Modern Development Workflows in LabVIEW

Greg Richardson

Distinguished Engineer and
LabVIEW Product Architect
NI Product R&D

EMERSON | NI

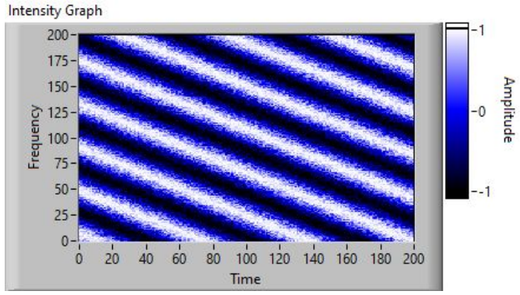
Margaret Hamilton
First programmer hired for Apollo module flight software

"I began to use the term **'software engineering'** to distinguish it from hardware and other kinds of engineering, yet treat each type of engineering as part of the overall systems engineering process."

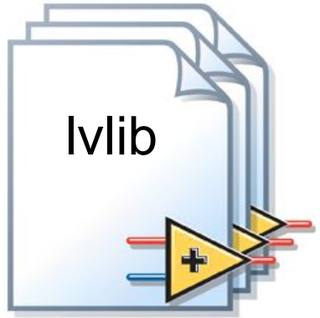
>32 years in LabVIEW R&D

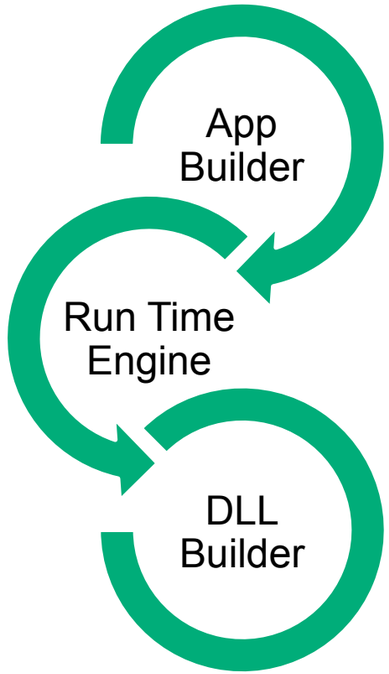
SuperSecretPrivateSpecialStuff=true
WindowsLongPaths=true (LV2021SP1)

DDE



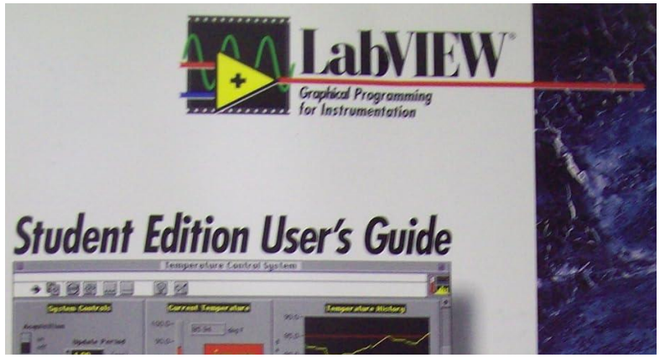
BRIDGEVIEW™





MICROSOFT® WINDOWS™ PowerPC™
Copyright © Microsoft Corporation 1985-1992.
All Rights Reserved.

