**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

---

**You Need to Prove:**

- Satisfies customer expectations
- Meets safety requirements
- The application is reliable
- Errors are handled gracefully
ISO 9000
Voluntary certification standard for consistent processes

CMMI
Process improvement model sponsored by the National Defense Industrial Association

DO-178B
FAA standard for avionics software

FDA 21 CFR Part 820
Medical device standard
National Instruments is ISO 9001 Certified (HW&SW)
Quality Management System (QMS)
Covers all of NI

New Product Introduction (NPI) process
Engineering/Marketing/Manufacturing/Sales

Software Engineering Process (SEP)
Engineering only
How SW Engineers and Project Managers execute NPI for SW products
A model for software development practices

Prescribes what needs to be done before moving on
• 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
• Defining a central repository of code
• Management of multiple developers
• Detection and resolution of code collisions
• Tracking behavioral changes
• Identification of changes are who made them
• Ensuring everyone has latest copy of code
• Backing up old code versions
• Managing all files, not just source code
Software Configuration Manager for LabVIEW

System Level View

Merge Graphical Code

Track Changes

Manage Files and Links

Integrate with SCC

View Revision History

ni.com
• Third-party source control integration with:
  - Microsoft Visual SourceSafe
  - Microsoft Team System
  - Perforce
  - Rational ClearCase
  - PCVS (Serena) Version Manager
  - MKS Source Integrity
  - Seapine Surround SCM
  - Borland StarTeam
  - Telelogic Synergy
  - ionForge Evolution
  - subVersion**

• Access SCC tools via LabVIEW Project
• Project specific settings*

*New in LabVIEW 8.5
**subVersion is open source and requires plug-in
The Software Engineering Process TOOLS

Software Engineering Tools and Best Practices

- Requirements Gateway
- Design Patterns
- Object Orientation
- Multicore
- Dataflow
- MathScript
- Statechart
- Simulation
- Express
- VI Analyzer
- Real Time Execution Trace
- Desktop Execution Trace
- Unit Test Framework
- Application Builder
- Real Time
- FPGA
- Embedded
Source Code Control Integration with LabVIEW

Software Engineering Tools and Best Practices

- **Requirements Gathering**
  - Design Patterns
  - Object Orientation
  - Multicore

- **Application Architecture**
  - Dataflow
  - MathScript
  - Statechart
  - Simulation
  - Express

- **Development**
  - VI Analyzer
  - Real Time Execution Trace
  - Desktop Execution Trace
  - Unit Test Framework

- **Debugging & Testing**
  - Application Builder
  - Real Time
  - FPGA
  - Embedded

- **Deployment**
• Agreeing upon what the application will do
• Defining acceptable and safe behaviors
• Defining tests for individual components
• Traceability from code to documents
<table>
<thead>
<tr>
<th>Upstream</th>
<th>Text</th>
<th>Downstream</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ_ControlLevel</td>
<td>The application must maintain a separate control algorithm for controlling tank levels.</td>
<td>Level.vi</td>
</tr>
<tr>
<td>REQ_ControlTemperature</td>
<td>The application must maintain a separate control algorithm for controlling temperature.</td>
<td>Temperature.vi</td>
</tr>
<tr>
<td>REQ_DisplayGraphInflow</td>
<td>The display must display a graph indicating the inflow rate.</td>
<td>Inflow Rate History</td>
</tr>
<tr>
<td>REQ_DisplayGraphLevel</td>
<td>The display must contain a graph indicating the tank level.</td>
<td>Tank Level History</td>
</tr>
<tr>
<td>REQ_DisplayGraphTemperature</td>
<td>The display must contain a graph indicating the tank temperature.</td>
<td>Tank Temperature History</td>
</tr>
<tr>
<td>REQ_DisplayHeatIndicator</td>
<td>The display must show whether furnace is on or off.</td>
<td>Heat Indicator</td>
</tr>
<tr>
<td>REQ_DisplayMinMaxLevels</td>
<td>The display must allow the operator to control the minimum and maximum levels for the tank.</td>
<td>Low Level Limit (Lbm)</td>
</tr>
<tr>
<td>REQ_DisplayMinMaxLevels</td>
<td>The display must allow the operator to control the minimum and maximum levels for the tank.</td>
<td>High Level Limit (Lbm)</td>
</tr>
<tr>
<td>REQ_DisplayMinMaxTemp</td>
<td>The display must allow the operator to control the minimum and maximum levels for the tank temperature.</td>
<td>Low Temp Limit (deg F)</td>
</tr>
</tbody>
</table>
Requirements Traceability Solution from NI

Dedicated Interfaces

DOORS

NI Requirements Gateway

Requirements

Traceability

Capture

Navigation

Dedicated Interfaces

ni.com
1. Requirements defined in Word, Excel, PDF, DOORS, RequisitePro, ...

