Designing Software Like LEGO® Sets
Stephen Moore – System Engineer - PhD, CLA, CTA
Elite Science and Engineering Services
WE SOLVE YOUR PROBLEMS
even when you're unsure what they are

2 LabVIEW Champions
5 Certified LabVIEW Architects
2 Certified LabVIEW Developers
1 Certified TestStand Architect
What makes LEGOs modular?

- Most pieces can connect with most other pieces
  - Standardized interface or API
- Removing one block doesn’t cause another side of the structure to collapse
  - Loosely coupled
- Each component is relatively simple
  - Ideally does one thing
- Lego designs are reused when possible
  - Share your modules!
- Legos can be acquired in many ways
  - If you can’t share a module, is it still a module??
Tools we can use to make modular code

- Subpanels (UIs)
- Libraries (.lvlib)
- Packed Project Libraries (PPL or .lvlibp)
- Classes
- Actors
  - MGI Monitored Actors
- MGI Panels and Panel Actors (UIs)
- Bonus! Code Distribution
Subpanels

- **Pros**
  - Resizing a lot easier
  - Helps split up the event handler into smaller more modular blocks
  - Each subpanel chunk can be reusable
  - Modular custom UIs are possible by changing which VIs are inserted when needed

- **Cons**
  - Launching and inserting VIs adds complexity
  - Properly stopping all the inserted VIs adds complexity
Libraries

- **Pros**
  - Gives you some control over access scope. Private VIs will not be available outside the Library
    - Helps us make an API
  - Namespacing (VIs with the same name will not conflict if in libraries)
  - Can be replaced by PPLs

- **Cons**
  - Doesn’t help create loosely coupled modules
  - Doesn’t hide the data
  - Doesn’t help simplify the files on disk
  - Can’t be extendible on its own
Packed Project Libraries

- **Pros**
  - All VIs used are included in the single .lvlibp file
  - Linking issues are generally eliminated
  - Works great with TestStand

- **Cons**
  - EXE, DLLs, and other PPL are not included
  - Can’t replace with source library
    - NXG will have this I think
  - Linking to other PPLs can be difficult

- **Pro or Con**
  - Cannot be modified
    - Built code with a specific LabVIEW version
Classes

- **Pros**
  - All the scoping benefits of libraries plus some
  - Extendable through inheritance
  - Can be included in libraries and PPLs
  - Data is encapsulated

- **Cons**
  - Data accessors are annoying sometimes
Actors

- An asynchronous process that contains state information and can send and receive messages

- Pros
  - All the benefits of classes plus more
  - Messaging framework and tools included
  - Makes running multiple threads a cinch
  - Event handling is broken down into smaller chunks

- Cons
  - Getting started is daunting
  - Debugging can be harder
  - Lots of simple VIs and message classes are needed (most are scripted for you)
  - Launcher splash screen is usually needed.
MGI Monitored Actors

- A child class of an Actor with great debugging features
- Pros
  - **Free**
  - All the benefits of actors
  - In development a window shows all running actors
    - Running actors' block diagrams can be opened
    - Actors can be stopped
    - Message Logging
- Cons
  - Doesn’t install by default with LabVIEW
MGI Panels / Panel Actors

- Simplifies creation of flexible user interfaces
- Supports launching, inserting, removing, and stopping VIs within subpanels or dialogs

**Pros**
- **Free**
  - Can be used with Actors, DQMH, or standalone
  - Has all the Pros of ALL the previous methods (Actors, Subpanels, Classes, etc)
- Can be extended with your own panel types

**Cons**
- Passing data between panels requires messages to be made
Monitor and Panel Actors Demo
Tips for using Actors

- First use Monitored Actor for everything
  - Note: Panel Actor is a monitored actor too
- If the actor has a UI element then it should be a Panel Actor
- Override “Handle Error.vi”
  - if you don’t your actor will stop when you get any errors with no notification
- Don’t try and launch nested actors in Pre Launch Init
- Messages need to be fast
  - If you have something that takes a while, off load it to a helper loop
- You’ll end up spending more time in the project then in massive VIs
Module/Package Distribution
Tools for Code Distribution

- Copy code manually 😞 😞 😞 😞 😞
- VIPM
  - Installs at a system level 😞
  - No record of what packages are used in a project
    - Makes it hard for multiple people to work the same project (.VIPC files required)
  - Hard to contribute to an open source package
  - NI controls submissions to the package list
- NIPM
  - Installs at a system level 😞
  - Closed source, maintained by National Instruments
  - Allows for custom feeds
  - Great for binary distributions but not for code
The new G Package Manager

- An Open Source tool to build and consume LabVIEW packages
- Installs packages with project scope
- Keeps information about package dependencies
- Easily restore package configuration
- Inspired by the Node Package Manager from the JavaScript community
- Installs via NIPM
- Works along side VIPM
Finding Packages

- Navigate to https://gpackage.io
- Search for a package (try @mgi for our packages)
- Read about it
- Copy the install command from the top right
- Use the install command to install the package in a project
Getting Packages Demo

Using the Browser
1. Create a new LabVIEW Project
2. Open GPM Browser
3. Find Relevant Package
4. Read about it
5. Install it
6. Import it into my LabVIEW project

Using the Command line
1. Launch the command line with the proper working directory
2. Use the install command to install a new package
3. Delete all installed package files
4. Use the install command to restore required packages
We need help!

- Submit more packages!
- This is beta software
- The repo’s issue tracker is populated with desired features and bugs
- **Submit your merge requests NOW!**
- [https://gitlab.com/mgi/gpm/](https://gitlab.com/mgi/gpm/)
Other useful information
Is the code modular?

- Don’t install stuff in instr.lib, vi.lib, or user.lib!
- Is there a manifest file describing dependencies
  - NO
- Does it use classes?
  - NO
- Does it use access scope?
  - NO
- Is the UI modular?
  - NO
The End…
Additional Resources

- More information on Sharing Code
  - Sharing LabVIEW code: What Tools Are Good For What?
  - Wed. 22 - 10:30AM - Room 12B

- https://mooregoodideas.com/categories/actor-framework/
- https://mooregoodideas.com/mgi-library/panel-manager/panel-actors/
Stay Connected During and After NIWeek

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