We help ideas meet the real world

Living and Learning with LabVIEW

Carsten Thomsen cth@delta.dk

Director of Engineering

DELTA

www.delta.dk

LabVIEW is . . .

We help ideas meet the real world

- **Easy** to get started with . . .
  - Easy to get in trouble with
- **Huge**
  - But you only need to know a small sub-set
- **Simple**
  - In understanding the basic paradigm
- **Complex**
  - in understanding the detailed execution model

Our Experiences with LabVIEW: Our background

We help ideas meet the real world

- Small to Medium size projects
  - 1 week to 2 man-months
- A few large ”team projects”
- Very few ”native” LabVIEW programmers
- Combination of internal/external end-user projects.

Lessons Learned

We help ideas meet the real world

1. Learn LabVIEW/Take the courses!
2. Read and use ”LabVIEW Style Guide” by Peter Blume.
3. Learn ”the LabVIEW way” of doing things. Don’t force text based language/object oriented concepts upon LabVIEW.
4. Use LabVIEW Design Patterns from day one.
5. User LabVIEW at simplest level possible for a given project size.
6. Readability is vital! (can you understand your code after two months?)
7. Clarity in naming conventions, structure, design, documentation.
8. Throw away your prototype. (unless you are *very good).*
9. People who can pass CLD are quite good. Not that easy an exam.

Some Examples

We help ideas meet the real world

- A practical rapid prototype program thrown together in about three weeks . . . (shown in Randers/Rungsted in fall of 2008)
- Evolution of prototype to production software (x10)
- You will see both the good, bad, and the ugly.

Design Patterns

We help ideas meet the real world

- Everyone *must* understand
  - TypeDefs
  - LabVIEW State Machines
  - Queued State Machines with Event Handling
- Without these, you *will* fail or suffer.
- You will write code that is
  - Unreadable
  - Unmaintainable
  - Non-scaleable.

Design Patterns...what I've learned so far

We help ideas meet the real world

- User Event, and based on it, you want to do something (Actions). (You should most likely lock front panel when calling queues)
- You make a list of Actions, and stuff them into a Queue.
- A parallel loops executes the Actions in order, ending with an Idle case.
- Data passed to actions: (Think about locking when you design)
  - Shift Reg
  - Functional Globals
  - Properties.
  - Queues
- More Sophisticated: Add parameters with your Actions. Often these are done as Variant. I'm not ready for this yet.

Design Patterns (cont.)

- The Event Driven State Machine (Blume p. 262) is
  - Extremely powerful
  - Compact (the ideal one screen diagram)
  - Easy to navigate without ever scrolling the screen.
  - Expands graciously to multiple parallel loops.
- Piping data in from parallel loops to multiple graphs (via queues).

Demo Time: LabVIEW Pasta

We help ideas meet the real world

- Queued State Machine
- Initialization (Huge)/comments on persistence
- Master TypeDef
- Master References Type
- Master Queue Cluster.

Design Patterns (concluding remarks)

- I have not used dynamic frameworks.
  - We have enough memory so it is not an issue.
  - Are harder to build, de-bug, and understand.
  - They are useful to create expandable systems.
  - Would only recommend for larger projects.

Moving Data Around in LabVIEW . . .

We help ideas meet the real world

- The classic way: Shift Registers
  - Efficient, visible
  - Maintain in carefully thought out TypeDefs.
- Globals: I've never used them. Not safe.
- Locals: OK to initialize, and for very local use.
- Property nodes: Less efficient but very readable. Have error terminal. Efficiency not an issue for simple user-interface controls
- References: Permits embedding in VI's. Inefficient, harder to de-bug.
- Queus: Efficient and safe, but not that readable.
- Functional Globals: Efficient, quite safe. Not as readable.

Advantages of Functional Globals

We help ideas meet the real world

- Data can be passed without wires, to any depth of Sub-VI.
- Data can be passed without wires to parallel loops.
- Great way of storing persistent data:
  - Just save (to disk) the Functional Global on shut-down, re-load at start-up.
- Functional Globals are protected against race conditions:
  - They are non-re-entrant
- Danger: They die if calling VI leaves memory, i.e. Watch out with dynamically loaded VI's.
- See: http://wiki.lavag.org/Functional_Globals

Functional Globals: Wireless LabVIEW

We help ideas meet the real world

- Demo
- Now using feedback nodes.

Data Structures in LabVIEW

We help ideas meet the real world

- Always use carefully structured TypeDefs (macaroni)
  - Everything must be named, don't use arrays except for things that really are arrays.
- No more than two levels in hierarchy: Make code complex to bundle/unbundle.
- Typical Groups:
  - Front Panel Controls
  - Internal parameters including File locations, states
  - Queues
  - References
  - Results

More Data Structures (2)

- Master Monster Cluster?
- Cluster of References instead/also?
  - Is more general (giving access to all properties, not just values)
  - Less Readable. Requires two property nodes per access.
  - Embeddable in Sub-VIs. (Harder to find)
  - More difficult to maintain, update.
- Keep it as simple as possible. Larger programs require more advanced mechanisms, but they will hurt you on small/quick projects.