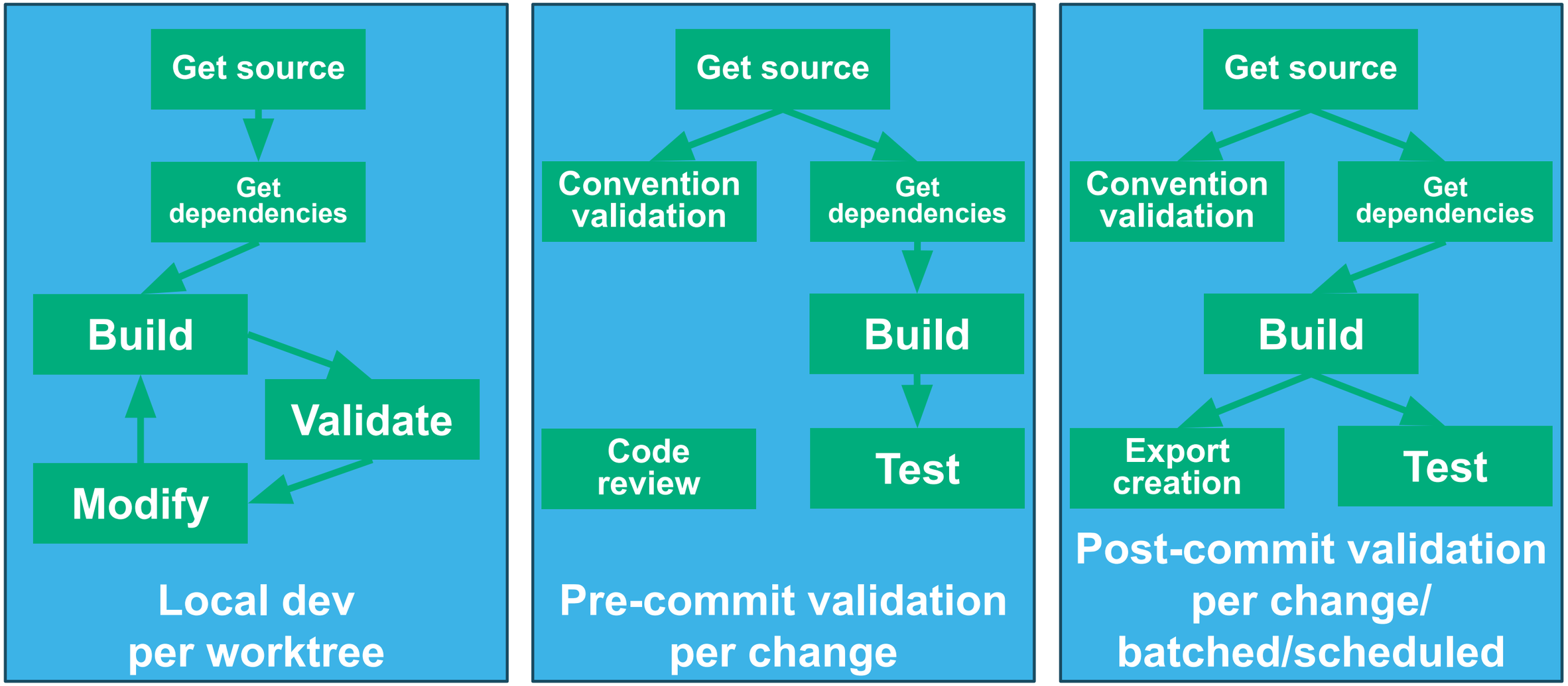
digital



LabVIEW™ NXG

EMERSON

Elements of modern development workflows

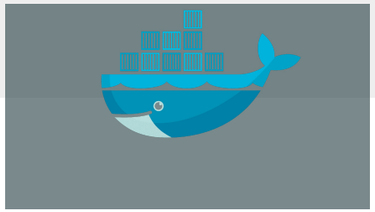


Machine Setup

Dev machine setup | Agent setup

- How much should be in base image vs dynamic?
  - Frequency of version changes
  - Number of versions in active support
  - Install time
- Which OS?
  - LabVIEW cannot cross-build between desktop OSs or bitness

- Tooling approaches
  - Manual/custom scripts
  - JKI Dragon – NIPM (including LV) and VIPM
  - Virtual machines – alleviates install time
  - Containers – lighter weight than VMs
- LabVIEW activation
  - Can be automated
  - CI/CD license



EMERSON | NI

Source Control

Get source

Code review

Backup

History

Release mgt

Code review

- Frequent, small commits
- All source should use “Separate compiled code”
- Set a “Save version” in all projects
- Avoid merging VIs
  - Modularity: isolate concepts
  - Small VIs
  - Communication
- Don’t put built output into SCC

