Structured Software Design Consultants

Steve Watts

SSDC Ltd

Lean LabVIEW

swatts@ssdc.co.uk

@swatzyssdc

Structured Software Design Consultants

NATIONAL INSTRUMENTS Silver Alliance Partner

18 Years

CSLUG

GDEV CON #1
Cambridge UK 4-5th September 2018

eCLA 2012 LabVIEW Smells
NIDays 2013 How to Polish Your Software and Development Process to Wow Your End Users
eCLA 2014 Process Smells
Various 2015 Immediacy
NIWeek 2016 TS9456 - ISO 9000 and LabVIEW
NIWeek 2016 PS9044-Shock Testing using Multiple Synchronised Racks
ECLA 2017 Risk and Mitigation
NIWeek 2017 The SSDC Way: Desire Paths to a Simple Software Process
ECLA 2018 Modular Design Deep Dive

Read my blog on ni.com – Random Ramblings on LabVIEW Design

Goal:

Show techniques that allow SSDC to produce large complex projects while keeping them responsive and light.

i.e. how to play nice with LabVIEW

What is Lean LabVIEW and Why?

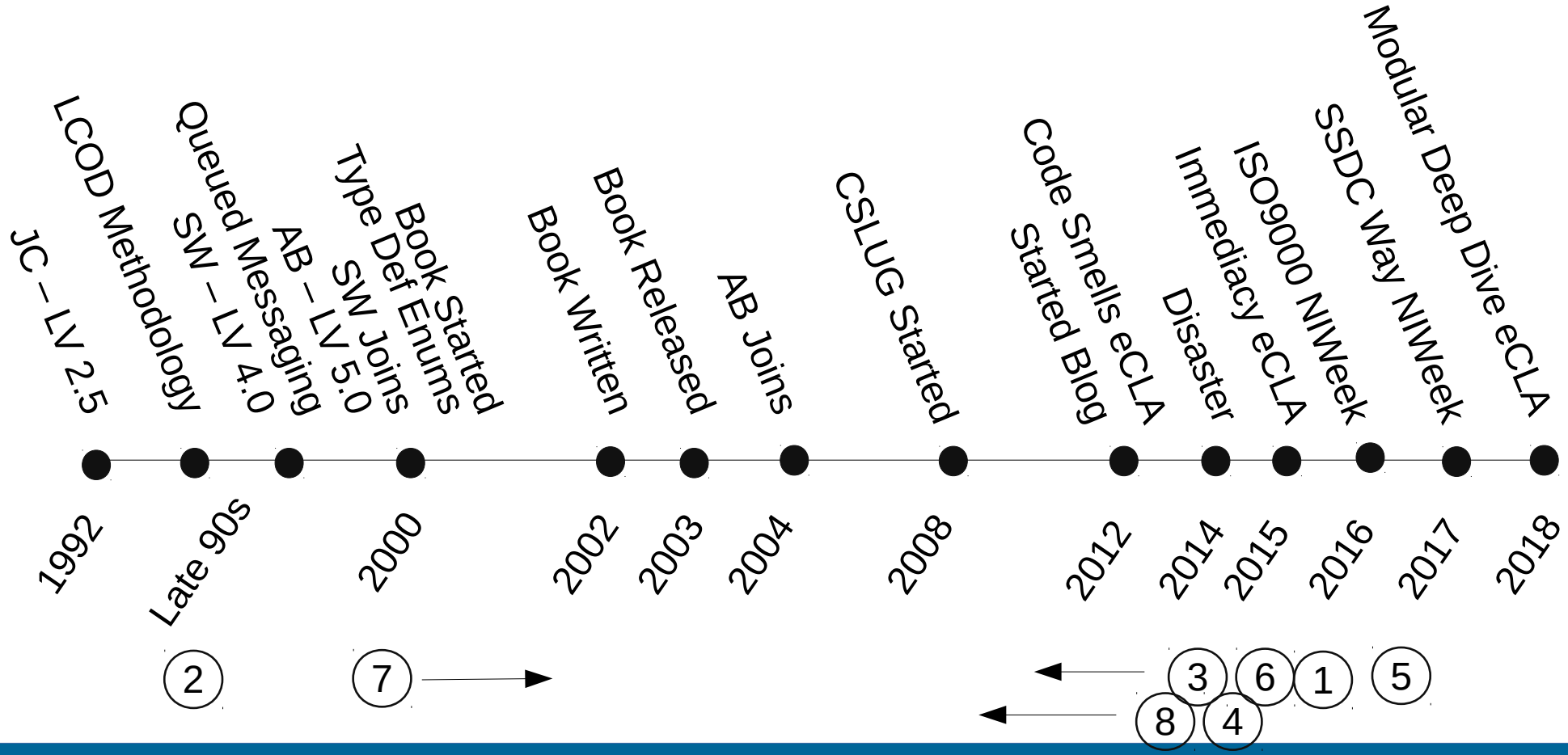
Key Aspects

1) A Tidy Project is a Happy Project
2) Cross Linking
3) Project Portability
4) View Dependencies with suspicion
5) Polymorphism – Bloat Alert
6) Immediacy – Glue Vs Do
7) Lean, Clean Code
8) Lean Error Handling

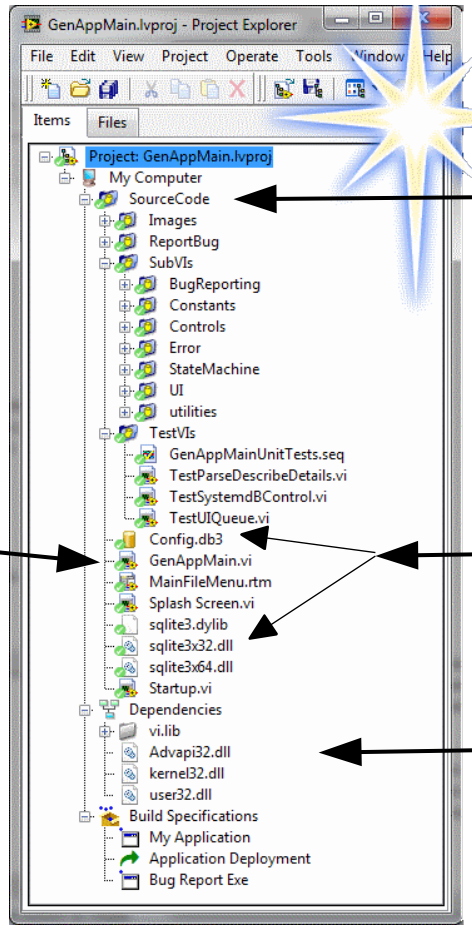
Structured Software Design Consultants

The importance of time

These techniques have been used by SSDC for many years and are proven by our many projects in many industries.



A Tidy Project is a Happy Project



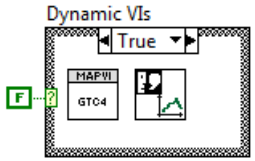
Auto-populating folders vision of change is important

Main Calling VI sits atop the hierarchy

dlls, config db etc are all at top level

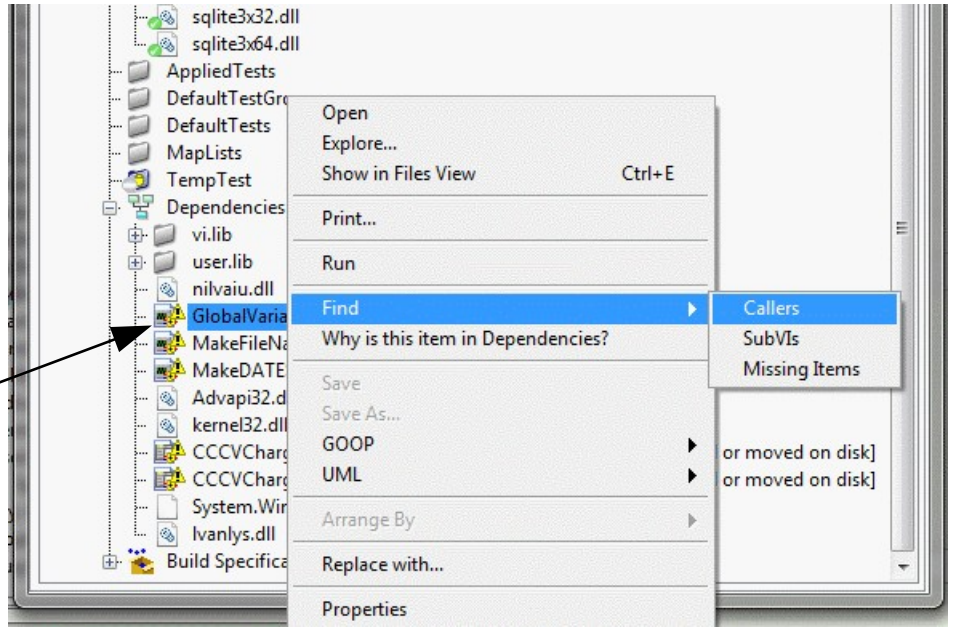
Uncompromising about dependencies

A Tidy Project is a Happy Project



Get LabVIEW to check your dynamic vis

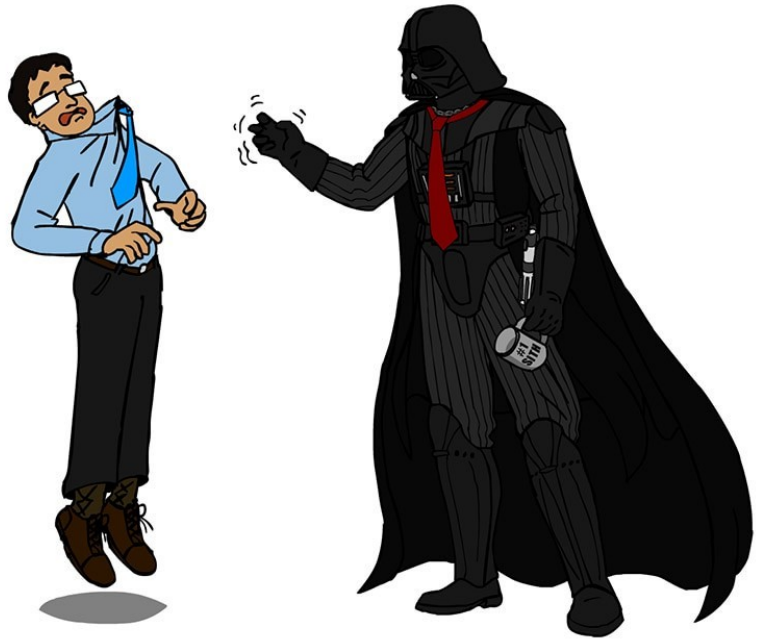
Eliminate these



Cross-linking

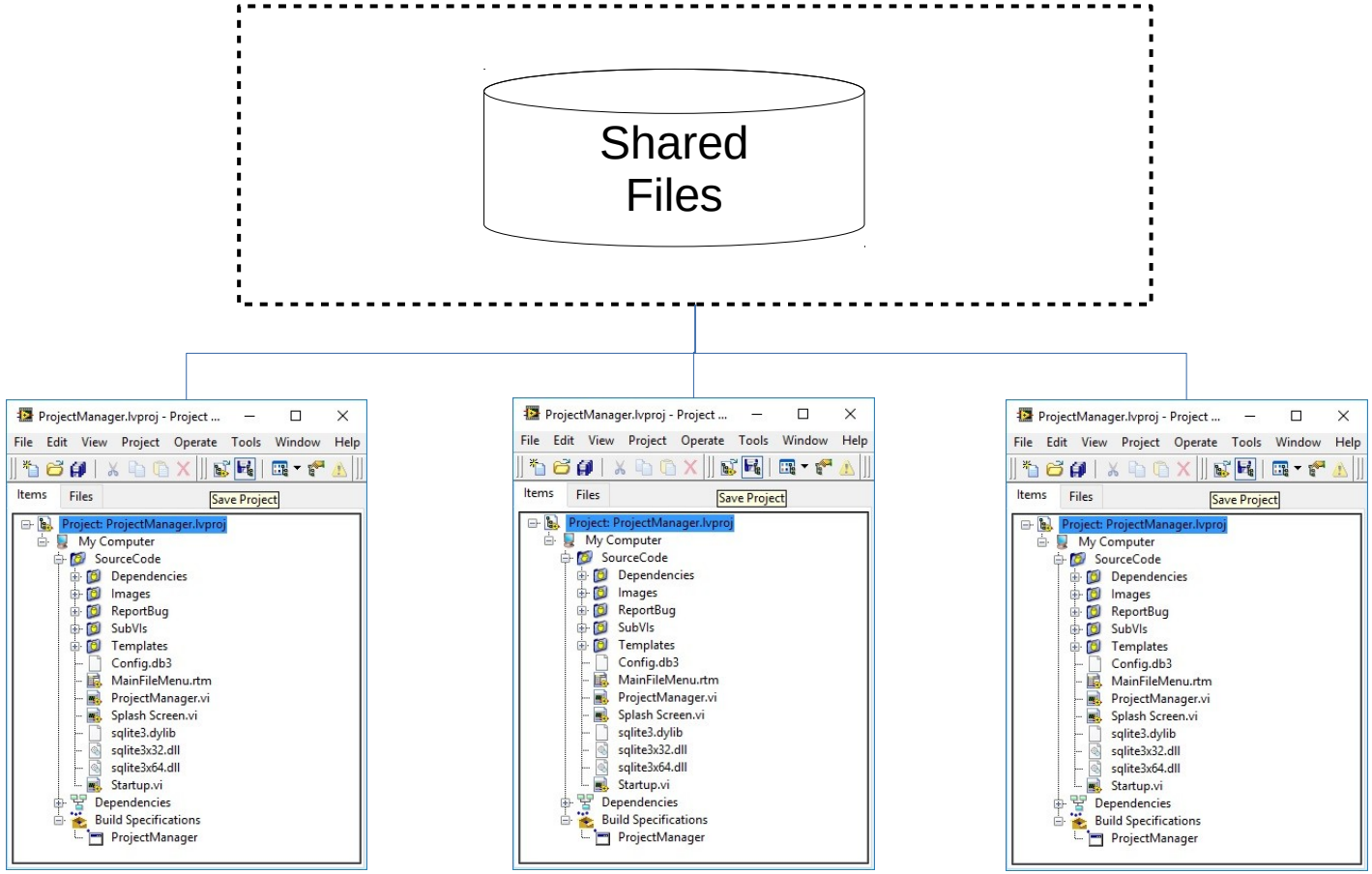
Vic Stennings Principles of Project Hygiene (1979)

Principle 5
Changes should be controlled, visible and of known scope.