More Data Structures (3)

We help ideas meet the real world

- Front Panel Display of Data Structures?
  - Shrink them, and then expand to edit
  - Declare in Sub-VI, route via wire or Functional Global.

Demo: Macaroni (Clusters)

We help ideas meet the real world

- (put the spaghetti inside the macaroni ;-))
- Type Def
- Cluster of References
- Functional Globals
- Queues.

Complex Front Panels in LabVIEW

We help ideas meet the real world

- Many different techniques (High Level)
  - Lots of pop-ups/wizards (confusing world view)
  - Dynamically loaded VI's/Sub-panels (hard to maintain/de-bug)
  - Static Layout/Few Main Modes (My Preference)
    - Adobe Lightroom is a very advanced application with this world view.
- Color-cues of selected tab sheets (Demo)

Multi-pane graph Manager

We help ideas meet the real world

- Explored splitter bars
- Created own graph manager
  - Four quadrants:
    - Each graph can be quarter/wide/full size.
    - Graphs live on transparent tab sheets.
    - Have local fields on tab, global fields above tab.
- Lesson Learning ”Scale Controls with Pane”
  - Text boxes vs. Numerics.
  - Type Defs vs. Not Type defs.

Demo: Graph Manager

We help ideas meet the real world

- Four Quadrant
- Quarter/Wide/Full States
- Demo of Parallelism and Locking

Graphs and audio

We help ideas meet the real world

- LabVIEW Graphs:
  - Cannot not set graph object size programmatically.
  - Cannot Color object.
  - Timing issues with cursor read/write.
- Audio
  - Suitable for very simple applications
  - Buffer size is tricky. Multiple issues for more advanced applications:
    - Go native, for example with ASIO drivers.

Keyboard shut-cuts

We help ideas meet the real world

- Single character shortcuts vs. CTRL Char
- Key-focus issues.

Demo: Keyboard Shortcuts (really shitty code)

- You can use F keys, assign via right click menu.
- Or you get keys directly.
- Give very heavily loaded nested cases.
- De-bug farm.

Ocean of Property Nodes vs. Front Panel Managers

We help ideas meet the real world

- Good CS talks about information hiding/encapsulation.
  - I agree for large projects, but believe less hiding is increases efficiency on small projects.
  - You must be wise in your choices. When you feel pain, then you have pushed the paradigm too far.
- Ocean of Property Nodes
  - Has all Property Nodes at Top Level VI.
  - They're easy to find
  - Easy to Read
  - Easy to Expand
  - Not considered good programming practice . . .?
  - You must carefully walk all property nodes to understand what is going on.

Bad Programming Practice...Everything at top level?

- With Event Driver State machines you can have **huge** amounts of code at the top level
  - Program can still be well structured.
  - You avoid the encapsulation/data hiding of sub-VI's:
    - Can be both Good and Bad.
  - The code can be extremely readable.
  - The code *can* be easy to maintain (depends on your self-discipline).
  - The code will eat more memory.
  - The approach will break down as code size grows. May not build . . .

More Bad Programming

We help ideas meet the real world

- But, you can graciously start creating Sub-VI's.
- All this assumes that your data structures are clean and well thought out
- That your state machines/execution model is well done.
- I believe this is a good way to have extensible code.
  - It is extremely fast to write
  - If done skillfully, scales very well.

Persistence

We help ideas meet the real world

- Users expect it
- Is fairly hard to do well.
  - Cannot be done as simple snapshot of front panel and restoring it.
  - Sequence/states at shutdown must be included.
  - Fairly sophisticated start-up that walks all relationships is required.
- Do it for the most important things:
  - File Paths (individual for each type of file)
  - Main ”User Project” settings, especially if he can lose a lot of work.

The Art of Programming

We help ideas meet the real world

- Is having the wisdom to apply the right design pattern (architecture)
- The biggest enemy of software quality today it ”time-to-market” pressure.
- The successful guys wisely select the right paradigm for the task at hand.
- With good understanding of LabVIEW data structures and design patterns you can be extremely efficient.
- My rule number 1: Use mechanisms you really understand.
- Test new mechanisms on side projects. Use the Sandbox!
- Don’t get overly ambitious in terms of elegance.
- Be humble.

Conclusion

We help ideas meet the real world

- Keep it Simple!
- Good Code Looks Simple.