2. LabVIEW references requirements

3. Define a project to indicate what files are involved
• Determine how code will be divided based upon functionality
• Develop standard interfaces and APIs for code modules
• Ensure code scales and is readable and maintainable
• LabVIEW Design Patterns
• LabVIEW Statechart Module
• UML Tools
• Object-Oriented Design
• Implement the application
• Fulfill requirements documentation
• Detect and resolve incorrect code behavior
• Functionality passes test parameters
• Abide by style guidelines
• Optimize performance and execution
Develop Using Multiple Models of Computation

Requirements Gathering → Application Architecture → Development → Debugging & Testing → Deployment

ni.com
• Iterative testing during development
• Manual peer code review
• Static and dynamic code analysis
• Validation against requirements
RISK

Minor
- Black box
- Static analysis

Moderate
- Black box
- Static analysis
- Code reviews

Major
- White box
- Black box
- Static analysis
- Code reviews
• Constantly compiling
• Highlight execution
• Breakpoint manager
• Diagram cleanup
• National Instruments Style Guideline
• VI Profiler
• Coercion (Memory) Dots
• Conditional disable structures
• VI Analyzer Toolkit
• RT Execution Trace Toolkit
• Desktop Execution Trace
Debugging & Testing TOOLS

- Debugging & Testing
  - Unit Test Framework
  - Functional Validation of Code
  - Dynamic Code Analysis
  - Static Code Analysis
  - Desktope Execution Trace
  - Real Time Execution Trace
- Simple
  - VI Analyzer Toolkit
- Advanced
- Application Complexity
- Low Risk
- High Risk
- Application Criticality

ni.com
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?
Monitor the execution of LabVIEW code at run-time in order to debug common problems in large applications such as memory leaks and un-handled errors.

**Trace During Run-Time:**
- Event Structures
- Memory Allocation
- Queues / Notifiers
- Reference Leaks
- Thread ID
- Unhandled Errors
- Dynamic / Static SubVIs
- Custom User Strings
Trace During Run-Time:

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

<table>
<thead>
<tr>
<th>SubVI Name</th>
<th>Event Type</th>
<th>Event Count</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strip Chart SubVI.vi</td>
<td>Memory Resize</td>
<td>7</td>
<td>Handle: 0x25CA3C8; Old: 142; New: 118</td>
</tr>
<tr>
<td>Strip Chart SubVI.vi</td>
<td>Memory Allocate</td>
<td>7</td>
<td>Handle: 0x25CA3C8; Size: 142</td>
</tr>
<tr>
<td>Strip Chart SubVI.vi</td>
<td>Memory Resize</td>
<td>7</td>
<td>Handle: 0x25CA3C8; Old: 142; New: 118</td>
</tr>
<tr>
<td>Strip Chart SubVI.vi</td>
<td>Error</td>
<td>7</td>
<td>Error: 7 (LabVIEW: File not found. The file might have been deleted or moved.)</td>
</tr>
<tr>
<td>Generate Trace Events.vi</td>
<td>User Defined</td>
<td>7</td>
<td>MyTestQ</td>
</tr>
<tr>
<td>Generate Trace Events.vi</td>
<td>Obtain Queue</td>
<td>7</td>
<td>MyTestQ - 0x66200002 : Created</td>
</tr>
<tr>
<td>Generate Trace Events.vi</td>
<td>Enqueue Element</td>
<td>7</td>
<td>MyTestQ - 0x66200002 : No Wait</td>
</tr>
</tbody>
</table>
LabVIEW Desktop Execution Trace Toolkit

Network

Run-Time Execution Information

VIs and Debuggable Executables

LabVIEW Real-Time Execution Trace Toolkit

Network

Run-Time Execution Information

Deployed Real-Time Applications
Automatically test and validate VIs against requirements in order to demonstrate that it is working correctly according to design documents.

Test vector = Input value(s) + Expected output(s)
LabVIEW Unit Test Framework

- Unit Test Framework
- Windows Desktop
- VI Under Test
  - Input Values
  - Output
  - Expected Output
- Automated Report Generation

Test vector = Input value(s) + Expected output(s)
Test vector = Input value(s) + Expected output(s)
Requirements Traceability Solutions

Dedicated Interfaces

Requirements → NI Requirements Gateway → Traceability

Capture → Navigation

Dedicated Interfaces

ni.com
• Determine repeatable process
• Deploy to determine behavior on other targets
• Deliver a product to customer or end-user
• Make every installer and driver required for operation easily portable
• Remove information about sensitive code
• LabVIEW Applications Builder
• LabVIEW Project
  - Many deployment documents on www.ni.com
• Replication tool for LabVIEW Real-Time
  - Norton Ghost, etc for Windows
NI LabVIEW Application Builder
Learn more about:

Software Engineering
Configuration Management
Development Practices
Tools for Validation and Testing