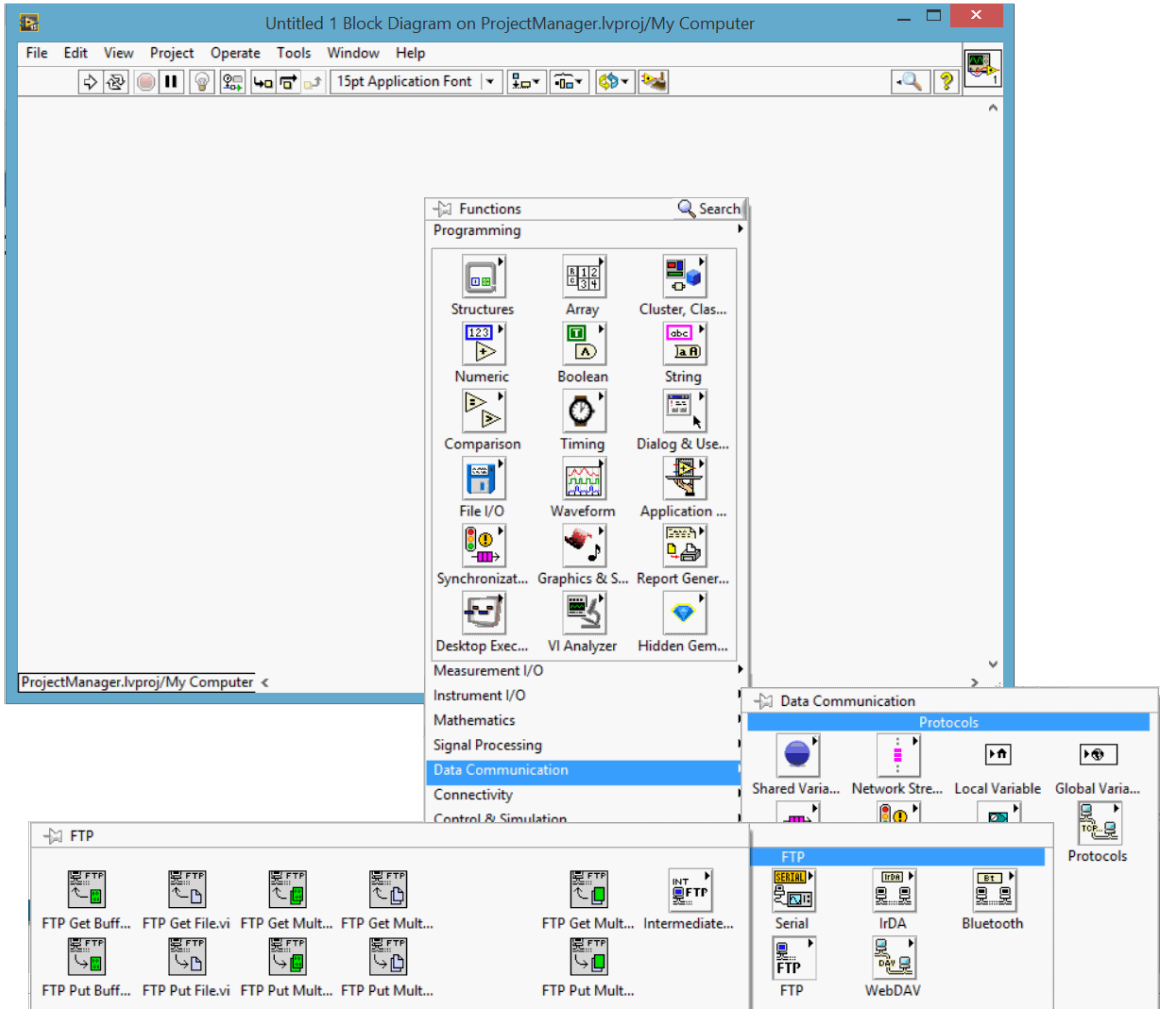
The enemy for predictable software is silent changes

Cross-linking



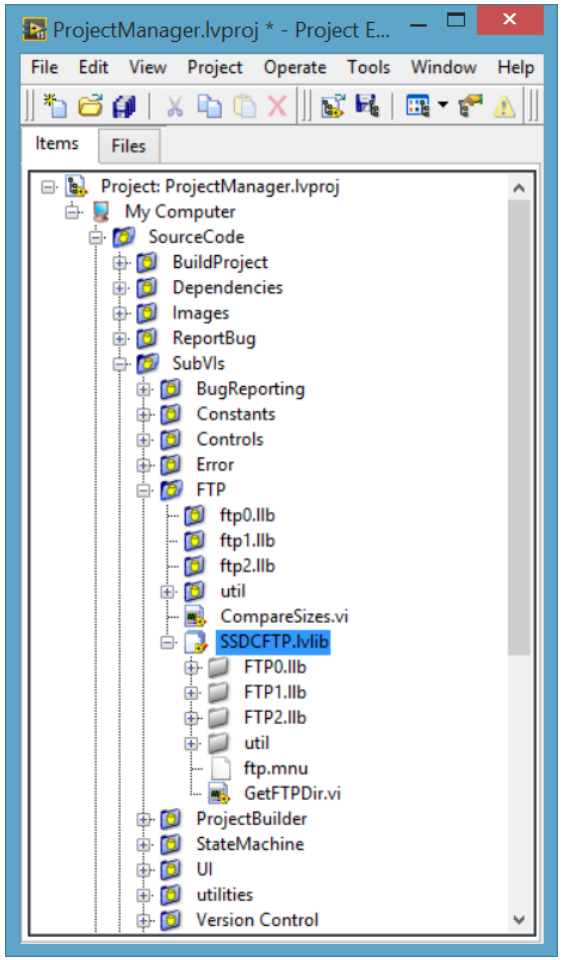
Silent changes

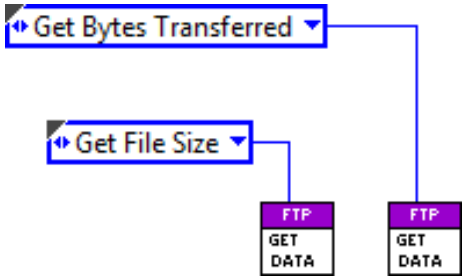
Cross-linking – Environmental changes

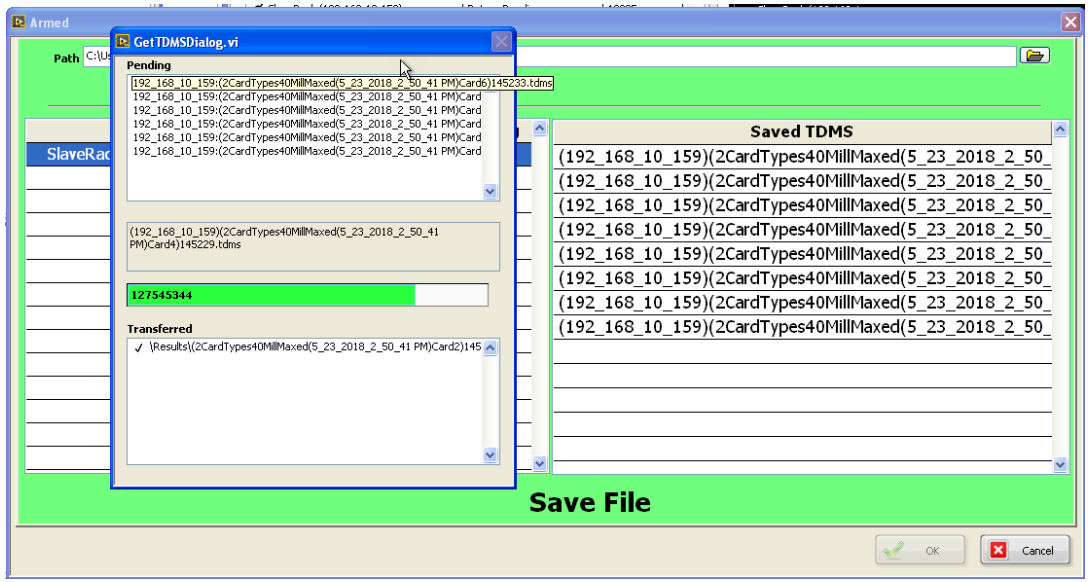


FTP Example

Cross-linking – Environmental changes

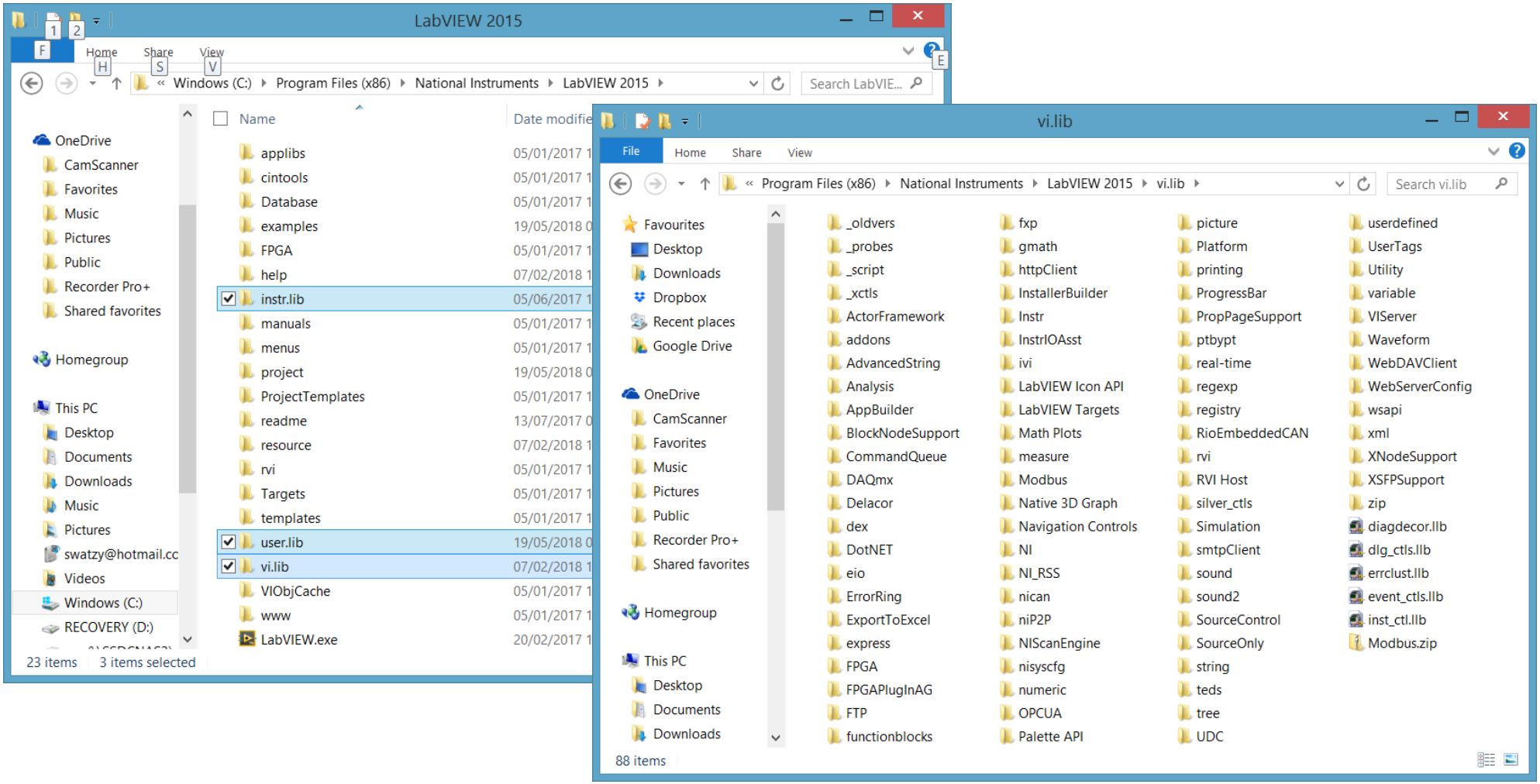






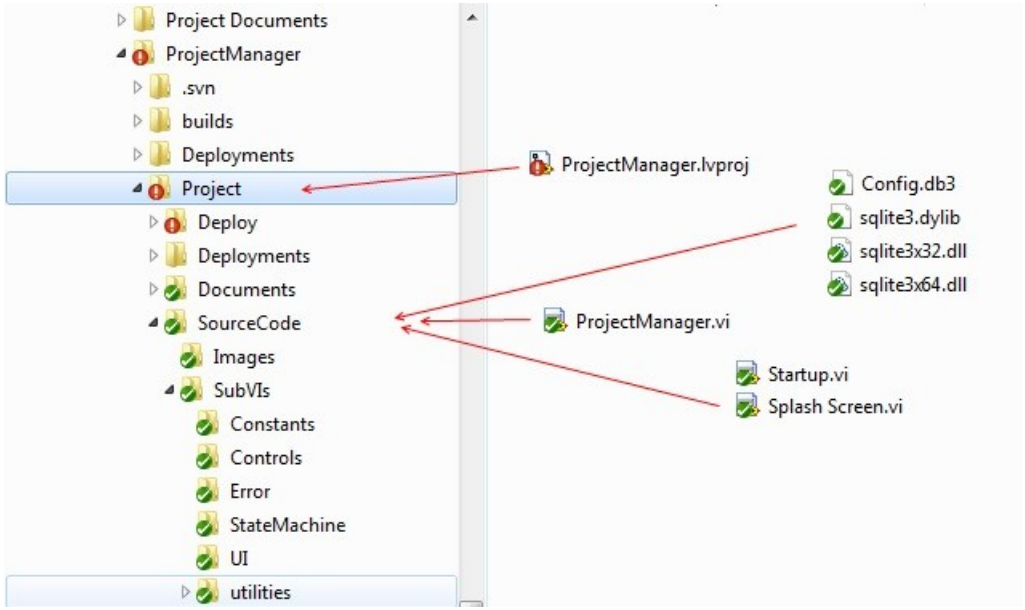
FTP Example

Cross-linking – Environmental changes



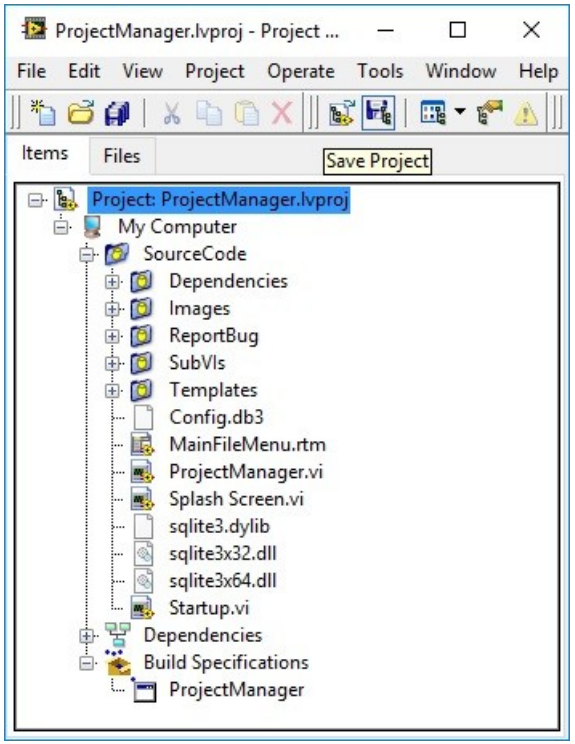
These all have VIs you can edit

Project Portability



Structured Software Design Consultants

Project Portability



The test for portability is one of our code review items and involves exporting the repository into a clean virtual machine (with LabVIEW Loaded) and into different directories (usually desktop and \User\Public\Something or other.

Structured Software Design Consultants

SSDC Limited	
Description	Code Review Check List
Issue	1:02
Date	Tuesday, 06 May 2014
Document Number	2013-175WI0004(Code Review Check List)

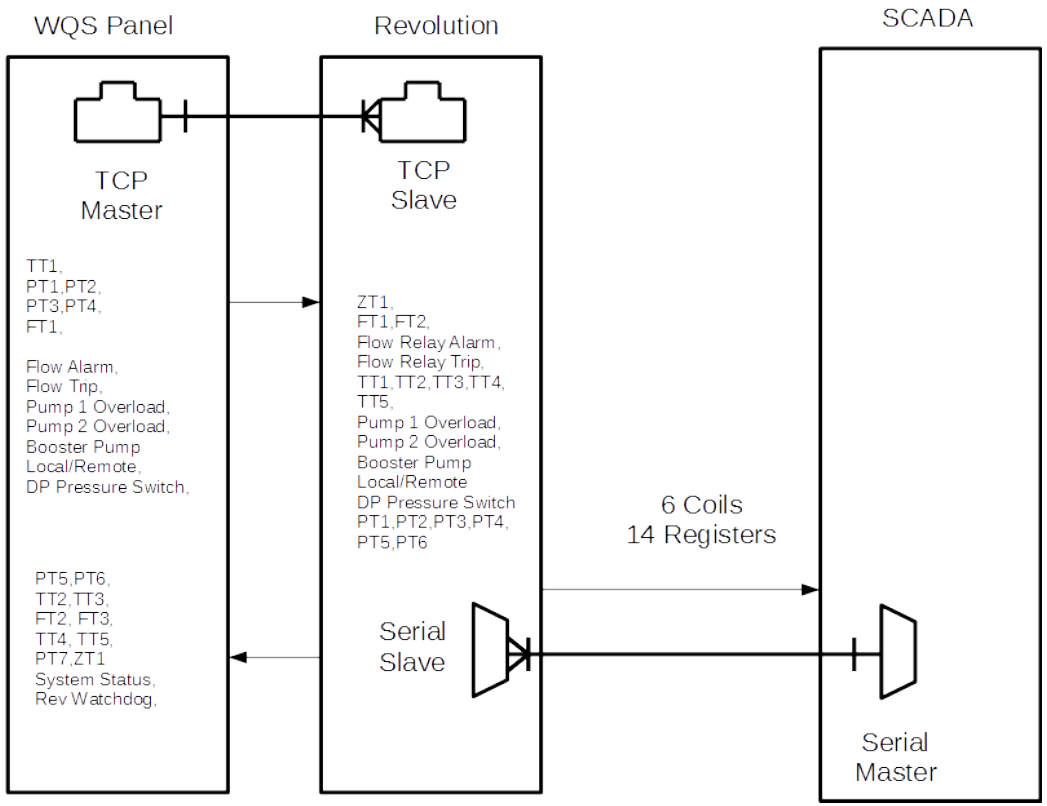
Initial Checks

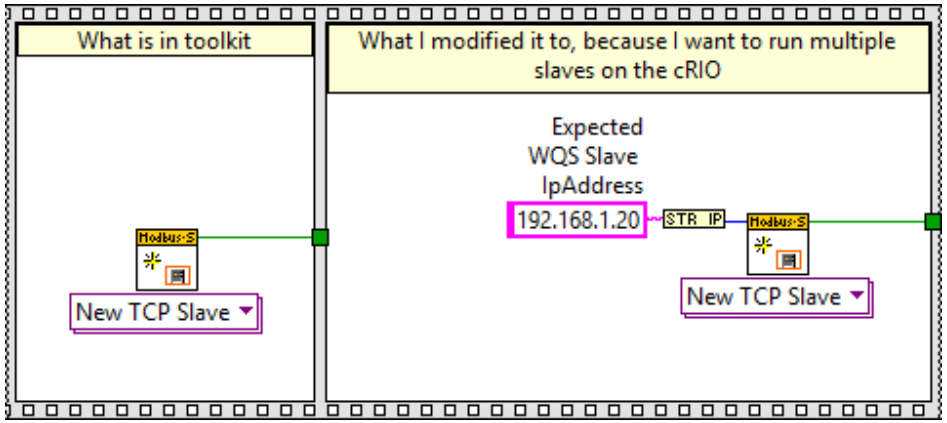
Check out from repository prior to starting. Ideally onto a clean machine or virtual machine

Details

This will test that the repository is set up correctly and the project can run, any dependencies and linked libraries, hardware libraries will need noting and their version documented.

