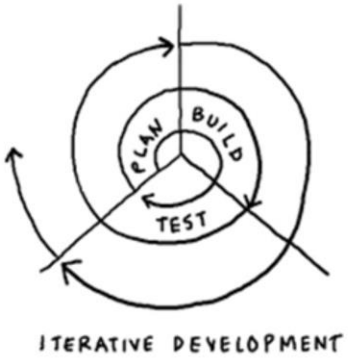
Practical Object Oriented Programming

James McNally (CLA & CLED)
Wiresmith Technology

Why I Care About This?



WIRESMITH Technology

Customers don't know what they want – need agile code

Reduces stress for developers!

Encapsulation promotes this by low coupling and high readability

There is a gulf between the "hows" and the "where and whens"

Why I Care About This?

When Do I Make A Class?

How Do I Make A Class?

WIRESMITH Technology

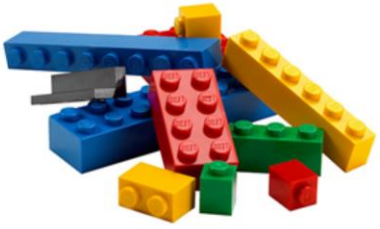
Customers don't know what they want – need agile code

Reduces stress for developers!

Encapsulation promotes this by low coupling and high readability

There is a gulf between the "hows" and the "where and whens"

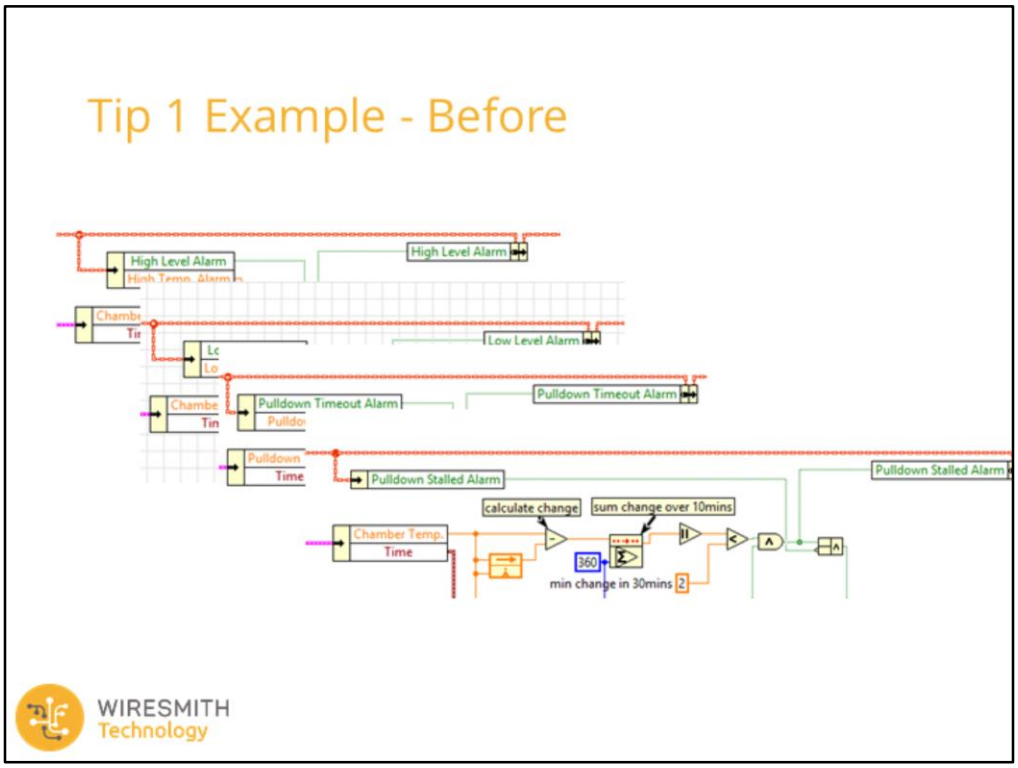
Tip 1 – You Can't Be Too Modular



WIRESMITH Technology

Easier to identify where to change code.
Less risk of side effects.

Code Examples:
Alarm & Alarm Engine?



I original had a piece of code with a simple alarm. If the alarm is either active or not.