Source Control

Get source

Mainline branching model

Code Review

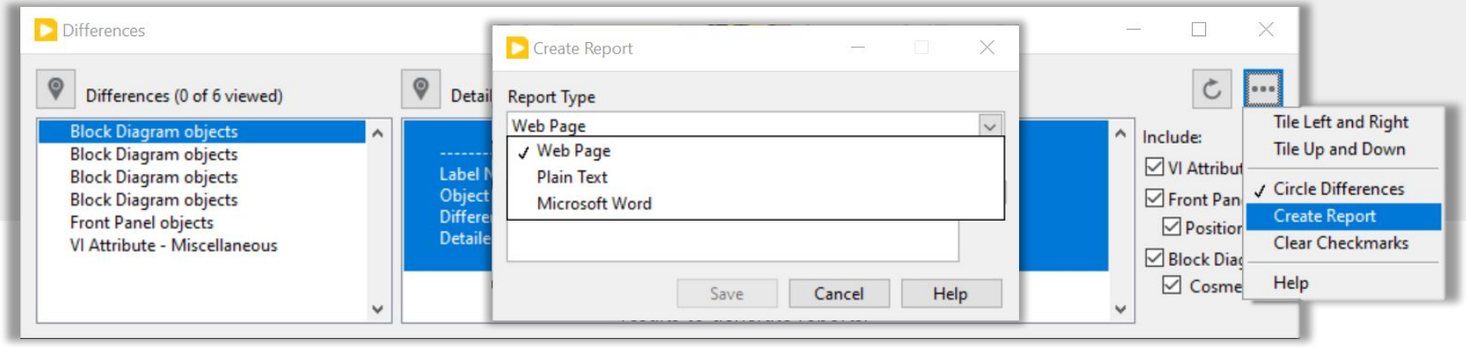
Code review

Multiple approaches

- Interactive diff with author
- Interactive diff without author
- Generated diff report
  - Linked from review tooling
  - Directly in review tooling

Tooling for diff reports

- Manually generate
- Automated
  - **SCCSup Compare Two VIs.vi**
    - [App]User Interaction>>Compare VIs XML
  - Picture generation (Get Image or Print)
  - Post to review tooling



Dependency Management

Export creation

Get dependencies

Files not authored by your team & files built from your team's source

- Dependency **list** is source
  - Dependency content is not source
- Artifact repository
  - NIPM/VIPM
    - JKI Dragon
  - JFrog Artifactory

Dependency Locations

Build | Get dependencies

Avoid cross contamination of dependencies between builds

- Put dependencies in an easily cleaned location

Don’t assume all machines have the same absolute paths

- Load dependencies from symbolic paths

- LVAddons.CustomLocation (2022Q3)
  - Overrides c:\Program Files\NI\LVAddons
- LVAddons.AdditionalLocations (2024Q1)
  - ‘;’ delimited list of paths
  - Overlays the addon root
- “TargetClass”.LibraryPaths (2021SP1)
  - Overlays the LabVIEW directory
  - Name varies by target type
    - LocalHost.LibraryPaths
    - NI.RT.LINUX.PXI.LibraryPaths
    - NI.RT.CDAQ.Linux.LibraryPaths

Build Output Locations

Build | Get dependencies

Instead of DNatt's c:\PPLs, build and install to custom <userlib>

- Additional LVAddon location
  - LVAddons.CustomLocation=**c:\Addons**
  - LVAddons.AdditionalLocations=**c:\Addons**
- Create **c:\Addons**\PPLs\1\lvaddoninfo.json
- Build to **c:\Addons**\PPLs\1\user.lib

- Extend with platform/bitness
  - **c:\Addons**\PPLs\1\Targets\NI\RT\user.lib
    (Starting in LV 2023 Q3)
  - **c:\Addons**\PPLs\1\Targets\win32\user.lib
  - **c:\Addons**\PPLs\1\Targets\win64\user.lib
  - **c:\Addons**\PPLs\1\Targets\linux\user.lib

Build Output Locations (alt)

Build | Get dependencies

"TargetClass".LibraryPaths

- Different paths
  - LocalHost.LibraryPaths=**c:\PPLs\win64**
  - NI.RT.LINUX.PXI.LibraryPaths=**c:\PPLs\RT**
  - NI.RT.CDAQ.Linux.LibraryPaths=**c:\PPLs\RT**
- Single path with target directories
  - **c:\PPLs**\Targets\NI\RT*user.lib*
    (Starting in LV 2023 Q3)
  - **c:\PPLs**\Targets\win32*user.lib*
  - **c:\PPLs**\Targets\win64*user.lib*
  - **c:\PPLs**\Targets\linux*user.lib*

Static Analysis

Convention validation

What to enforce?

