Test & Measurement Solutions

How to turn Spaghetti code into a Wordclass Software Architecture

Speaker: Arnoud de Kuijper
Managing director T&M Solutions - Netherlands

NATIONAL INSTRUMENTS Gold Alliance Partner

Spaghetti code

"Spaghetti code is a pejorative term for source code that has a complex and tangled control structure."

http://en.wikipedia.org/wiki/Spaghetti_code

It is not only a term reserved for LabVIEW

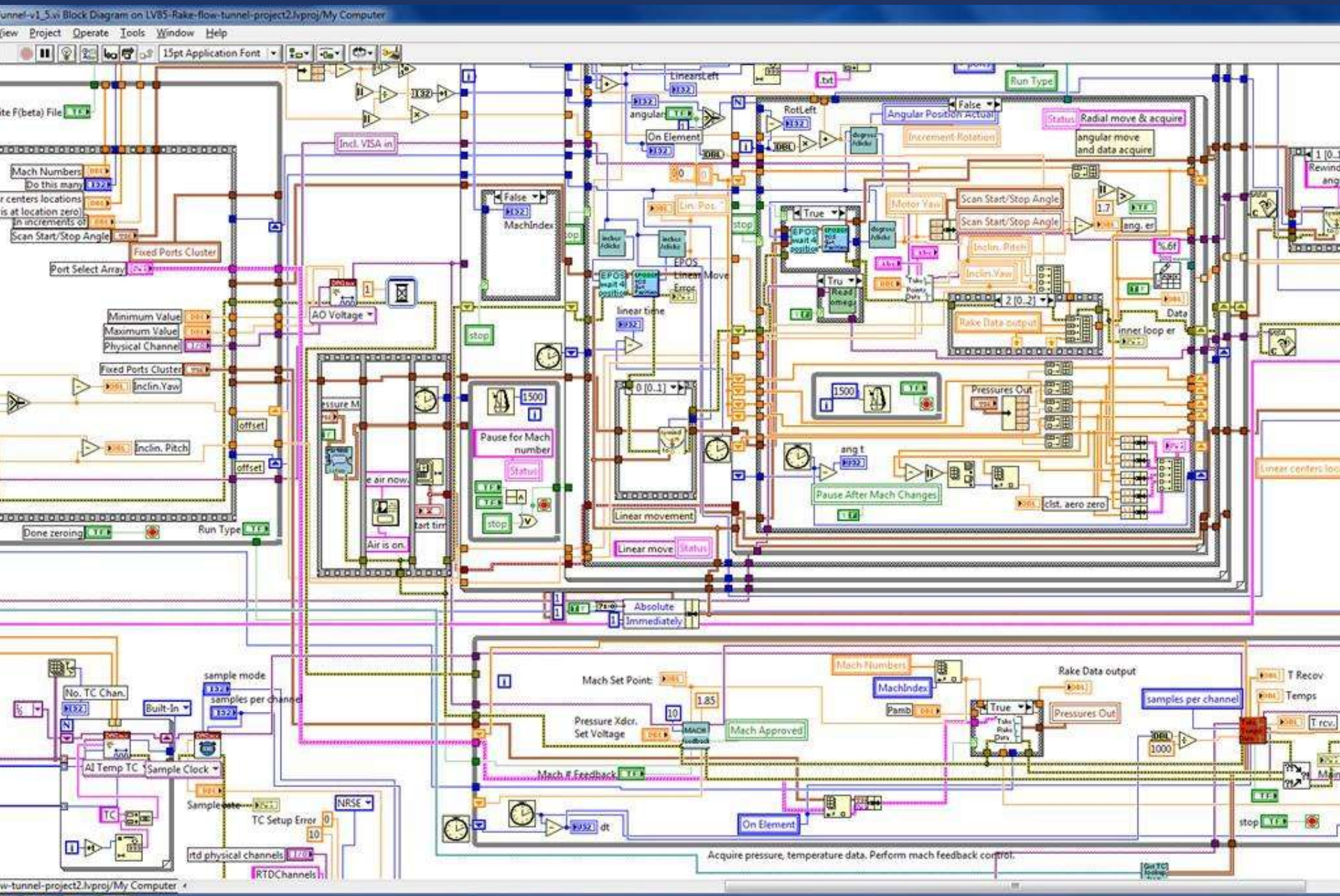
```
10 i = 0
20 i = i + 1
30 PRINT i; " squared = "; i * i
40 IF i >= 10 THEN GOTO 60
50 GOTO 20
60 PRINT "Program Completed."
70 END
```