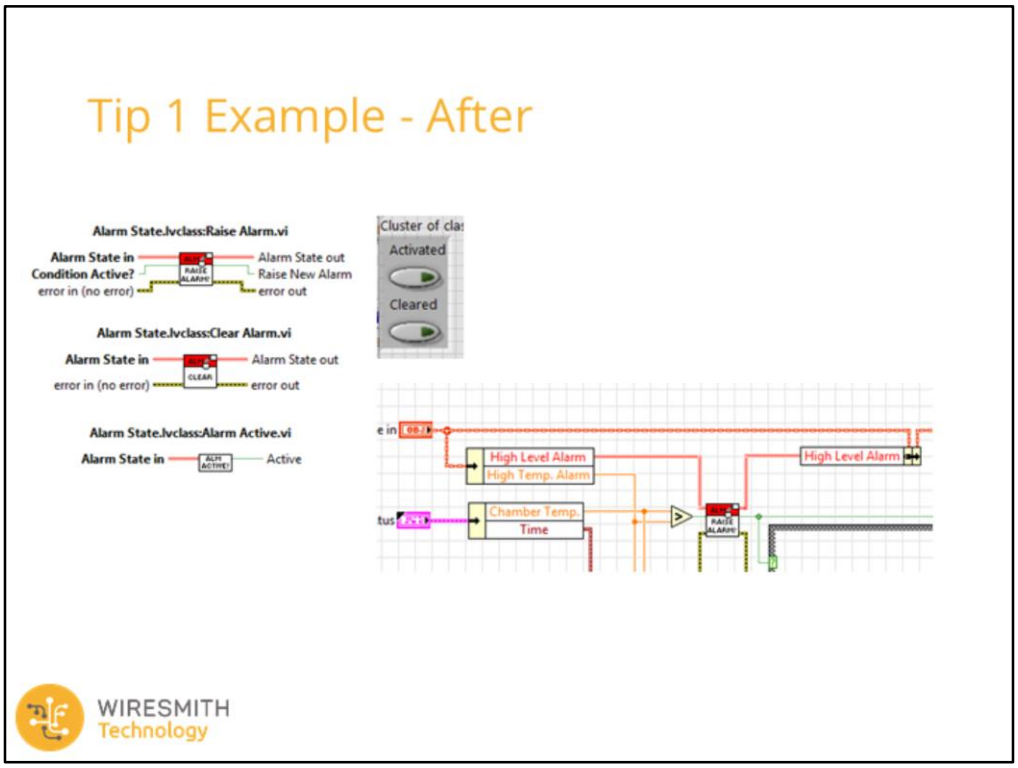
The compound arithmetic there is performing the rising edge detection that this is a new alarm.

But I wanted to add the option to clear an alarm – I can add a new Boolean called High Level Alarm Cleared and add logic for that.

BELL 1: I have a set of data (alarm and cleared) and some function related to that and only that (has it been cleared – then don't report the alarm)

I realised I had to do this in several places.

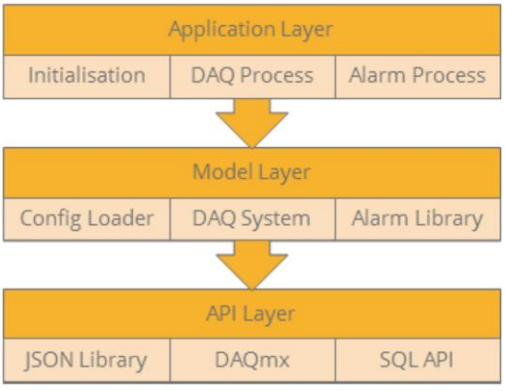
BELL 2: DRY



Rule: 7+-2 data fields
Rule: Interfaces should not let type-defs propagate too far