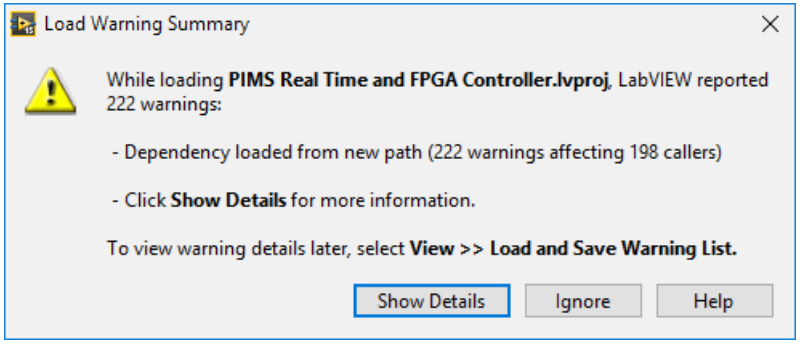
Project Portability
Case Study - Step 1 Move Directory from vi.lib





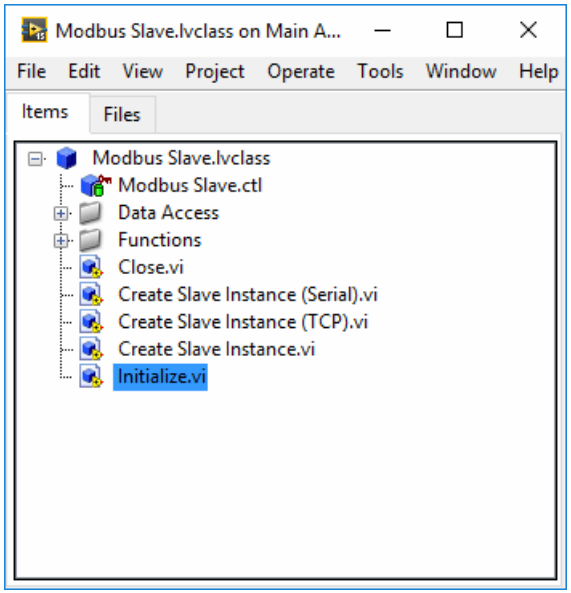
Why we wanted to modify the library

Project Portability
Case Study -Step 2 Loading Project With Embedded Library



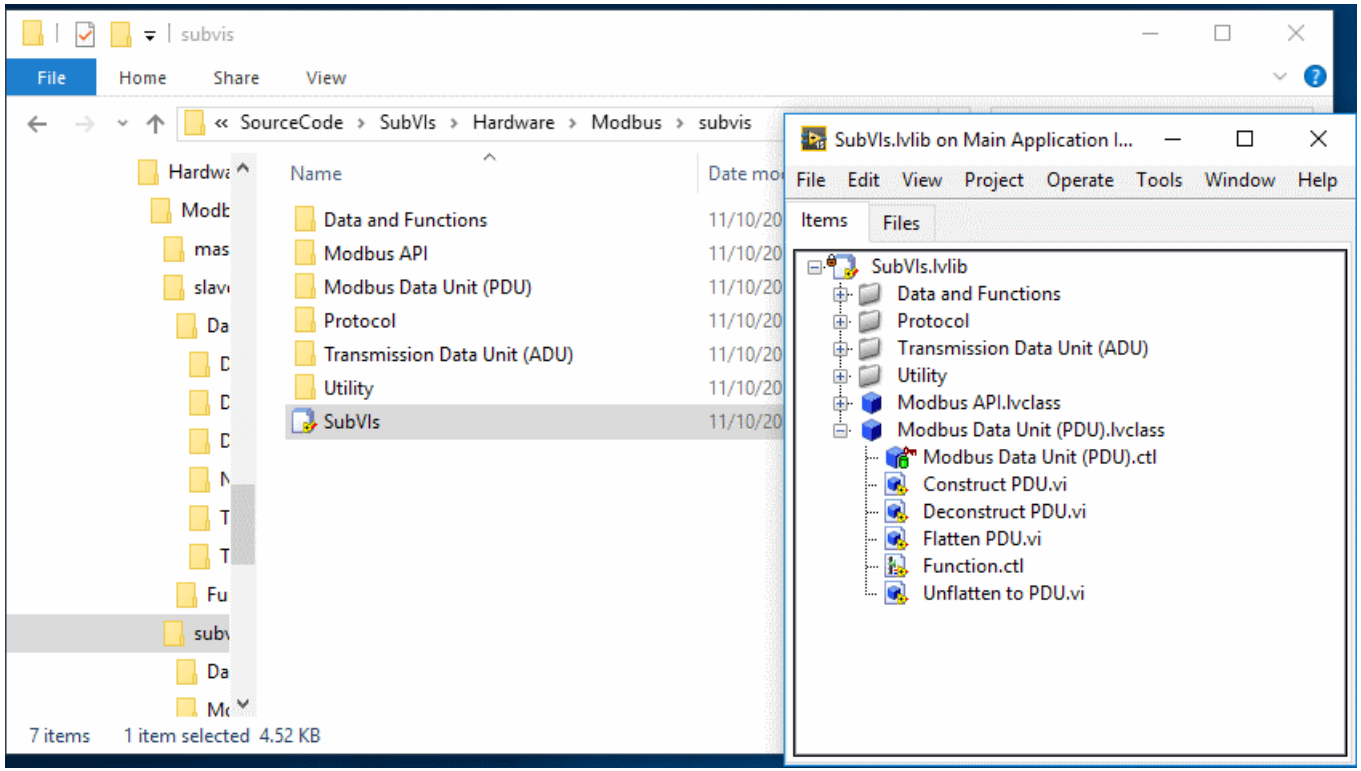
Warnings!

Project Portability
Case Study - Step 3 Tackle Slave Class On Its Own



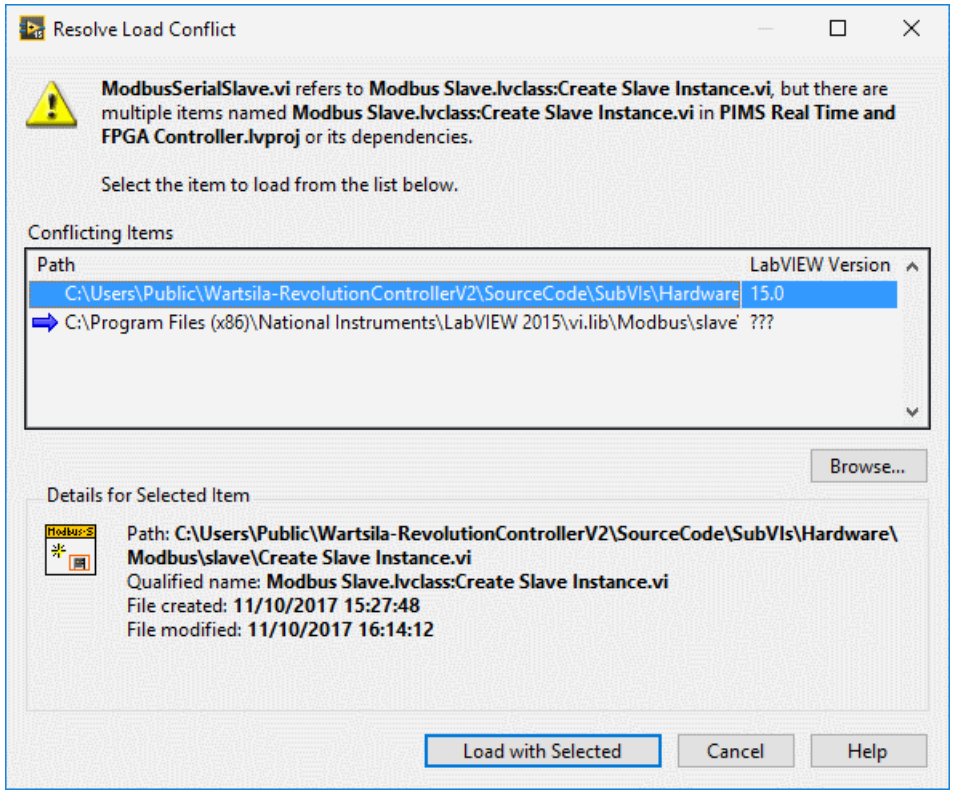
Only use what you need – this is project work, not API

Project Portability
Case Study - Step 4 Saving SubVIs.lvlib



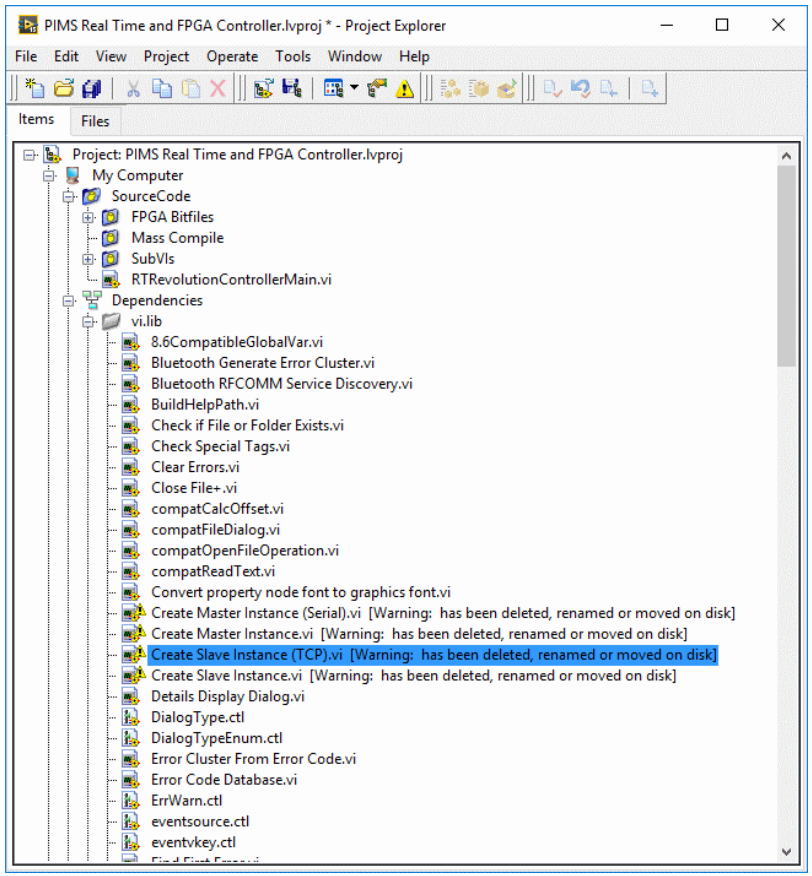
LabVIEW likes file hierarchy

Project Portability
Case Study - Step 5 Back In The Project



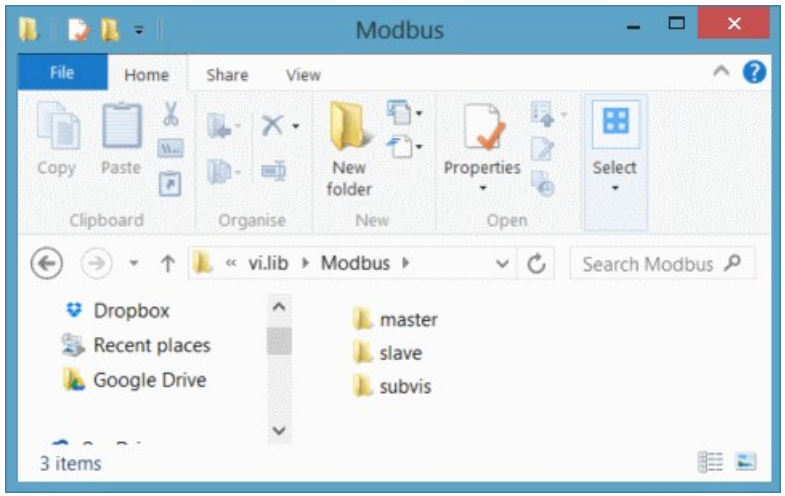
Sort Conflicts

Project Portability
Case Study - Step 6 Cleaning House



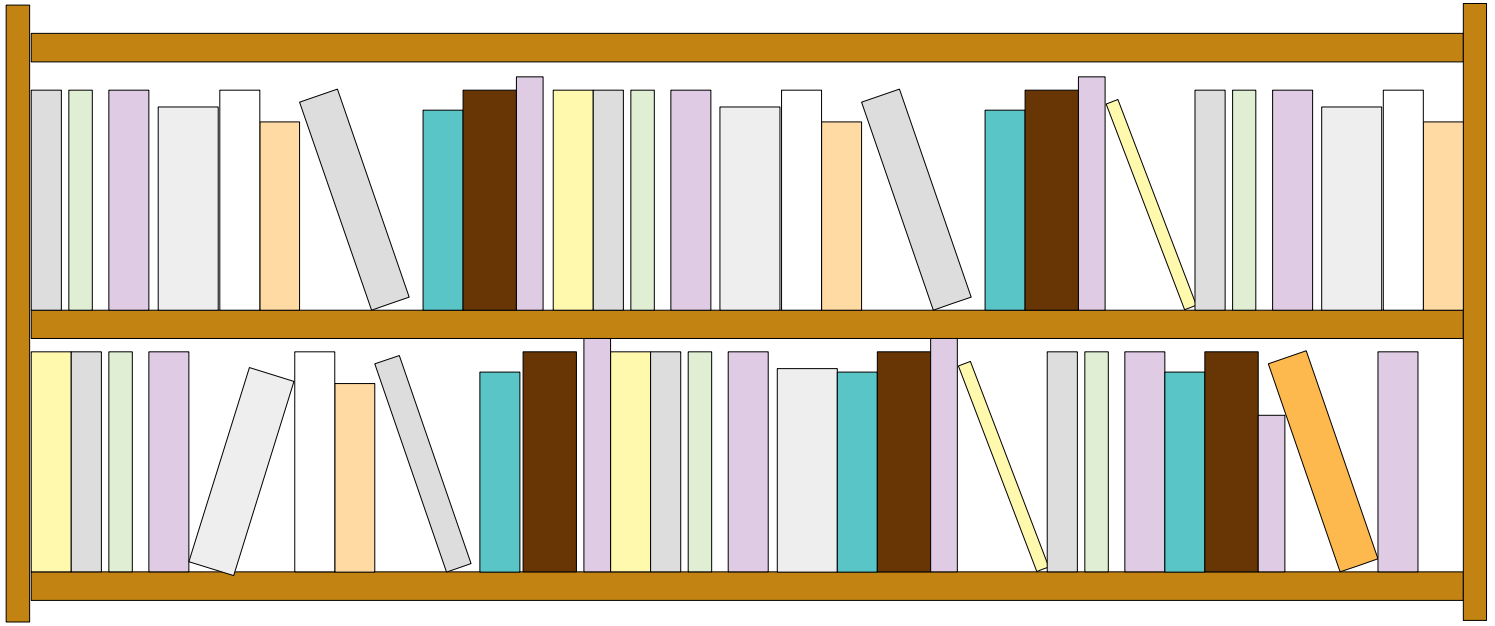
Why is this in dependencies?