A beginners Journey: Things I found difficult

We help ideas meet the real world

- Arrays: Visualizing the dimensions/structure
- TypeDef: Why didn't I hear about it before?
- Run-time menus. Two Types. You must manage states yourself.
- How to move data around
  - Data caught in the loop
- Really understanding the execution model.
- The more advanced LabVIEW programs become, the less data flow!
- Choosing the right execution model. (Ask the experts)
- Wiring References into Sub-VI's

A beginners Journey: Things I found difficult (2)

We help ideas meet the real world

- Building Clusters of References
- Maintaining Clusters of References (dangerous to delete)
- Orphan Controls spawned from TypeDefs are not updated.
- Connecting Bundle by name to Cluster input of VI's.
- Automatic Sizing of diagram makes structures bigger and bigger!
- Understanding the Project Manager: Overview and many details that are not easy to understand.
- Become a wiring expert.

Practical Little Hints

We help ideas meet the real world

- Learn one new shortcut/trick per week and share it! Look over your colleagues shoulders. There are hundreds of smart things in LabVIEW.
- Use lasso when moving constants. Dangerous to select!
- Be careful when examining bundle/unbundle. Safer to CTRL H or expand bundle to see what's in it.
- Re-order unbundle terminals to prevent crossed wires.
- Don't use CTRL B for large programs. You may lose wires that are not mandatory and will have a hard time de-bugging
- Think carefully about the order in which you do things. Wire broken wires, then delete the "sinner".
- Have a Yellow master Comment and Wire Label in Each VI.
- Right Click "intelligent menu" Insert Index Array/Build Array.

More Tips and Tricks

We help ideas meet the real world

- Keep diagram to one screen width at top level. Parallel loops may extend vertically.
- Data algorithm Sub-vi's may be wider, can actually be more readable.
- Have farms/pastures outside main diagram: Un-used/temporary controls, internal variables/property nodes, de-bug help.
- Show a queue status, on front panel during development.
- For simple keyboard shortcuts, use the native Keyboard Shortcut. Limited/Slow but easy to assign.

More Tips and Tricks (3)

We help ideas meet the real world

- Always start up-stream when working with Case/Event Structures.
  - Make TypeDef
  - Create Common Code (wires/shift reg. /error cluster.
- Make Case for Every value may seems clever . . .BUT Duplicate Case is often smarter!!!
- Feed Queues with Array of Commands. Disadvantage: LabVIEW shrinks width when you update typedef.
- Default values: Put inside Case, put property node outside. Makes code more readable.

More Tips and Tricks (4)

We help ideas meet the real world

- CTRL Mouse Wheel to Scroll in Case/Event Structure.
- Every dialog must also handle Cancel Case gracefully.
- Initialize ALL front panel controls programmatically. Is much safer. Enables persistence. Don't rely on initialization via "Current Values as Default" for large projects.
- Editing of Tab Controls: Everything can fall apart.
- Native LabVIEW Controls are more flexible than System Controls.
- All internal time in UTC.
- All Internal Indexes must be Zero Based. Convert to/from front panel.

More Tips and Tricks (5)

We help ideas meet the real world

- Carefully examine all Shift Register for initialization
- Danger: Multiple Auto-index Tunnels on input of loop!!! Who wins, what if data array is incomplete.
- Divide by zero is NOT an error in LabVIEW. You must explicity trap it!
- Mixing ValSgn with Queued commands . . . Cheap and nasty . . .
- Comb the diagram along the way.
- Before you delete a control, check to see if it has references and property nodes!!!
- Beware of generic property nodes in sub-vis.....

More Tips and Tricks (6)

We help ideas meet the real world

- Keep a project notebook (paper works great!)
- Back-up EVERY day! (Build zip)
- Zip before re-designing large Type-defs.
- Only have one instance of code unzipped
- Learn how the project manager works (it appears to be designed by non-LabVIEW people . . .;-)
- Error handling: For all DAQ I/O and file I/O as minimum.
- Graph controls: Be careful playing with their properties, they may remember things you can't undo. (Go to a sandbox)

More Tips and Tricks (7)

We help ideas meet the real world

- Data Structure/Naming conventions:
  - Noun (Adjective) Verb (adjective) structure
  - Carefully thought out/maintained.
- Sandbox
- Pay me now or pay me later.
- Self-discipline.
- Comments, also self-critical, also what you don't understand, haven't fixed.
- Don't get too ambitious
- References are challenging.
- Learn to use LabVIEW Search. Quite powerful.
- VI Analyzer can be overly helpful, but excellent with filtering.

Good Resources

We help ideas meet the real world

- NI's LabVIEW Courses
- Darren's nuggets.
- NI LabVIEW Discussion Groups: these are really getting good.
- NI Support: Look in the mirrow before you call.
- Google: learn to write good search expressions.
- http://forums.lavag.org/blog/champions/index.php?showentry=111
- Use your colleagues. Beg, borrow, steal!