- Editor version
- Source only
- Broken VIs
- Connector patterns
- VI complexity

Tooling

- VI Analyzer
  - Runs VI-based rules
  - Broad abilities

- pylabview
  - Parses LV file formats
  - Limited but fast

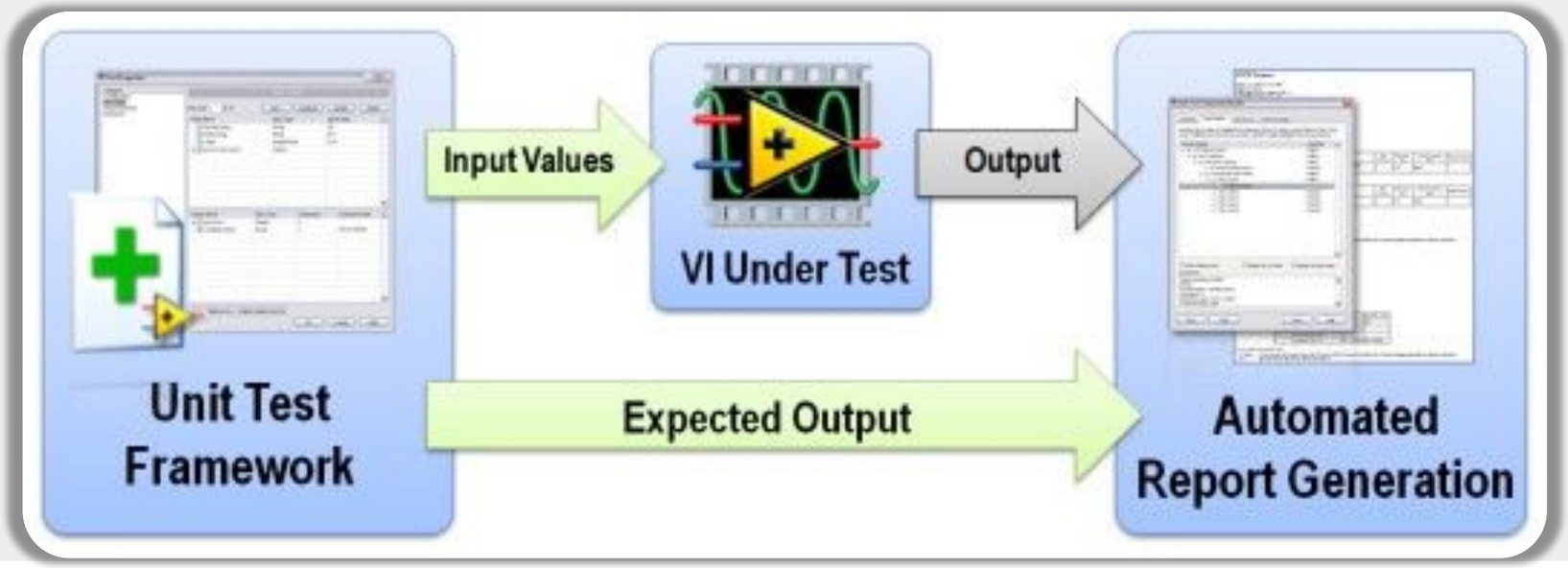
EMERSON | NI

Testing

Test

Avoid building your own framework

- Caraya
- NI Unit Test Framework
- VI Tester
- InstaCoverage
- AST Unit Tester



Automation mechanism

Convention validation | Build | Test

Use a CLI

- Provide access to stderr/stdout
- Can separate operation lifetime from LabVIEW lifetime
- Designed for headless operation

LabVIEW CLI

- Named operations stored in predefined location
  - Class based with per operation help
- Writes stdout at end

G CLI

- VI path provided as argument
  - Loose VI based
- Writes stdout incrementally

Questions or Comments?