Project Portability
Case Study - Step 7 Making It Portable



LabVIEW likes file hierarchy

View Dependencies with suspicion



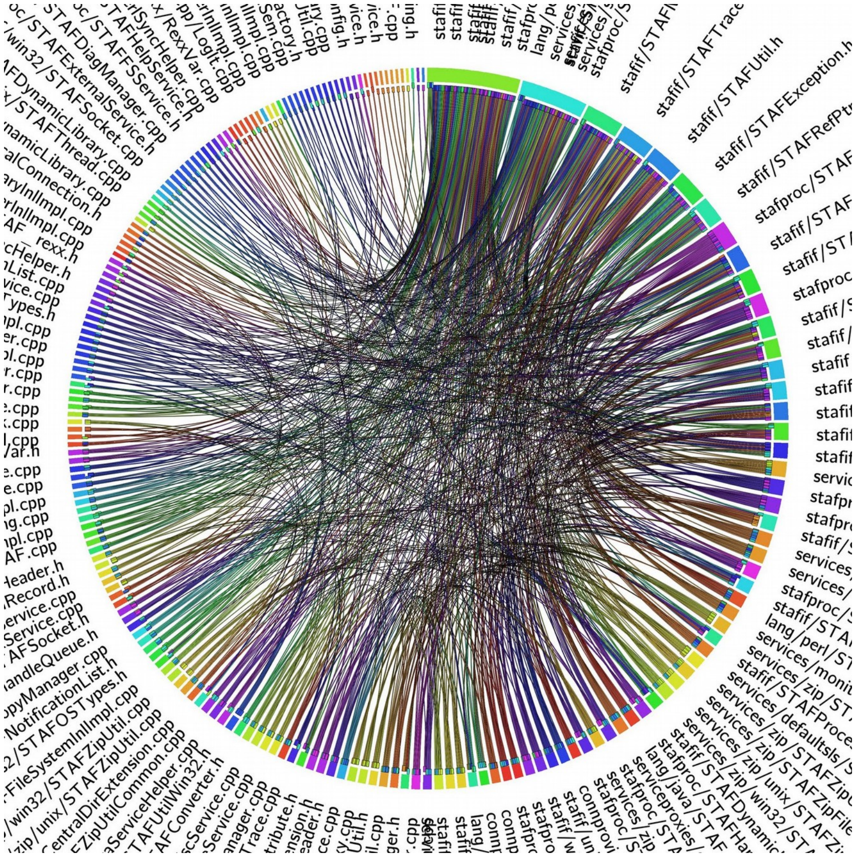
Picking from a library should not be the first consideration

View Dependencies with suspicion

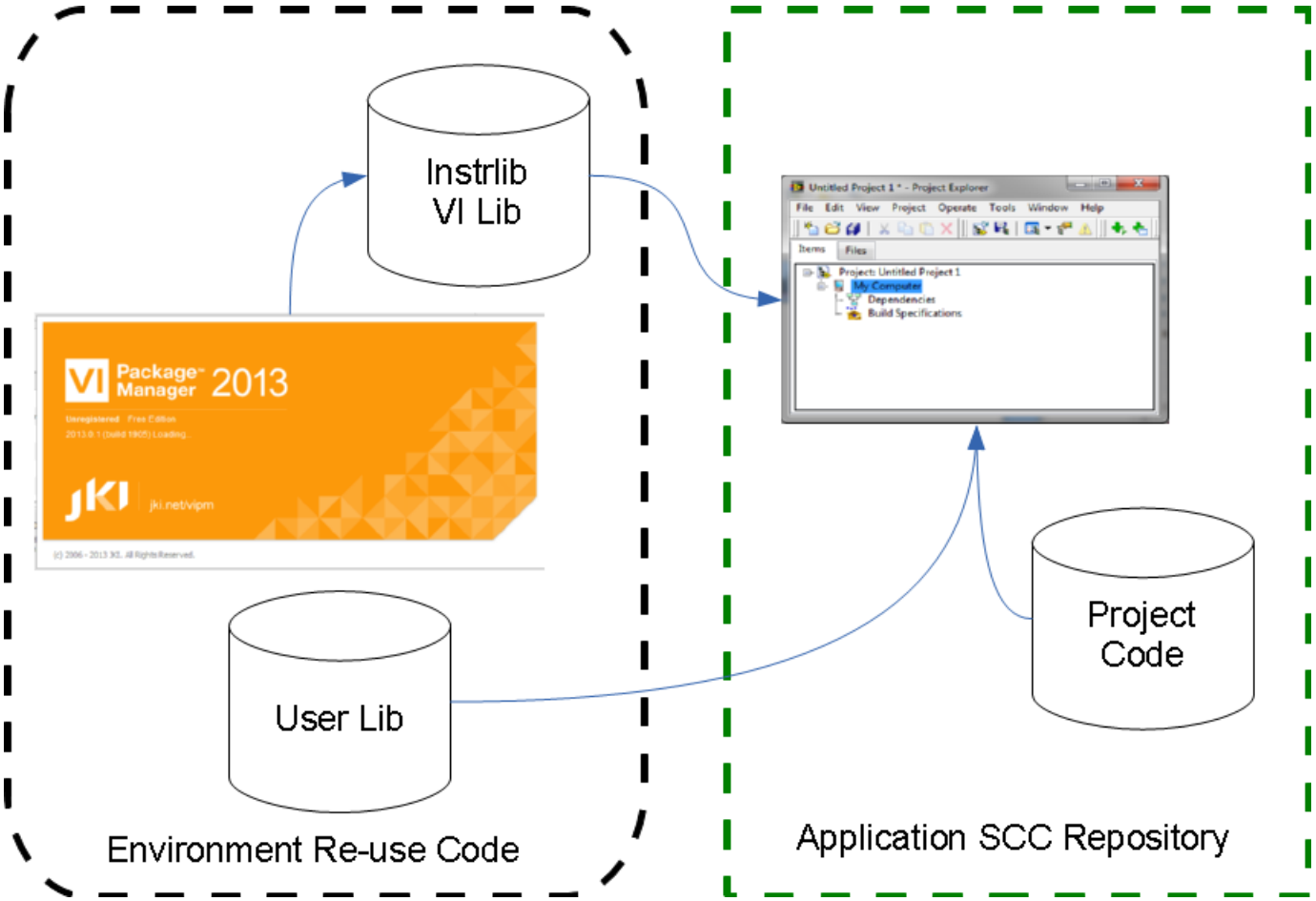
Learning to use a tool rather than learning how the tool works

View Dependencies with suspicion

"It can be better to copy a little code than to pull in a big library for one function. Dependency hygiene trumps code reuse."
Rob Pike



View Dependencies with suspicion



How much of your editable code is under SCC?

Polymorphism – Bloat Alert

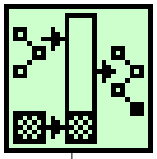
A traditional polymorphic VI is very handy to simplify selection from function palette, but beware that after selection you get bloat with little return

Polymorphism – Bloat Alert

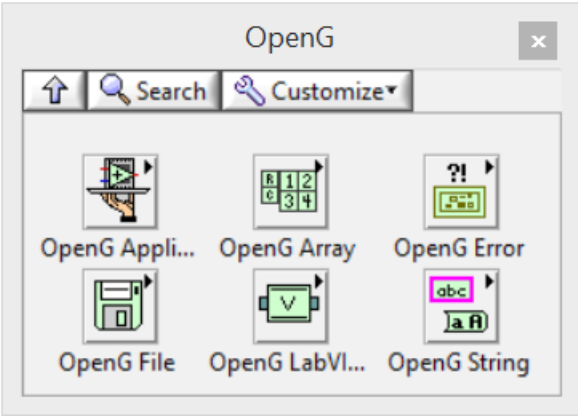
Extensible: "but suppose if someone wants to add X here tomorrow". Be wary unless tomorrow is a real date, and someone is a real person.....

Consider just using the instance you need and not the entire polymorphic VI

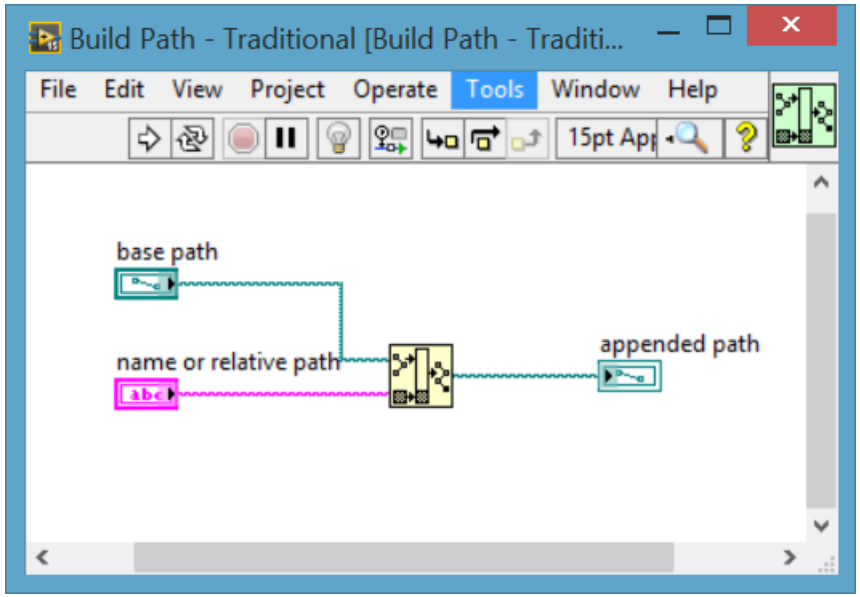
Is there a prim for that?



Comes with this



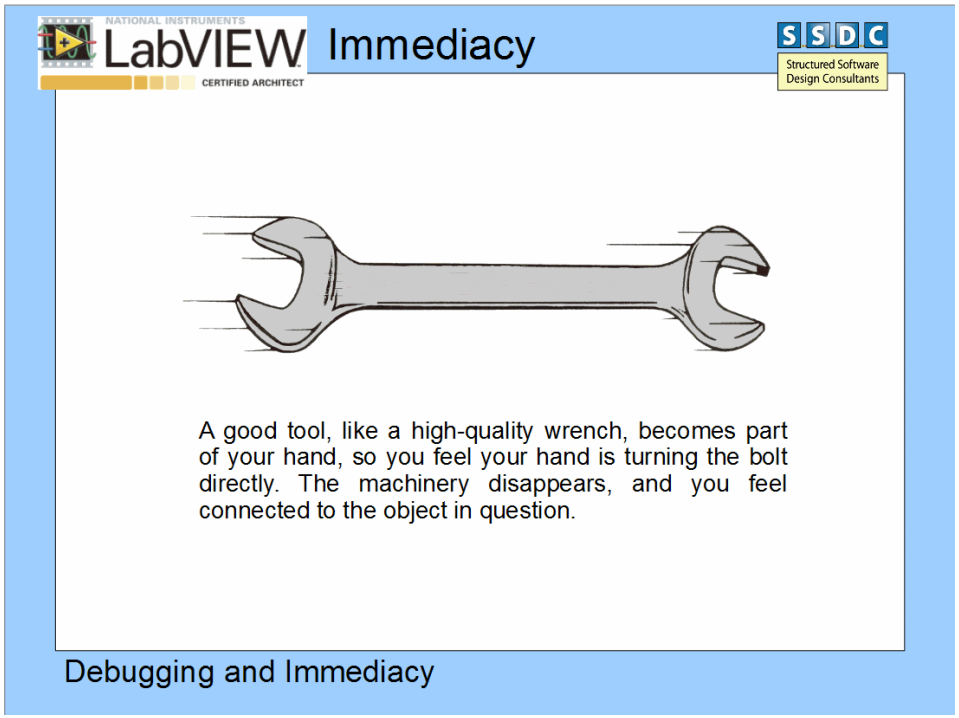
This is the block diagram



Immediacy – Glue Vs Do

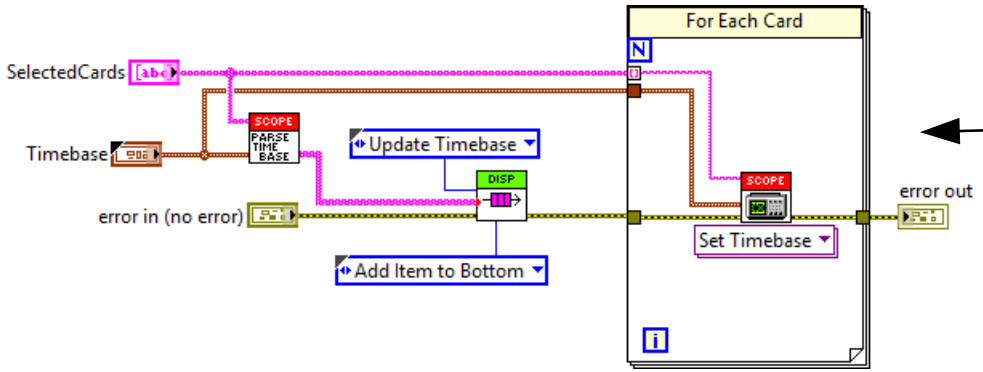
You'll spend a lot of you time messing around with your codebase.

Tools, Design and Development Environment all have an impact on how productive you are. Immediacy is how responsive these are for the developer.



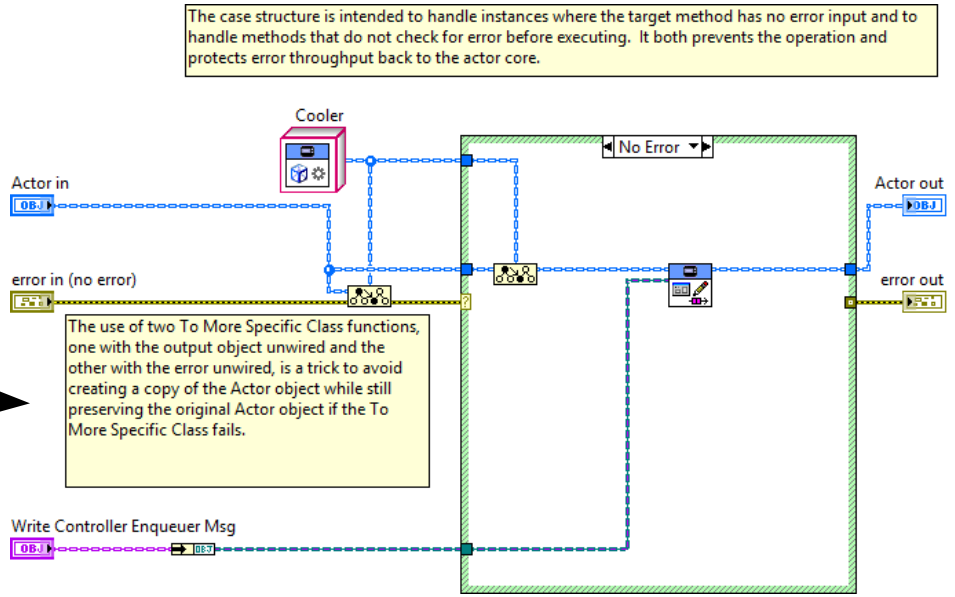
What's Immediacy

Immediacy – Glue
VIs that can't be tested immediately



The Queue VI needs an initialised Queue or it throws and error

A framework may have many VIs that do not run without running the entire framework



VIs That Don't Run or that don't do anything

Immediacy – Immediately Testable VIs

Where a single VI (and subVIs)
can be stopped and the input data fiddled with or the block diagram changed immediately.

This is one of LabVIEWs main benefits, why discard it?

