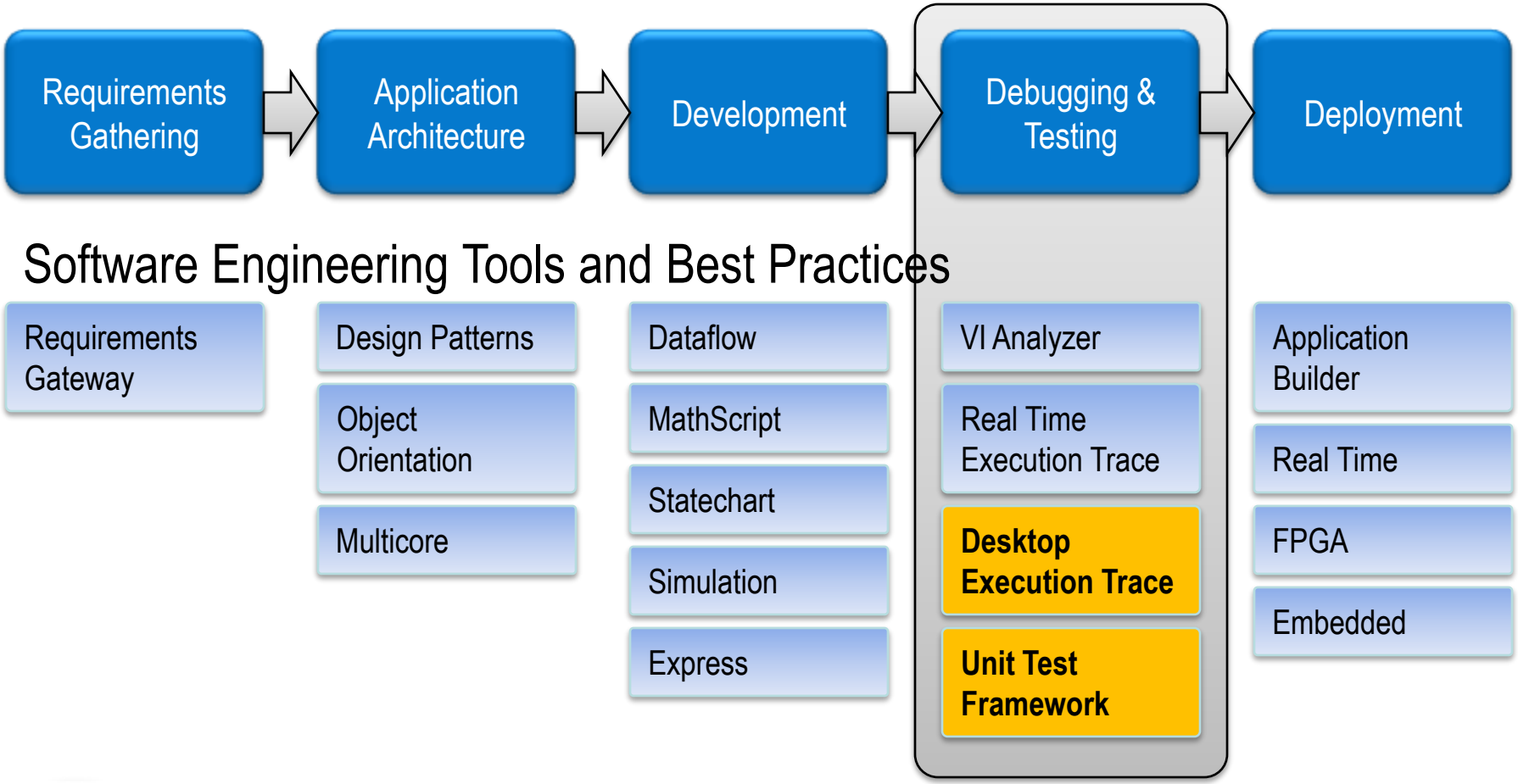
Integration with Requirements Gateway



Requirements Traceability Solution from NI



The Software Engineering Process



Software Engineering Best-Practices

Software Engineering Best Practices and Guidelines

Software Engineering with LabVIEW

View extensive documentation that provides procedural guidelines and recommendations across all phases of development for large software applications written in National Instrument's LabVIEW. Topics Include:

- Configuration Management and Source Control
- Requirements Gathering
- Application Architecture and Design
- Code Re-Use
- Debugging and Optimization
- Testing and Verification
- Application Deployment

Information about additional tools from National Instruments and third-party providers related to software engineering practices with LabVIEW is available on the LabVIEW Tools Network.

Click here for more about LabVIEW Software Engineering Tools.

ni.com/largeapps

LabVIEW Learning Paths