FULL FORCE AHEAD
25 YRS OF NIWeek May 20–23, 2019 | Austin, Texas
Taylor Donahue
Project Engineer at Bloomy
Streamline Your LabVIEW Real-Time Development

Standardizing on Design and Testing Strategies

Taylor Donahue
Project Engineer
Taylor.Donahue@Bloomy.com
Bloomy Quick Facts

- NI Platinum Alliance Partner
  - Designated LabVIEW NXG Migration Services Partner in 2018
  - Awarded America’s Partner of the Year in 2017
  - Awarded “Outstanding Technical Resources” 2013 and 2014
  - 15 NI Certified LabVIEW Architects
  - 3 NI Certified LabVIEW Embedded Systems Developers
  - 3 NI Certified TestStand Architects
  - 3 NI Certified Training Centers
  - 9 NI Certified Professional Instructors
- Published “The LabVIEW Style Book” © 2007, Prentice Hall
- ISO 9001:2015 Certified Quality Management System
What is LabVIEW Real-Time?

- Real-Time Operating System (RTOS)
  - Run tasks deterministically
- LabVIEW Real-Time Module
  - Add-on to LabVIEW
  - Run deterministic application on RTOS
  - National Instruments Real-Time Controller
Why Should You Use It?

- Determinism
- Reliability
- Standalone application
- Built in LabVIEW code modules
- NI hardware products
  - cRIO
  - Real-Time PXI
  - Etc.
Real-Time Application

- Project Definition
  - A system that interfaces with a Full Authority Digital Engine Control (FADEC) to upload software and download information from the system

- Development Effort
  - RT application running on cRIO
The Dark Side of LVRT

- Alfred gets assigned this project
- He opens up LabVIEW and starts throwing stuff together
- Selects Scan Engine to monitor analog and digital channels
- Late in development he realizes there is a device in the system that communicates over SPI bus
- He scrambles to switch from Scan Interface over to FPGA
- He realizes there is additional data that needs to be sent between the RT app and the Windows HMI
- He scrambles to update the network layer
- Alfred leaves the company
- Harold gets assigned to complete the application
- It takes him a while to understand what is happening in the application
- He notices a major gap in the FPGA and has to redesign
- Several compiles later
- Harold is being pressured to get the application finished by the deadline
- Changes start to pile up
- ...
- There must be a better way
Complexities of LVRT

- RT applications involve many layers
Streamline the Code Base

- Initial Development
  - Organization
  - Identify design issues early on
  - Ensure the functionality of the code base before release

- Multiple Developers
  - Easy to split up work across several people

- Future Development
  - Easily understand design and add functionality later
Three Phases of a Successful Application

Design

Development

Testing
Three Phases of a Successful Application

- Development Time
- Design & Testing Time
Three Phases of a Successful Application

- Development Time
- Design & Testing Time
Be Proactive

- Be proactive and catch any issues as early on as possible
- Fixing issues reactively can be costly (time and money)
Design
Design

Documentation + Architecture
Documentation

- Before you even open up LV, you should have a clear picture of your design
- Make it very clear between developers what the plan is
- Design reviewed before start of development
- Ensure design covers the requirements
Documentation

- **Design diagrams**
  - Designing the application ahead of time allows you to break up the work across multiple developers (each section of the code is clear)
  - Future developers get a quick idea of what is happening in the code before even looking at it

- **VI documentation**
  - More detailed than design diagrams
  - VI description, free labels, icons, control names – general good style practices
Documentation

- **UML**
  - What kinds of things need to be modeled
    - Communication between system levels (FPGA – RT – Windows)
    - Processes/loops in RT
    - FPGA functionality
  - Use the diagram types you need
    - Structure diagrams
    - Behavioral diagrams
    - Interaction diagrams
Application Example

- Requirements
  - cRIO
  - Data acquisition
  - Logging
  - Run headlessly for weeks at a time
  - DIO
  - AI
  - Relay control
  - SPI bus
Architecture