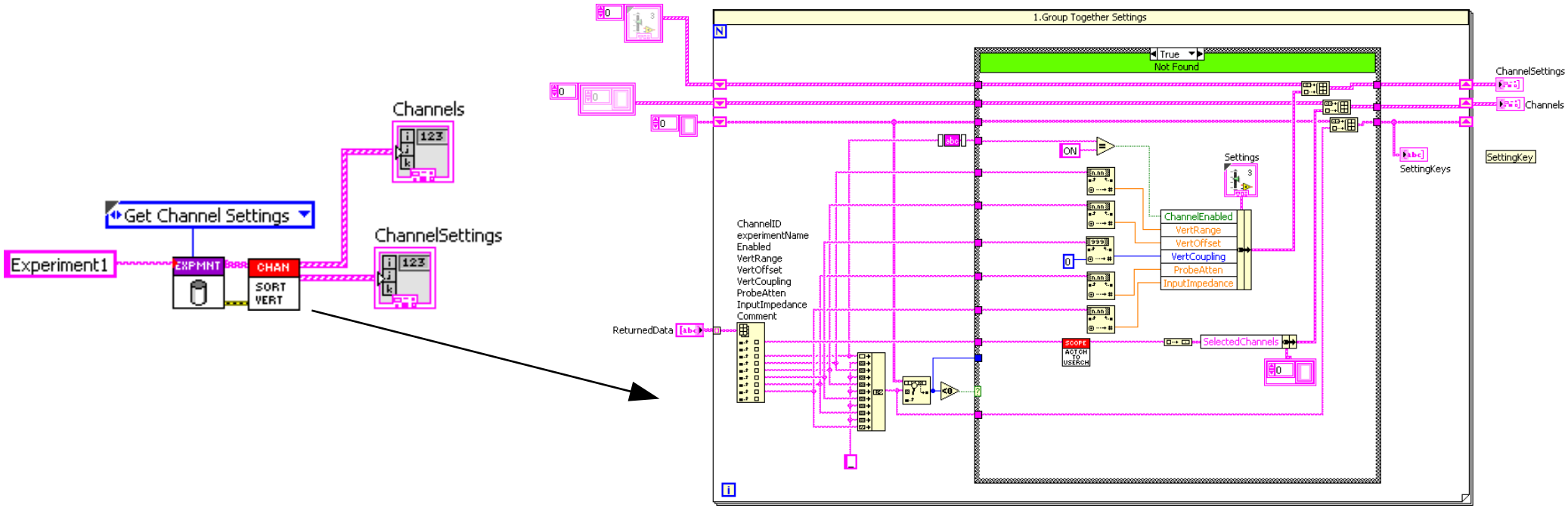
Immediacy – Design to take advantage of Immediacy

Separate Running code from Non-running code

Immediacy – Do

Separate Glue VIs from Algorithm (Do) VIs

Bending cohesion



VIs That Run and do stuff immediately

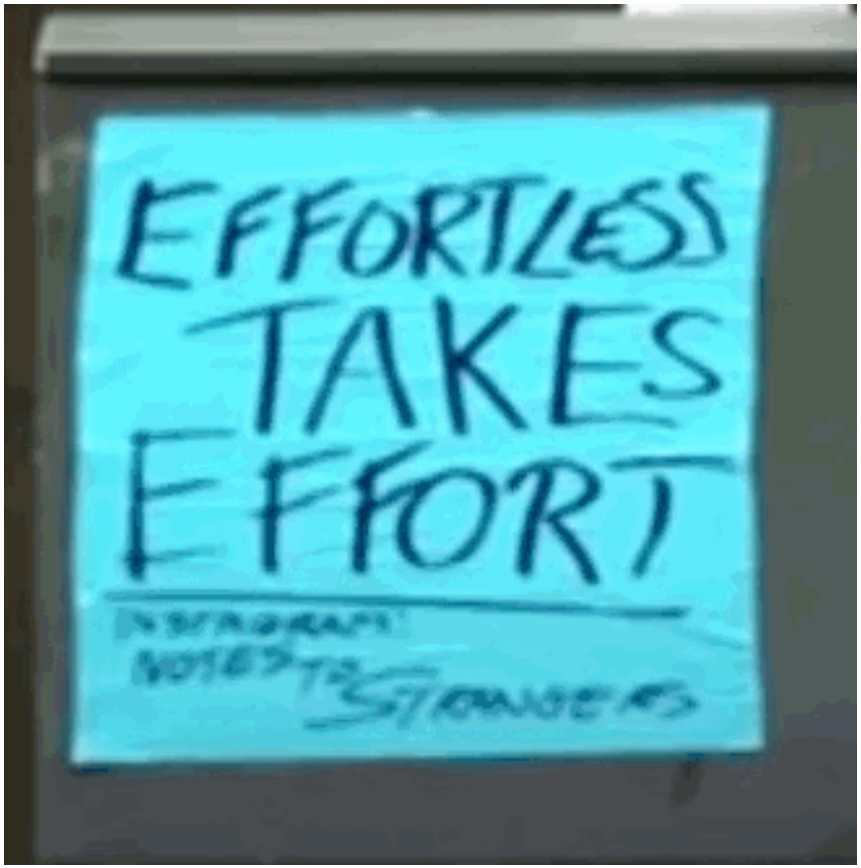
Immediacy – ITVIs – Lean Unit Testing

Focus your Unit Testing on Immediately Testable VIs
And integration testing on support VIs

Lean, Clean Code – Clean BDs

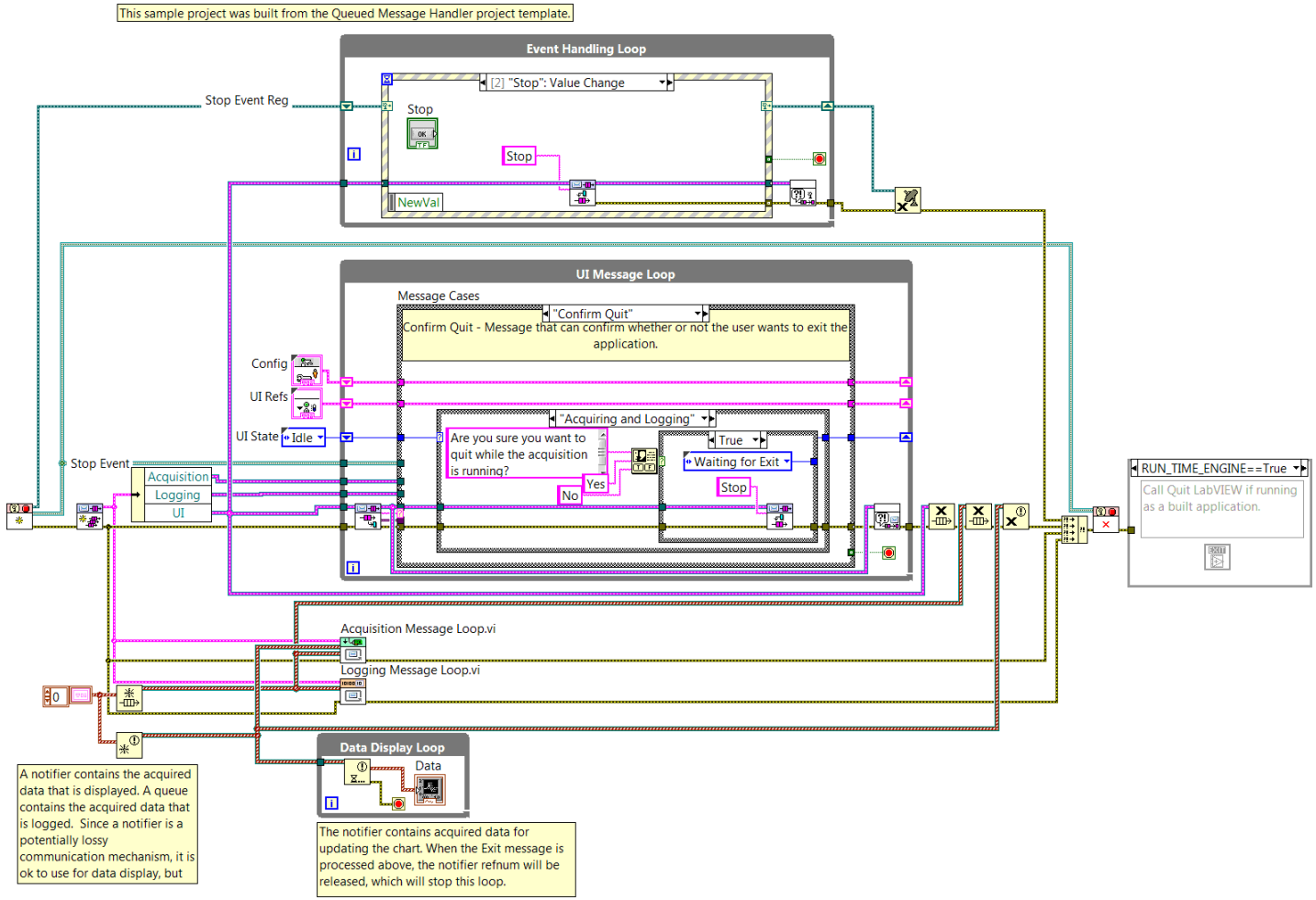
“Programs must be written for people to read, and only incidentally for machines to execute.”

— Harold Abelson, 1984



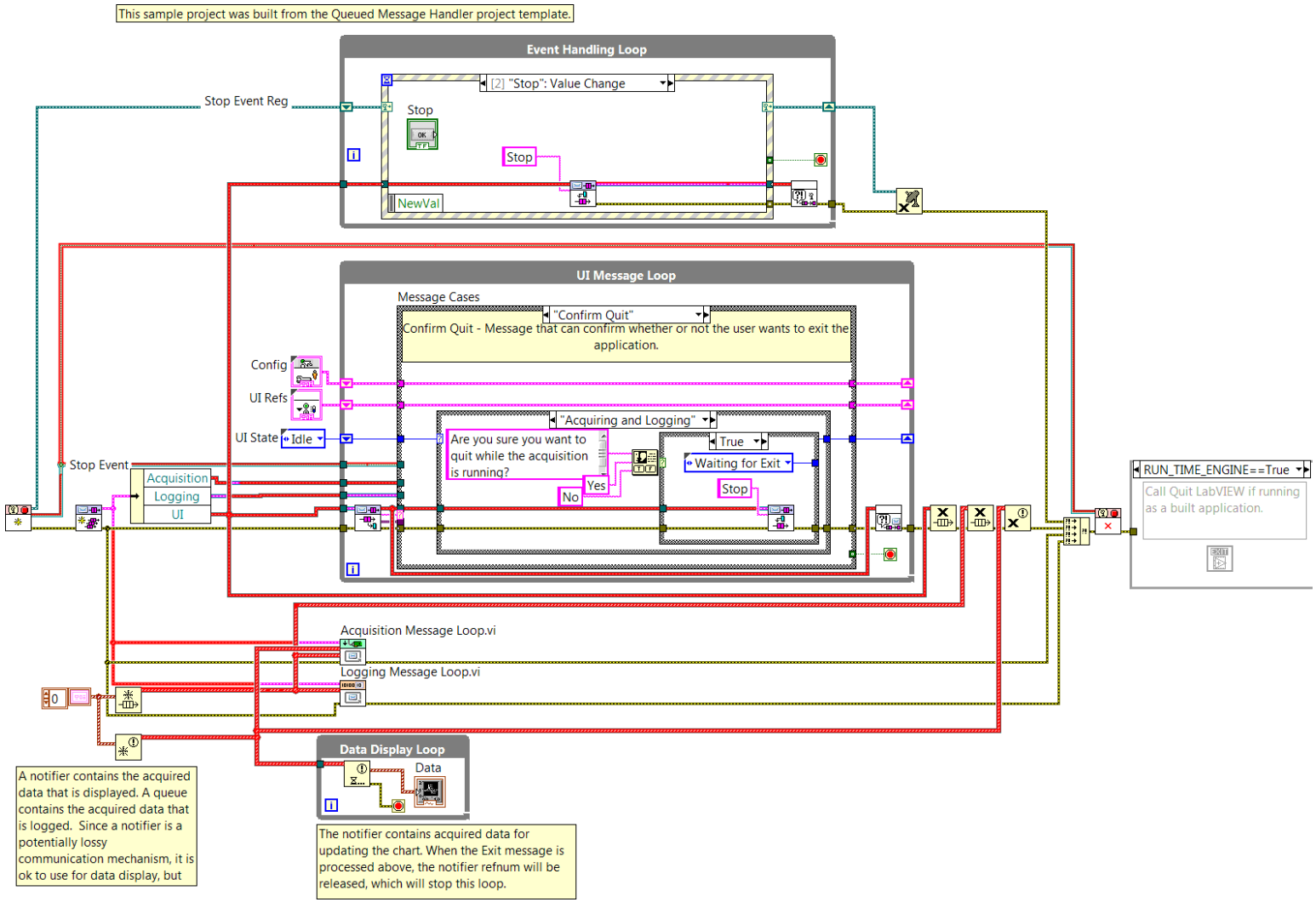
Program slow, debug fast

Lean, Clean Code – Clean BDs



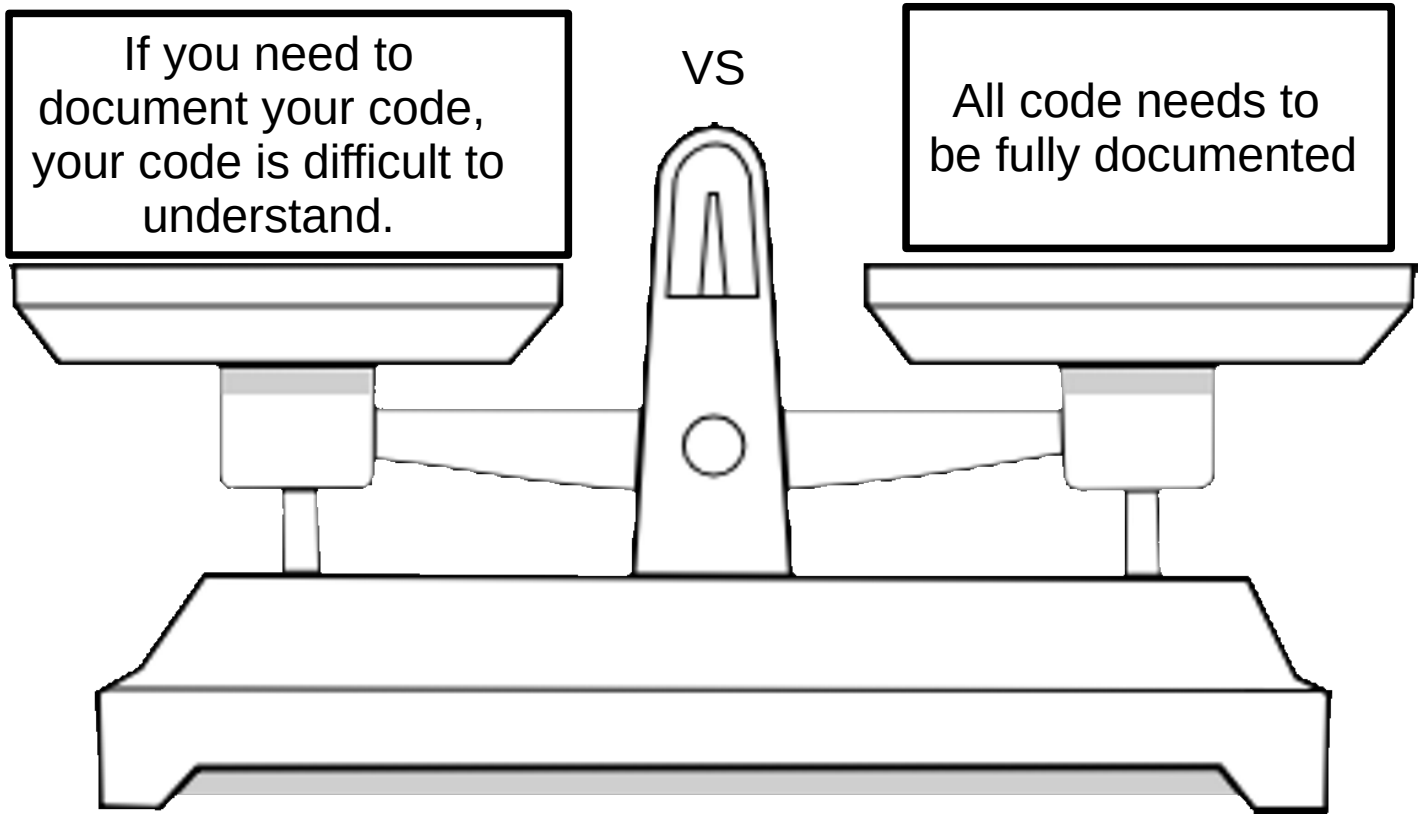
Keep your block diagram clean!