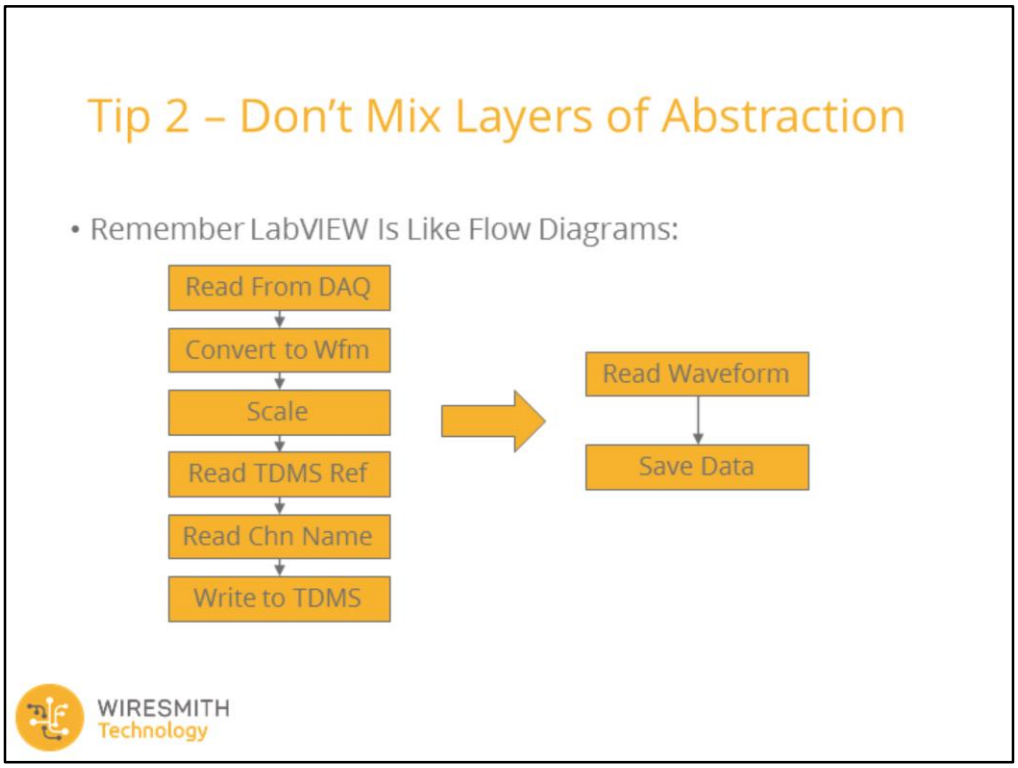
Tip 2 – Don't Mix Layers of Abstraction

- All elements in one subVI should be a similar level.



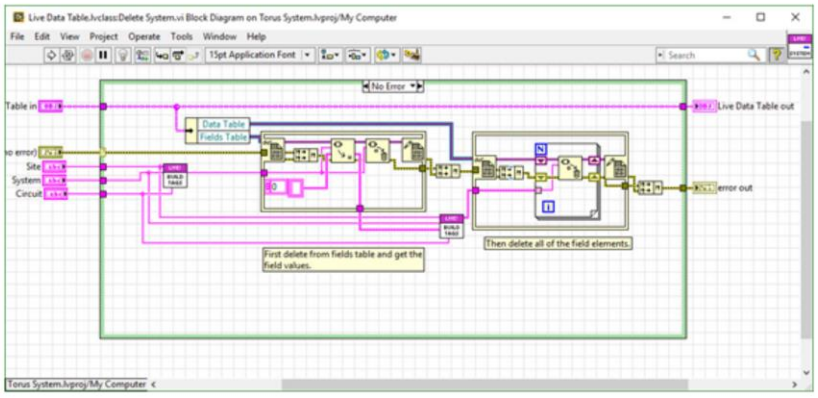
WIRESMITH Technology

Shouldn't see primitives next to high level code
Reduces mental burden
One technique initially is to consider discrete layers in your code
(I find this is support initially that eventually isn't required)



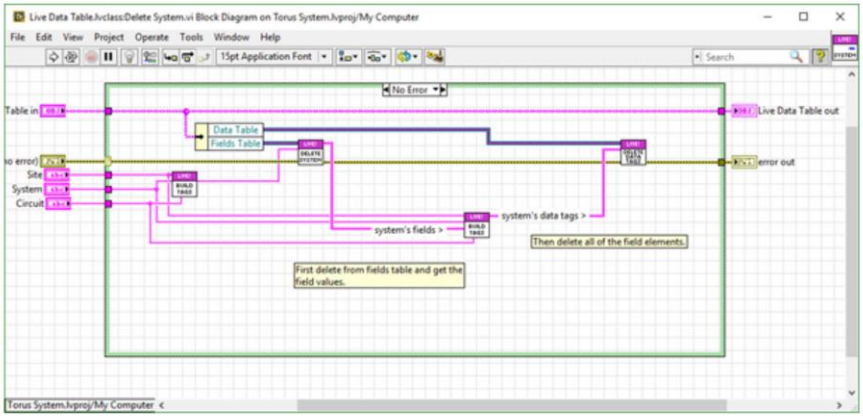
Shouldn't see primitives next to high level code
Reduces mental burden
One technique initially is to consider discrete layers in your code
(I find this is support initially that eventually isn't required)