- “Architecture”
  - Design patterns – solves reoccurring problems in software construction
  - Architectural patterns – fundamental structural organization for software systems
LVRT Golden Rule

Separate deterministic and non-deterministic tasks!
Modularity

- Separate processes accordingly

**Deterministic**
- Closed-loop control
- Decision-making logic
- Safety logic
- FPGA or RIO Scan Interface
- Timing critical tasks

**Nondeterministic**
- File I/O
- Network or serial communications
- Tasks involving large memory allocations
- Calls to nondeterministic libraries or drivers
Architecture

- Hardware abstraction
  - For simulation testing and easy future hardware changes
  - Decide whether FPGA is necessary/useful for your use case
Architecture

- Reuse common architectures
  - Using common architecture across your company
    - Will make it easier for your developers to maintain all applications
    - Debugging becomes less of a mystery
  - Using architectures used commonly across the industry
    - Easier to find help info
    - Bringing in new developers – less time teaching them a new custom architecture
  - Don’t reinvent the wheel every time
Architecture

- LVRT architectures currently used in the industry
  - LV Examples/Templates
  - Distributed Control & Automation Framework (DCAF)
  - LVOOP*
    - Your company’s own custom architecture
- Use the best architecture for your specific application
Doesn’t matter so much **WHAT** architecture you use
It matters **THAT** you use one
(and are consistent about it)
Development
Development

- RT development can be extensive
  - Don’t drag it out longer than it has to be
- Architecture and design documentation are defined ahead of time – makes it much easier to delegate tasks to developers
- Development time significantly reduced! (for both initial and future development)
Testing
Unit Testing

- Individual components of a software are tested
- Validate that each unit of the software performs as designed
- Unit
  - Smallest testable part of any software
  - Usually has one or a few inputs and usually a single output
- Unit tests should not require external hardware to run
Testing

- Initial development
  - Confirms the code base works as you expect
  - Before integration testing in real-time
- Future development
  - Ensures any changes to your code still pass units (don’t break other untouched parts of the code)
Testing

- Test coverage
  - Unit test all public methods
  - Expected error conditions
  - Expected passing conditions

- Hardware simulation
  - Essential for thorough testing of code (units) before integration testing
  - FPGA
  - External hardware

- Unit tests run on entire code base
Testing

- Unit tests run automatically on builds
Testing

- Special considerations in LVRT
Testing

- Examples
  - JKI Caraya
  - LabVIEW Unit Test Framework (UTF)
Application Example
Application Example
Next Steps

- Failed unit tests feed back into development
- Integration testing
  - Individual software units have been validated
  - Test hardware behavior knowing the software works as designed
  - Validate the integrated system
Doesn’t matter so much **WHAT** testing framework you use
It matters **THAT** you use one
(and are consistent about it)
Future Updates

- Anyone should be able to understand the code relatively quickly
- Additions should be minimally invasive
  - Modular code
- Update design documentation
- Update unit tests
  - Add new tests as you go
  - Always test full code base
  - Edit tests only if design requirements change
  - Create additional tests for new functionality
Who’s Happy?

You

Your Team

End User

Future Developer
Summary

- Quality software product
- Faster time to solution
- Less expensive
- Quicker upgrades
Helpful Resources

- cRIO Developer’s Guide
- LabVIEW Help
- LabVIEW Real-Time Project Templates
- Distributed Control & Automation Framework (DCAF)
- JKI Caraya
- UML.org
- Bloomy blog
Before you go, take the survey.
Questions?

Taylor Donahue
taylor.donahue@bloomy.com
(860) 607-2055

Stop by the Bloomy pavilion at Booth #123

Follow our blog at www.bloomy.com/support/blog
Stay Connected During and After NIWeek

ni.com/niweekcommunity
facebook.com/NationalInstruments
twitter.com/niglobal
youtube.com/nationalinstruments