Lean, Clean Code – Clean BDs



Red Lines convey little information

Lean, Clean Code



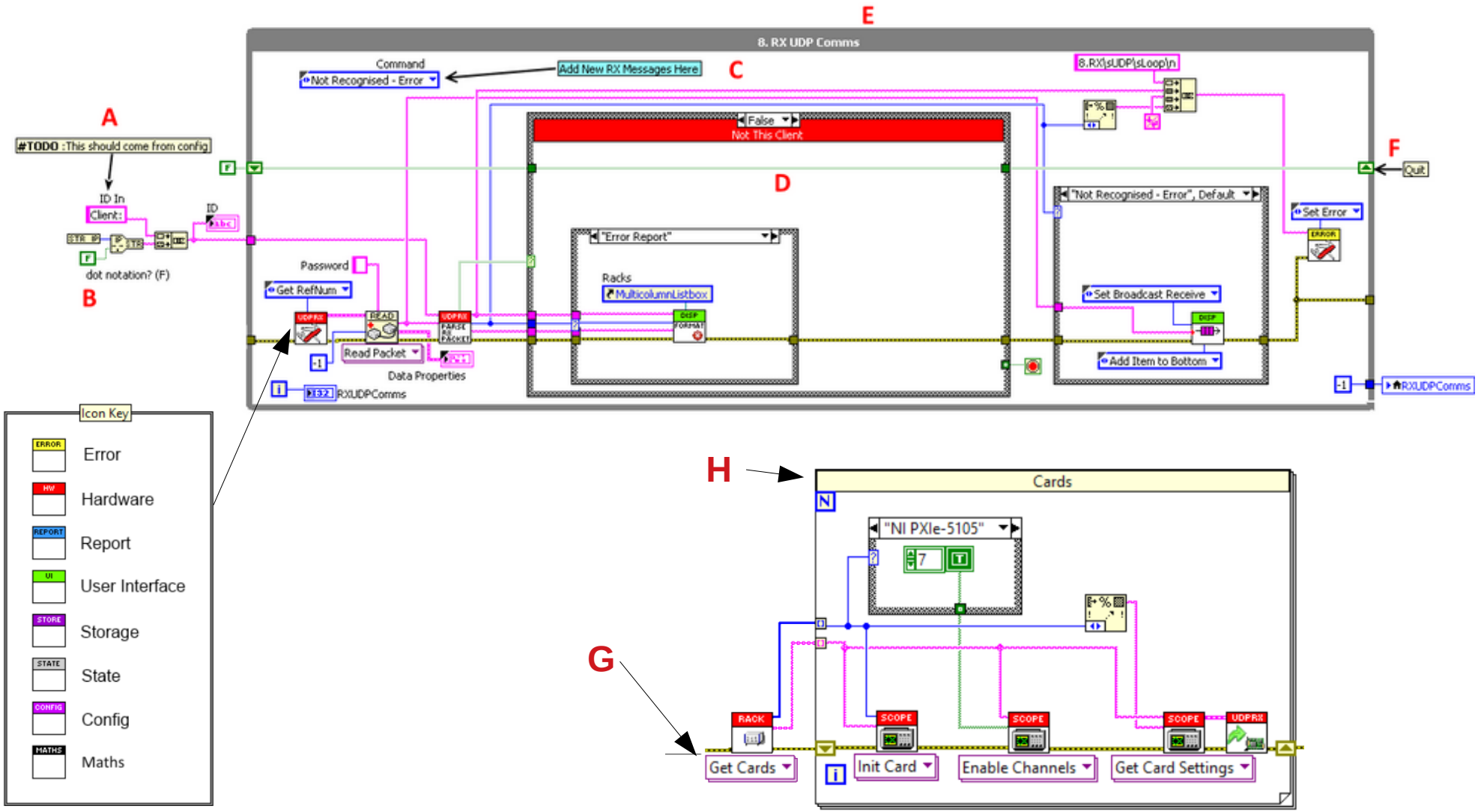
Comments

Lean, Clean Code

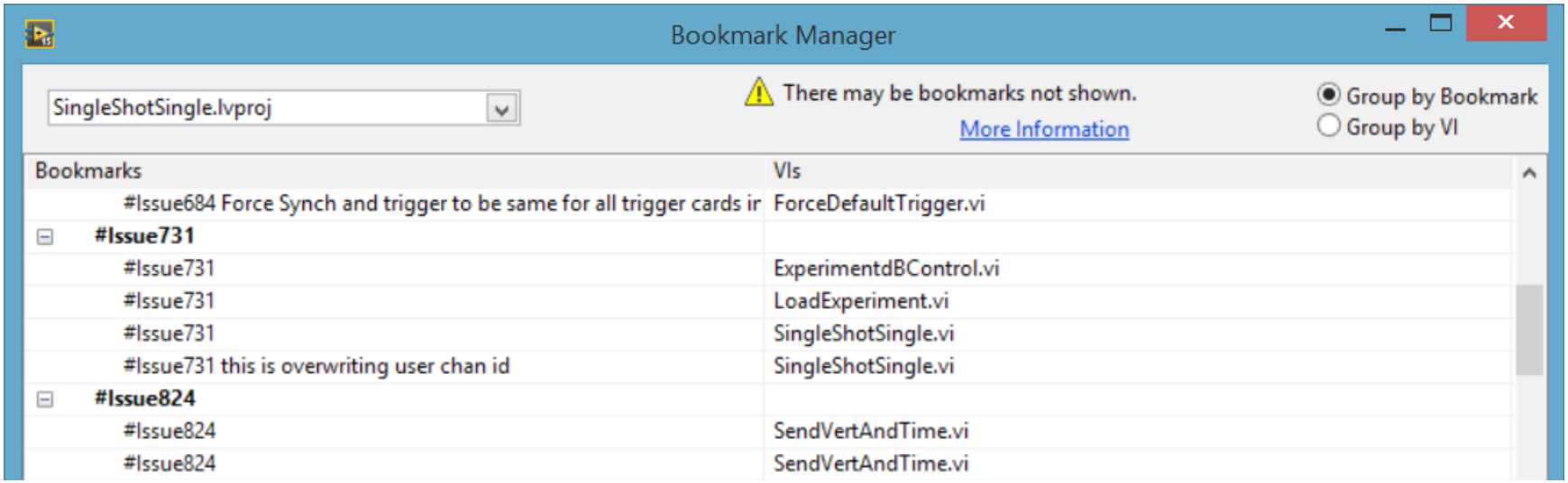
A common fallacy is to assume authors of incomprehensible code will somehow be able to express themselves lucidly in comments.
K Henney

Comments

Lean, Clean Code

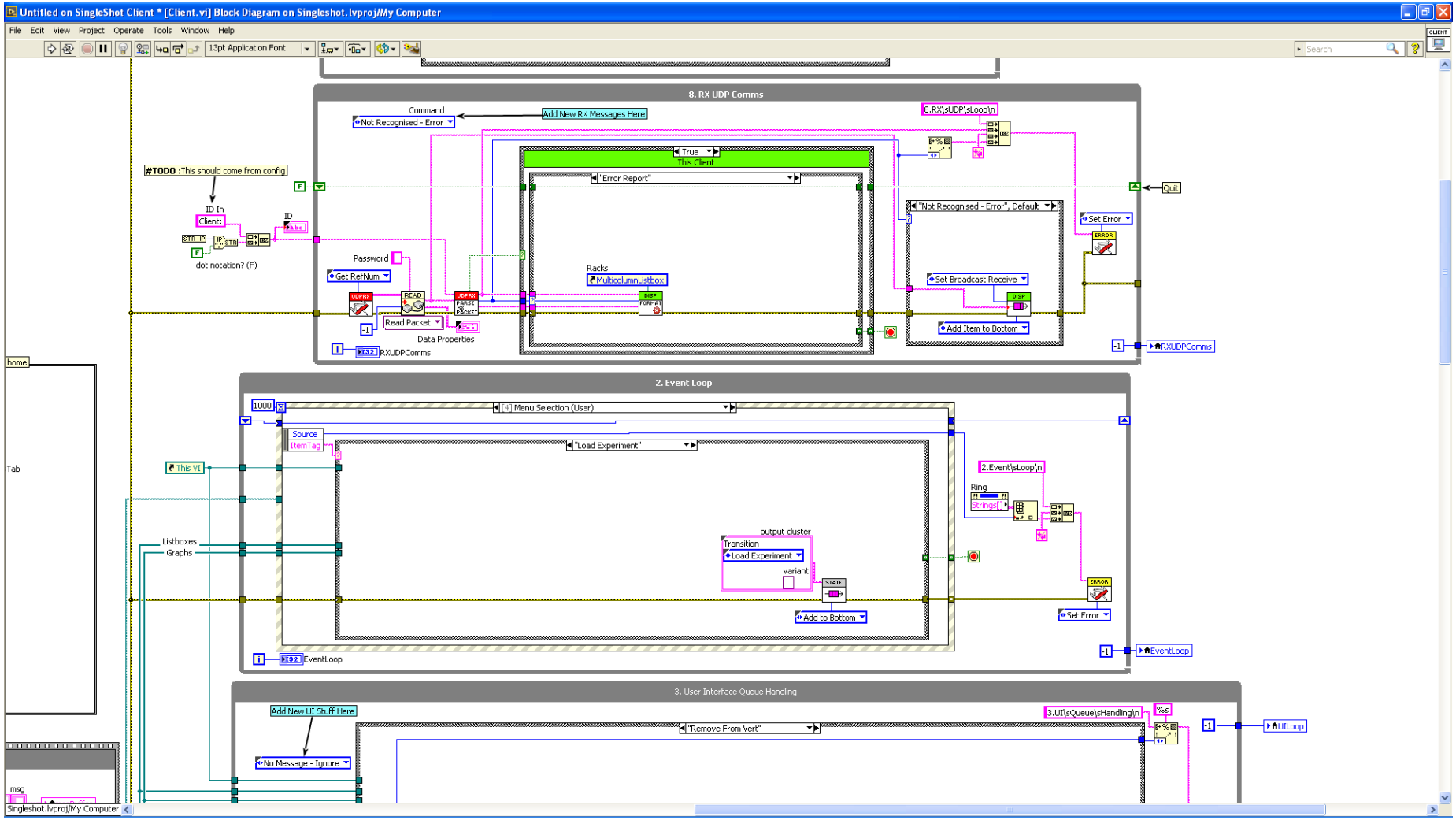


Lean, Clean Code



Issue Navigation in the code

Lean – Error Handling



Do you really want to know about errors at a VI level?


G DEV CON #1
Cambridge UK 4-5th September 2018

Structured Software Design Consultants

Steve Watts

SSDC Ltd
Lean LabVIEW
swatts@ssdc.co.uk
@swatzyssdc

Structured Software Design Consultants

NATIONAL INSTRUMENTS Silver Alliance Partner

18 Years

CSLUG

GDEV CON #1
Cambridge UK 4-5th September 2018

eCLA 2012 LabVIEW Smells
NIDays 2013 How to Polish Your Software and Development Process to Wow Your End Users
eCLA 2014 Process Smells
Various 2015 Immediacy
NIWeek 2016 TS9456 - ISO 9000 and LabVIEW
NIWeek 2016 PS9044-Shock Testing using Multiple Synchronised Racks
ECLA 2017 Risk and Mitigation
NIWeek 2017 The SSDC Way: Desire Paths to a Simple Software Process
ECLA 2018 Modular Design Deep Dive

Read my blog on ni.com – Random Ramblings on LabVIEW Design

Goal:

Show techniques that allow SSDC to produce large complex projects while keeping them responsive and light.

i.e. how to play nice with LabVIEW

What is Lean LabVIEW and Why?

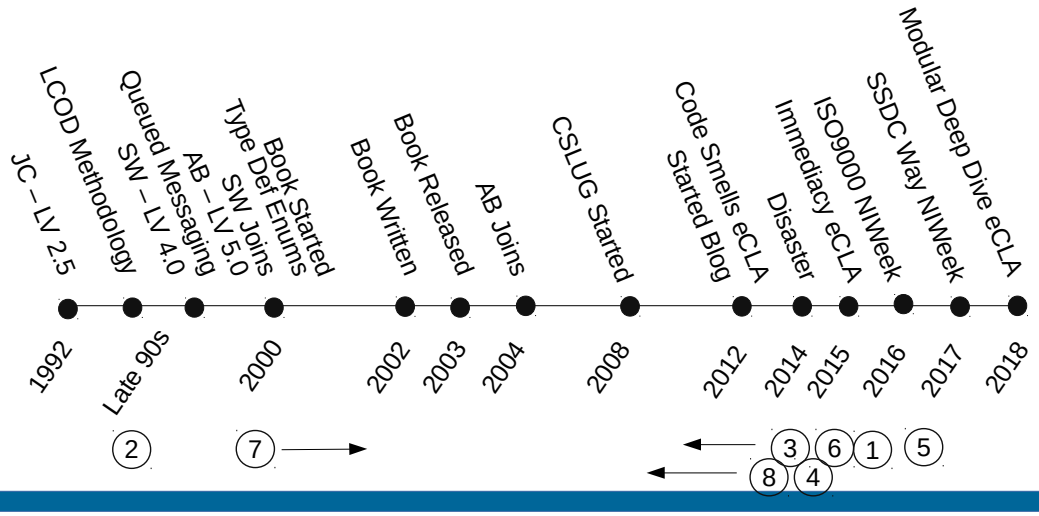
Key Aspects

1) A Tidy Project is a Happy Project
2) Cross Linking
3) Project Portability
4) View Dependencies with suspicion
5) Polymorphism – Bloat Alert
6) Immediacy – Glue Vs Do
7) Lean, Clean Code
8) Lean Error Handling

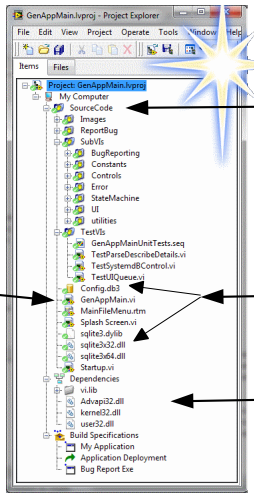
Structured Software Design Consultants

The importance of time

These techniques have been used by SSDC for many years and are proven by our many projects in many industries.



A Tidy Project is a Happy Project



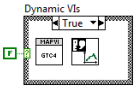
Auto-populating folders vision of change is important

Main Calling VI sits atop the hierarchy

dlls, config db etc are all at top level

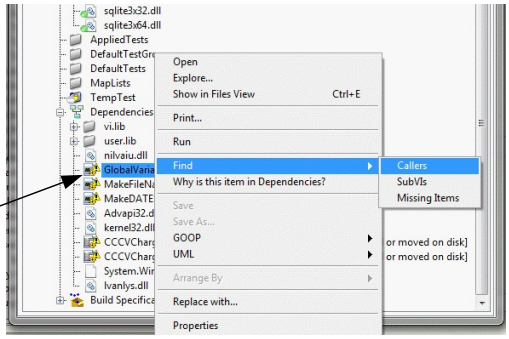
Uncompromising about dependencies

A Tidy Project is a Happy Project



Get LabVIEW to check your dynamic vis

Eliminate these



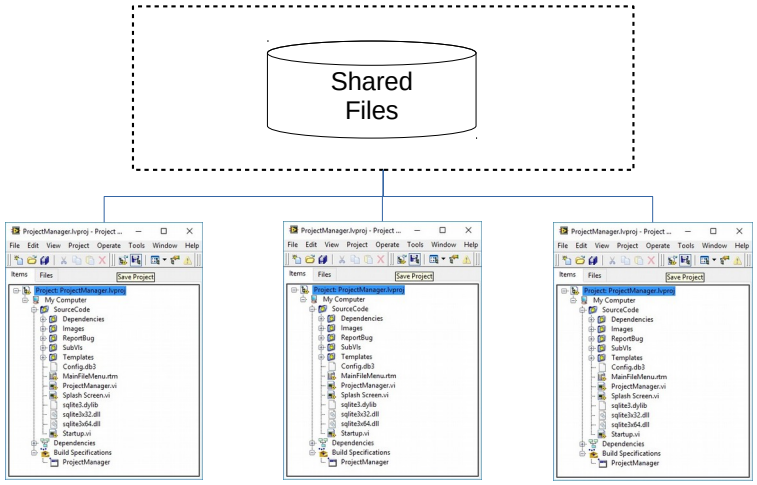
Cross-linking

Vic Stennings Principles of Project Hygiene (1979)

Principle 5
Changes should be controlled, visible and of known scope.