Tip 2 Example - Before

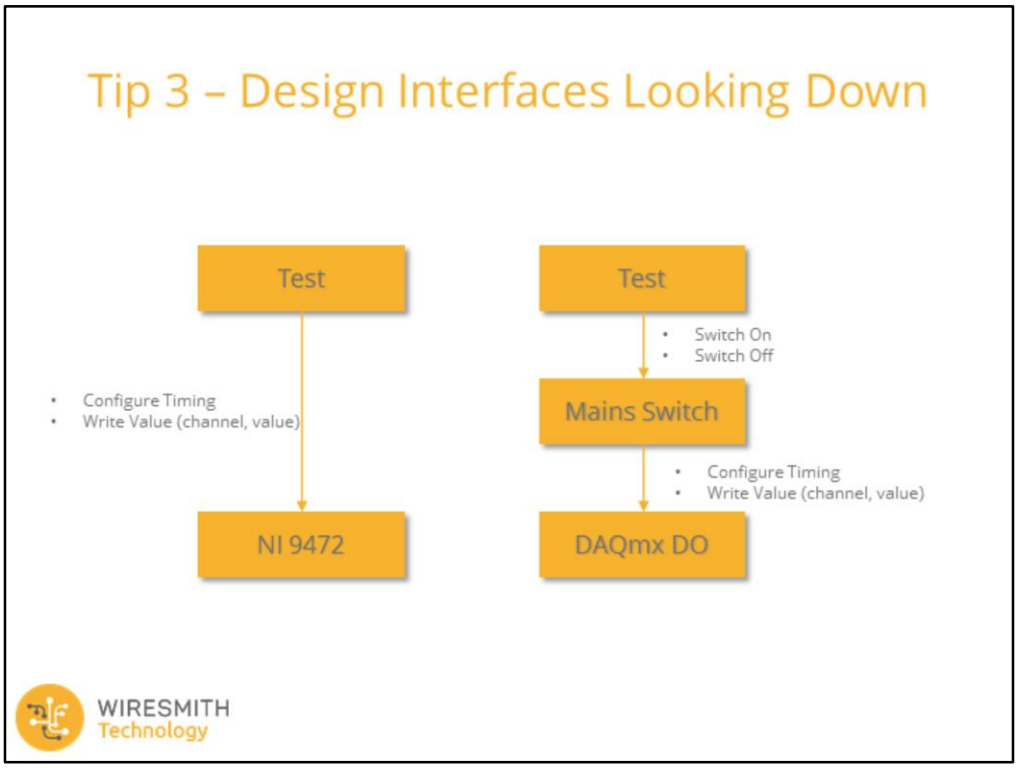


WIRESMITH Technology

Tip 2 Example - After



WIRESMITH Technology



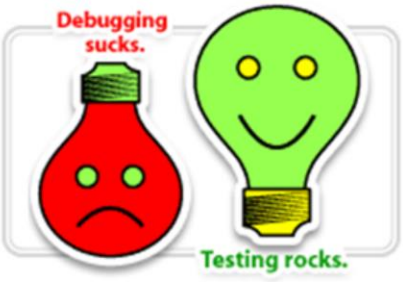
Improves readability in calling code
Reduces feature creep

Tip 4 – Refactor Constantly

- Courses & Blogs makes out that you can sit and design everything up front.
- This is NOT how it works in real life.
- Remember you see code when it is done, not the various iterations.
- Designs change over time – avoid flaccid scrum

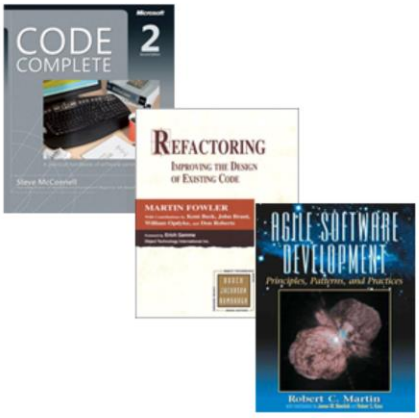
WIRESMITH
Technology

Tip 5 - Testing



WIRESMITH Technology

Tip 6 – Learn From Others



WIRESMITH Technology

- Using OO gives us a common language with other environments.
- If you can read 'C' style syntax there is loads of Java code demonstrating good practice.
- If you have code – share it – helps us all develop
- Reading List for this –
    - Code Complete
    - Refactoring
    - Agile – Uncle Bob
    - GoF?

Learn More From Me

https://devs.wiresmithtech.com/mlug-2016

- Twitter: @jamesmc86
- NI Forums: JamesMcN
- Feedback or Questions to james@wiresmithtech.com

WIRESMITH
Technology