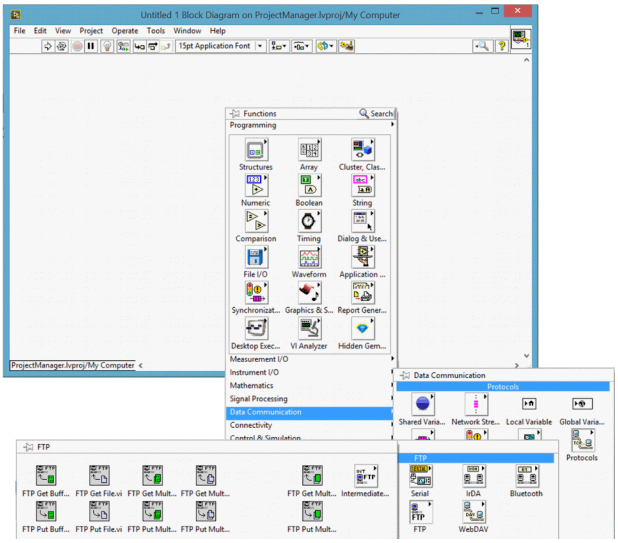
The enemy for predictable software is silent changes

Cross-linking



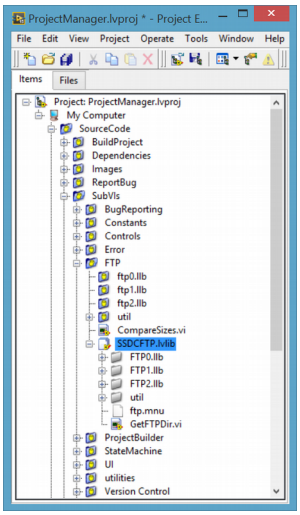
Silent changes

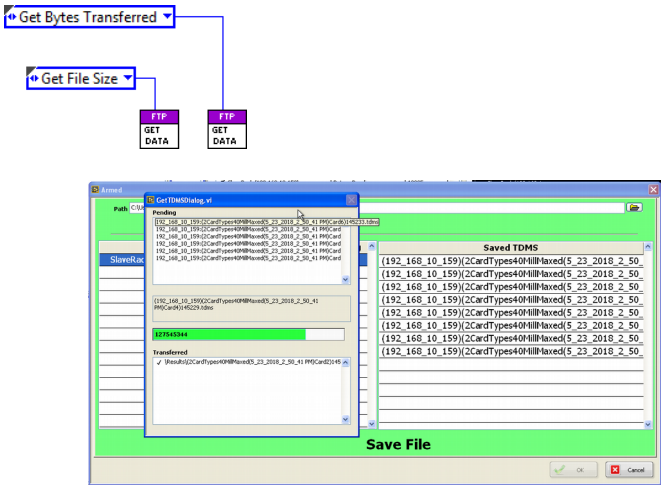
Cross-linking – Environmental changes



FTP Example

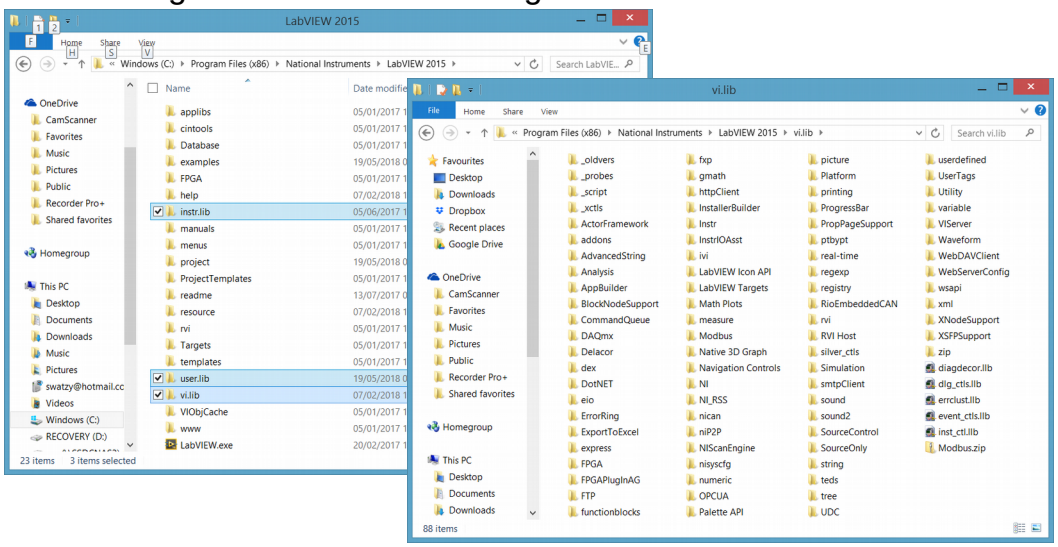
Cross-linking – Environmental changes





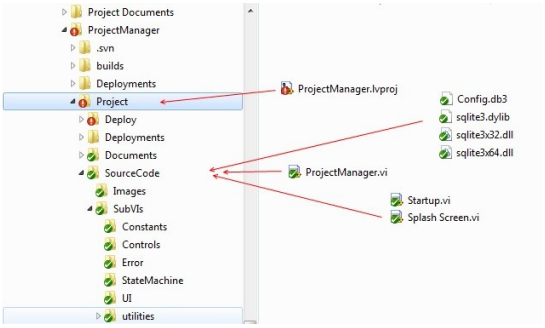
FTP Example

Cross-linking – Environmental changes

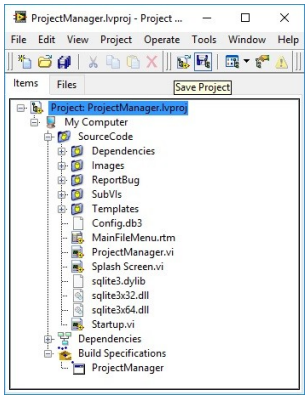


These all have VIs you can edit

Project Portability



Project Portability



The test for portability is one of our code review items and involves exporting the repository into a clean virtual machine (with LabVIEW Loaded) and into different directories (usually desktop and \User\Public\Something or other.

Structured Software Design Consultants

SSDC Limited	
Description	Code Review Check List
Issue	1.02
Date	Tuesday, 06 May 2014
Document Number	2013-175\M0004(Code Review Check List)

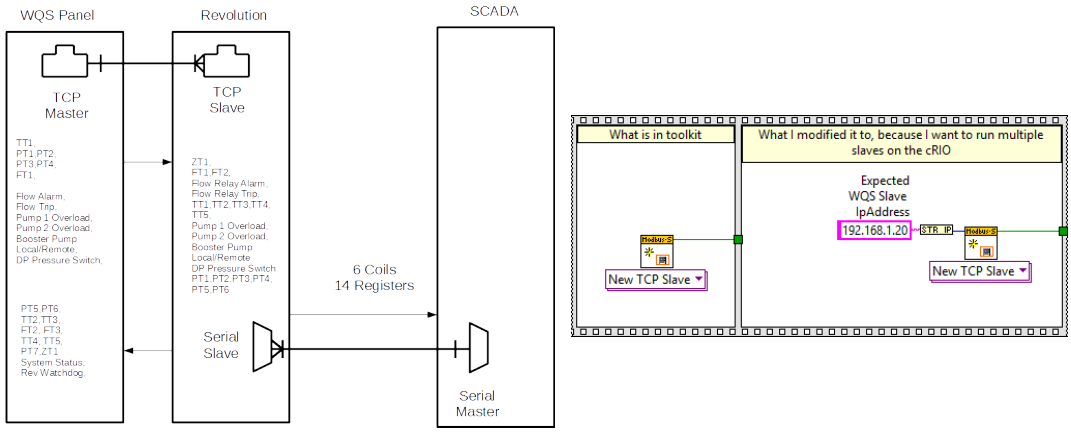
Initial Checks

Check out from repository prior to starting. Ideally onto a clean machine or virtual machine

Details

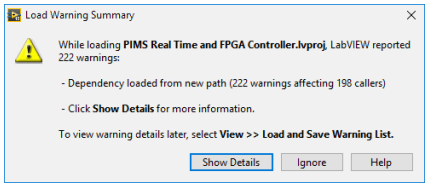
This will test that the repository is set up correctly and the project can run, any dependencies and linked libraries, hardware libraries will need noting and their version documented.

Project Portability
Case Study - Step 1 Move Directory from vi.lib



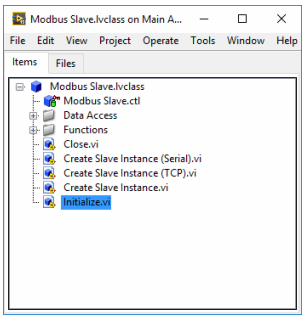
Why we wanted to modify the library

Project Portability
Case Study -Step 2 Loading Project With Embedded Library



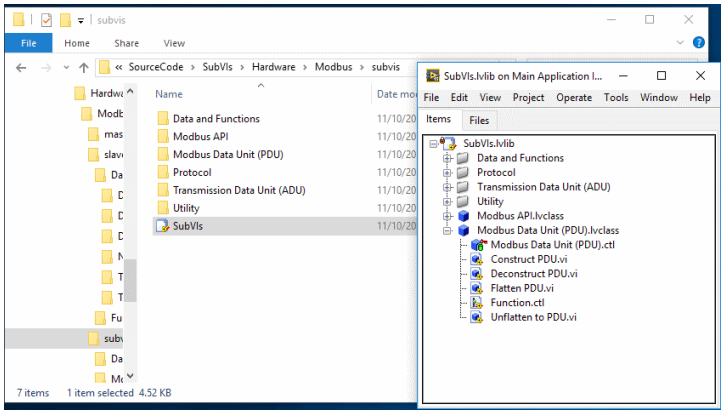
Warnings!

Project Portability
Case Study - Step 3 Tackle Slave Class On Its Own



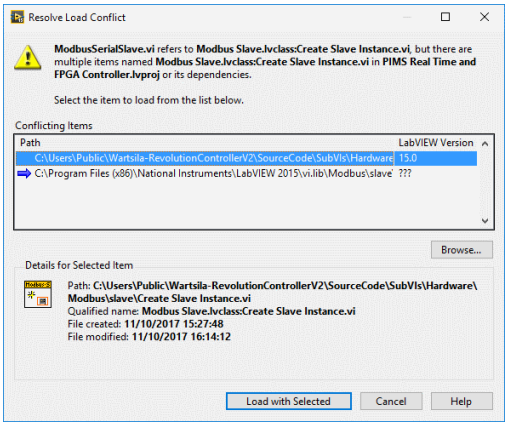
Only use what you need – this is project work, not API

Project Portability
Case Study - Step 4 Saving SubVIs.lvlib



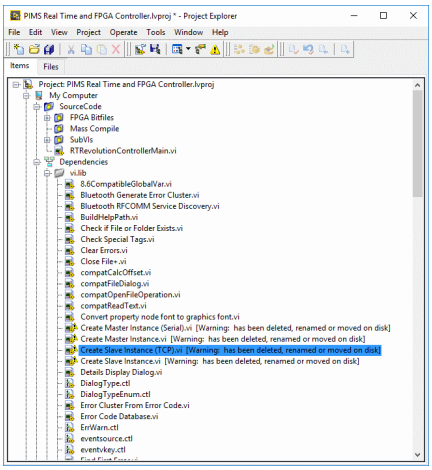
LabVIEW likes file hierarchy

Project Portability
Case Study - Step 5 Back In The Project



Sort Conflicts

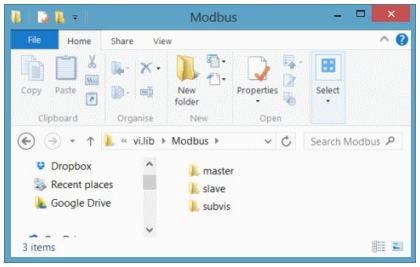
Project Portability
Case Study - Step 6 Cleaning House



Why is this in dependencies?

SSDC
Structured Software Design Consultants

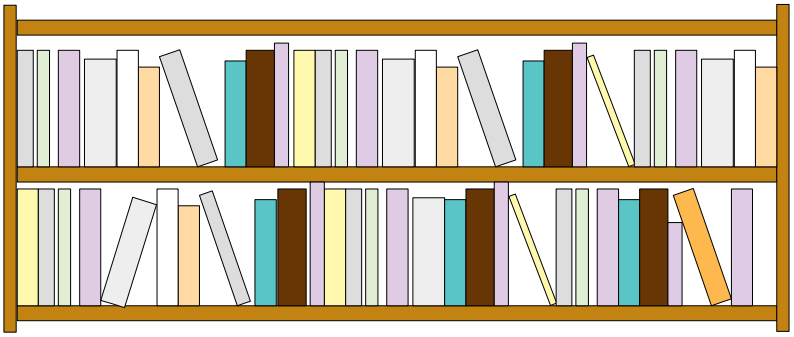
Project Portability
Case Study - Step 7 Making It Portable



LabVIEW likes file hierarchy

I moved the subvis subdir into the slave subdir and deleted the master directory as it was not required. And now I can export it from my repo to anywhere and it will work. I've said this before but never actually done the hard graft to prove it

View Dependencies with suspicion



Picking from a library should not be the first consideration

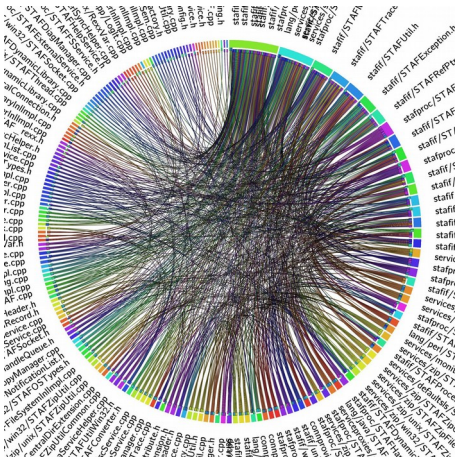
View Dependencies with suspicion

Learning to use a tool rather than learning how the tool works

SSDC
Structured Software Design Consultants

View Dependencies with suspicion

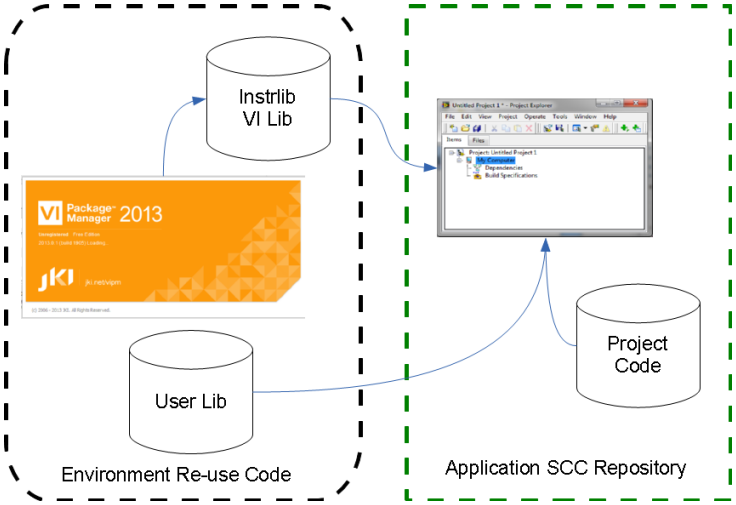
"It can be better to copy a little code than to pull in a big library for one function. Dependency hygiene trumps code reuse."
Rob Pike



Robert Pike (born 1956) is a Canadian software engineer and author. He is best known for his work at Bell Labs, where he was a member of the Unix team and was involved in the creation of the Plan 9 from Bell Labs and Inferno operating systems, as well as the Limbo programming language.

Also Go Language used for youtube, bbc

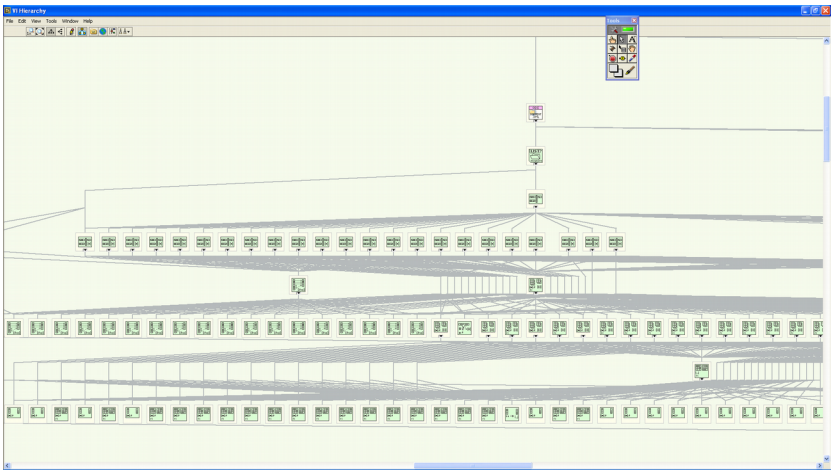
View Dependencies with suspicion



How much of your editable code is under SCC?

Polymorphism – Bloat Alert

A traditional polymorphic VI is very handy to simplify selection from function palette, but beware that after selection you get bloat with little return



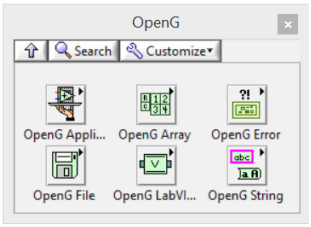
Polymorphism – Bloat Alert

Extensible: "but suppose if someone wants to add X here tomorrow". Be wary unless tomorrow is a real date, and someone is a real person.....

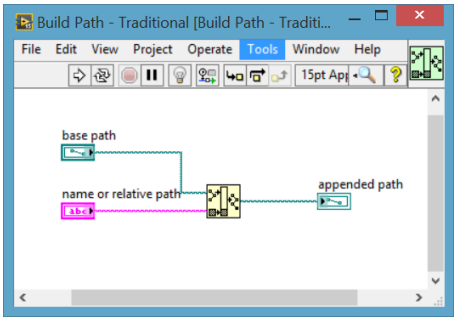
Consider just using the instance you need and not the entire polymorphic VI

Is there a prim for that?

Comes with this



This is the block diagram

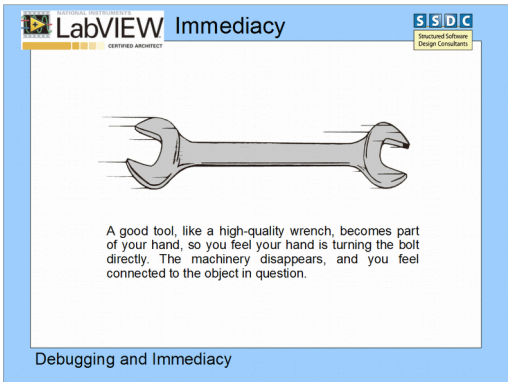


SSDC Structured Software Design Consultants

Immediacy – Glue Vs Do

You'll spend a lot of you time messing around with your codebase.

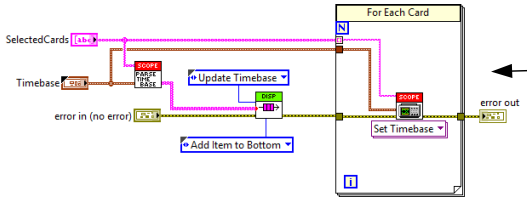
Tools, Design and Development Environment all have an impact on how productive you are. Immediacy is how responsive these are for the developer.



What's Immediacy

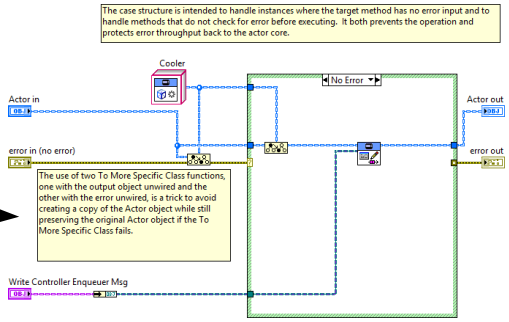
Brian Powells comments on continuous measurement and logging example vs DAQ

Immediacy – Glue
VIs that can't be tested immediately



The Queue VI needs an initialised Queue or it throws and error

A framework may have many VIs that do not run without running the entire framework



VIs That Don't Run or that don't do anything

Brian Powells comments on continuous measurement and logging example vs DAQ

Immediacy – Immediately Testable VIs

Where a single VI (and subVIs)
can be stopped and the input data fiddled with or the block diagram changed immediately.

This is one of LabVIEWs main benefits, why discard it?

Brian Powells comments on continuous measurement and logging example vs DAQ

Immediacy – Design to take advantage of Immediacy

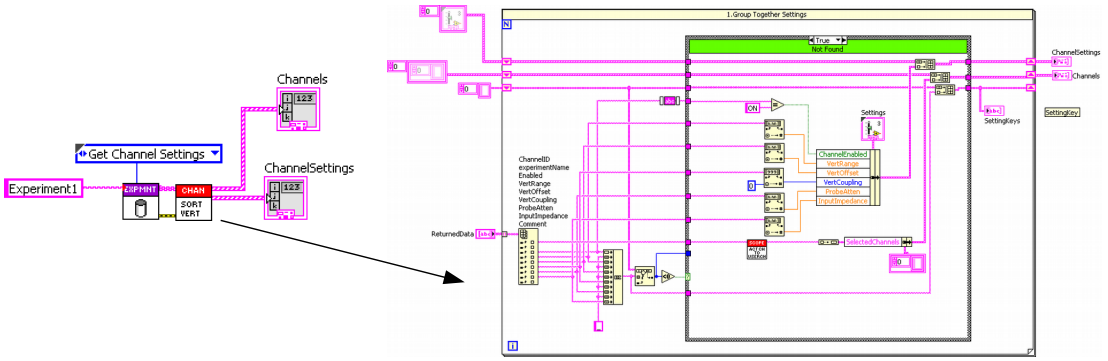
Separate Running code from Non-running code

Brian Powells comments on continuous measurement and logging example vs DAQ

Immediacy – Do

Separate Glue VIs from Algorithm (Do) VIs

Bending cohesion



VIs That Run and do stuff immediately

Brian Powells comments on continuous measurement and logging example vs DAQ

Focus your Unit Testing on Immediately Testable VIs
And integration testing on support VIs

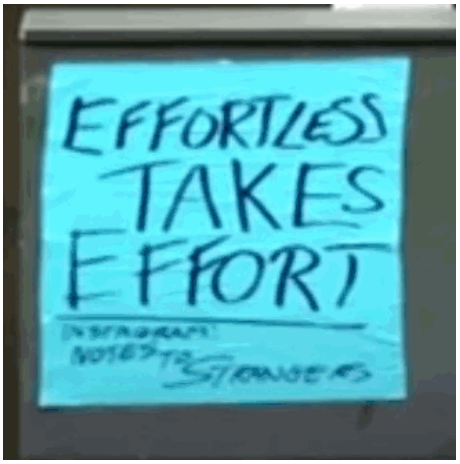
Brian Powells comments on continuous measurement and logging example vs DAQ

SSDC
Structured Software Design Consultants

Lean, Clean Code – Clean BDs

"Programs must be written for people to read, and only incidentally for machines to execute."

— Harold Abelson, 1984

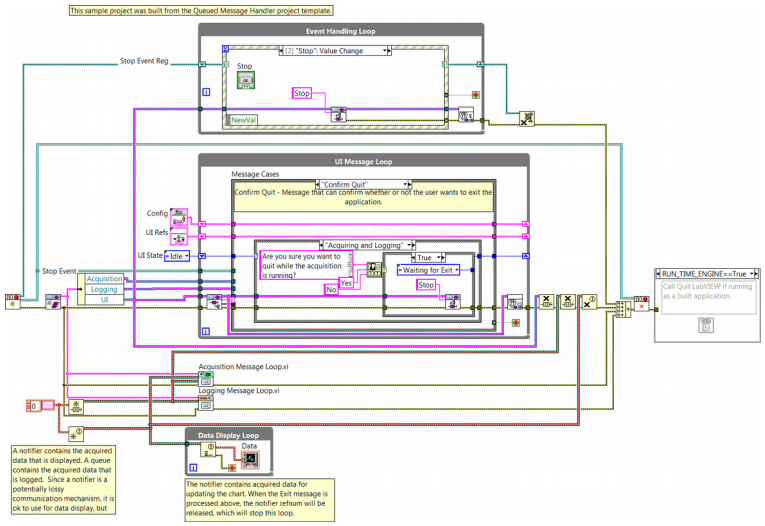


Program slow, debug fast

Brian Powells comments on continuous measurement and logging example vs DAQ

SSDC Structured Software Design Consultants

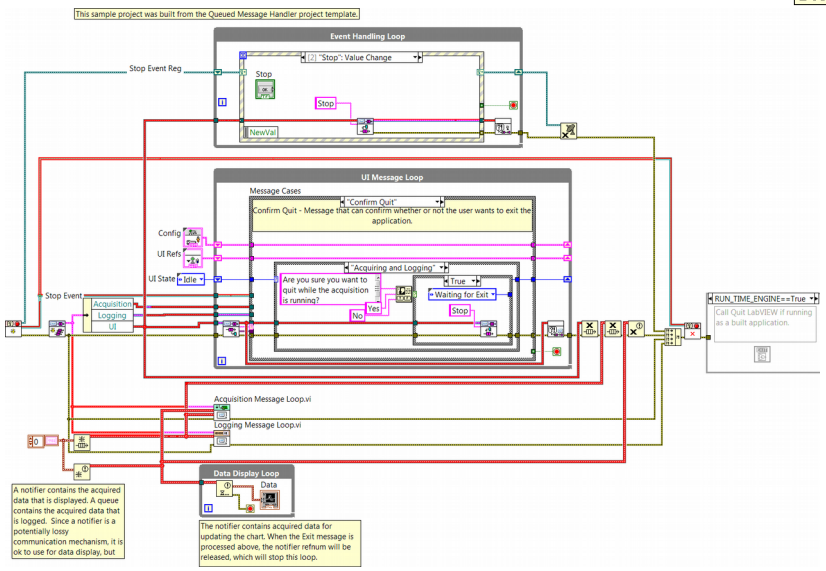
Lean, Clean Code – Clean BDs



Keep your block diagram clean!

Brian Powells comments on continuous measurement and logging example vs DAQ

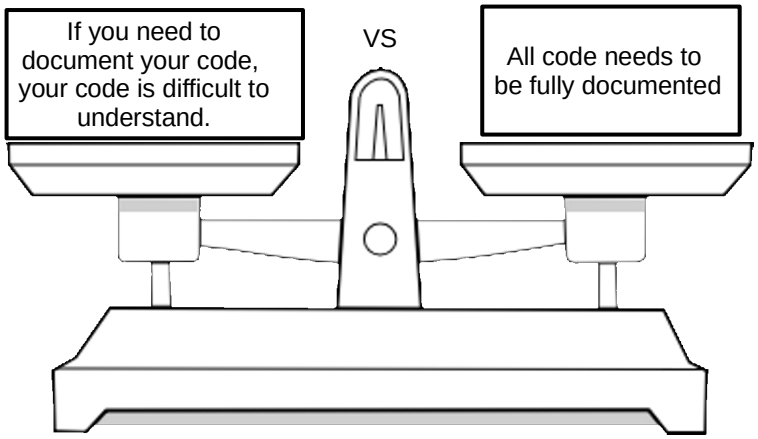
Lean, Clean Code – Clean BDs



Red Lines convey little information

Brian Powells comments on continuous measurement and logging example vs DAQ

Lean, Clean Code



Comments

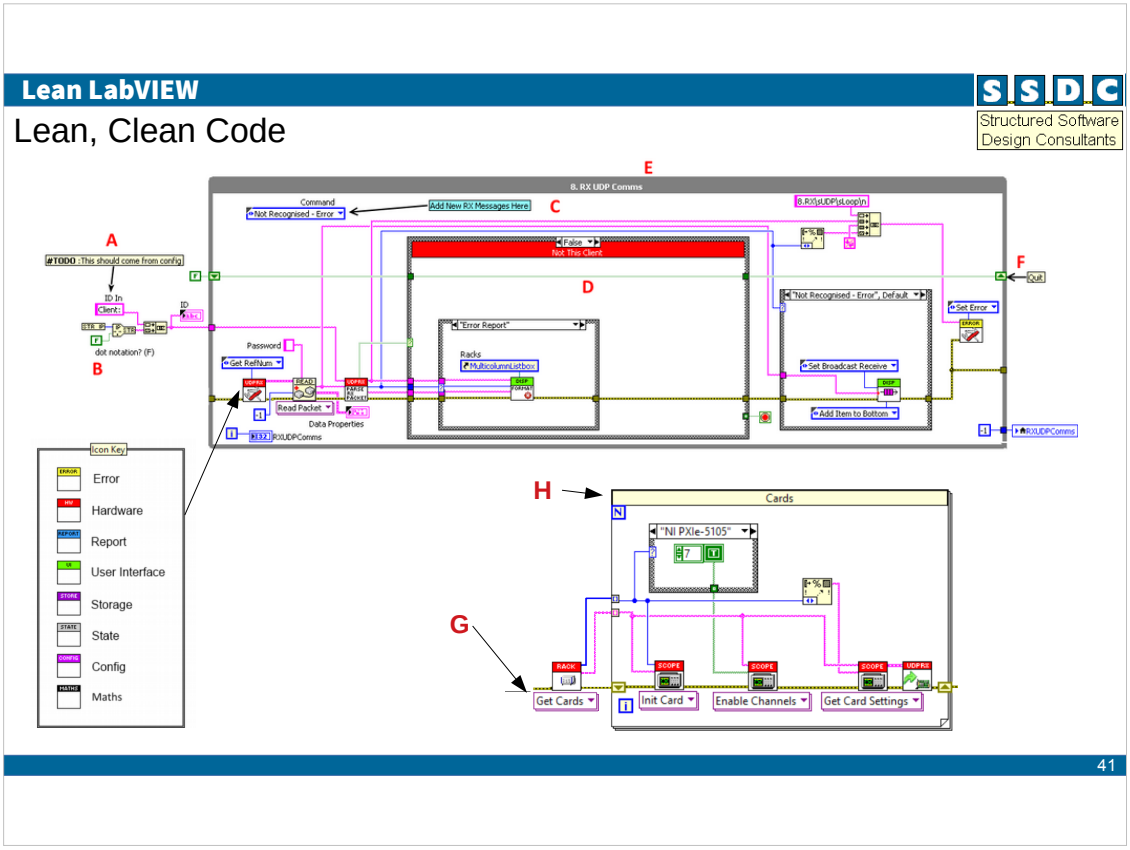
Brian Powells comments on continuous measurement and logging example vs DAQ

Lean, Clean Code

A common fallacy is to assume authors of incomprehensible code will somehow be able to express themselves lucidly in comments.
K Henney

Comments

Brian Powells comments on continuous measurement and logging example vs DAQ



Brian Powells comments on continuous measurement and logging example vs DAQ

A) Bookmarks are great, we use them for quick #TODO notices, tracking issue numbers and traceability for large changes.

B) Set LabVIEW to display constant label names, it improves the readability of the code at little cost.

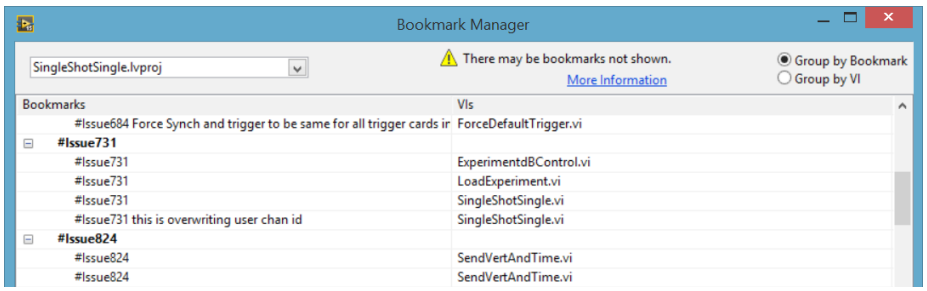
C) I like the blue template comments that indicate bits of the template that can be modified. In this case adding new messages to be handled by the UDP receive loop.

D) Labelling case statement logic, improves readability and as a bonus improves logical design. This is because if you have to write the logic down, it becomes very clear if it is becoming complex.

E) We Number and label our loops, this is good for telephone support to guide people to the correct structure and similar to D) labelling your loops clarifies your design decisions.

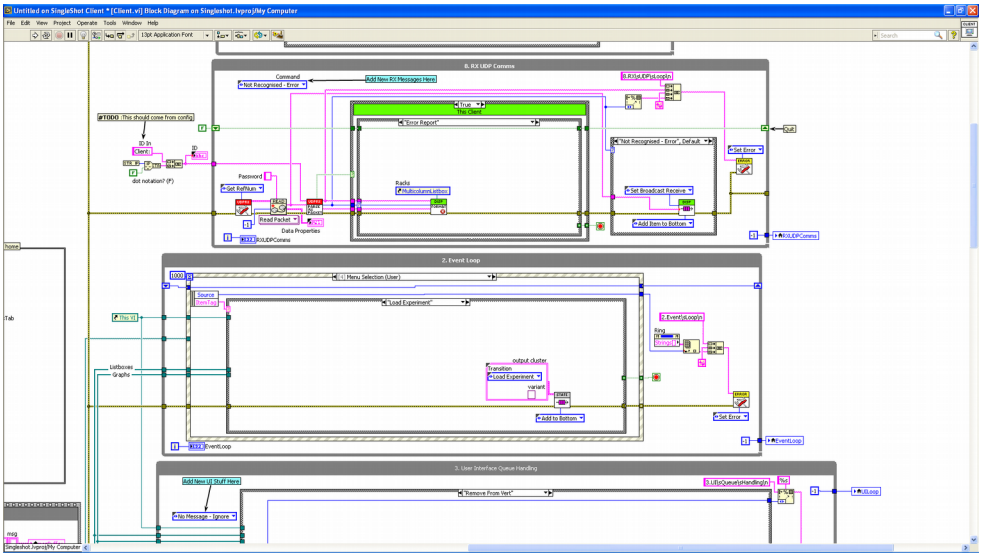
F) Shift registers benefit from being labelled

Lean, Clean Code



Issue Navigation in the code

Lean – Error Handling



Do you really want to know about errors at a VI level?

Brian Powells comments on continuous measurement and logging example vs DAQ