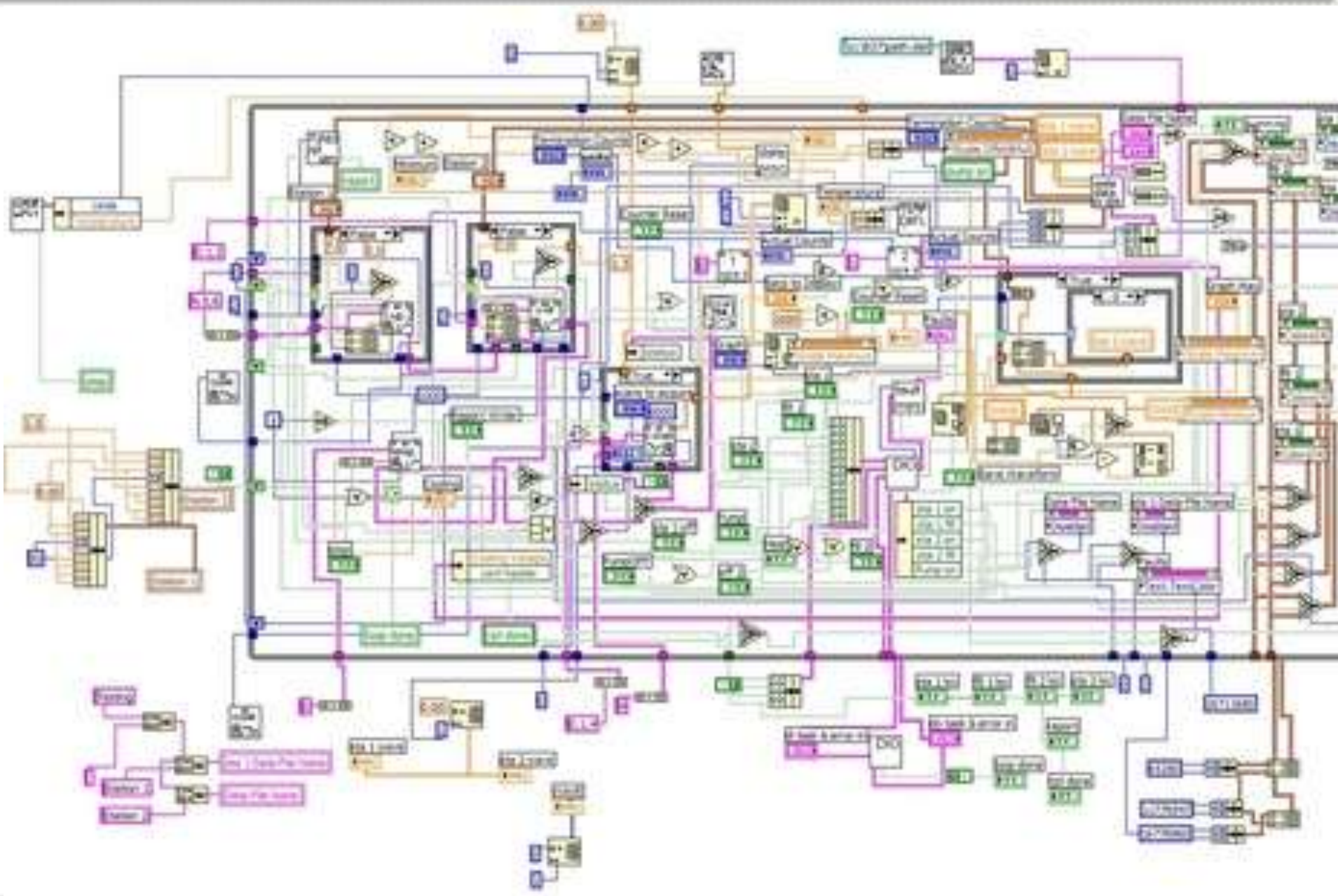
Here is the same code written in a structured programming style:

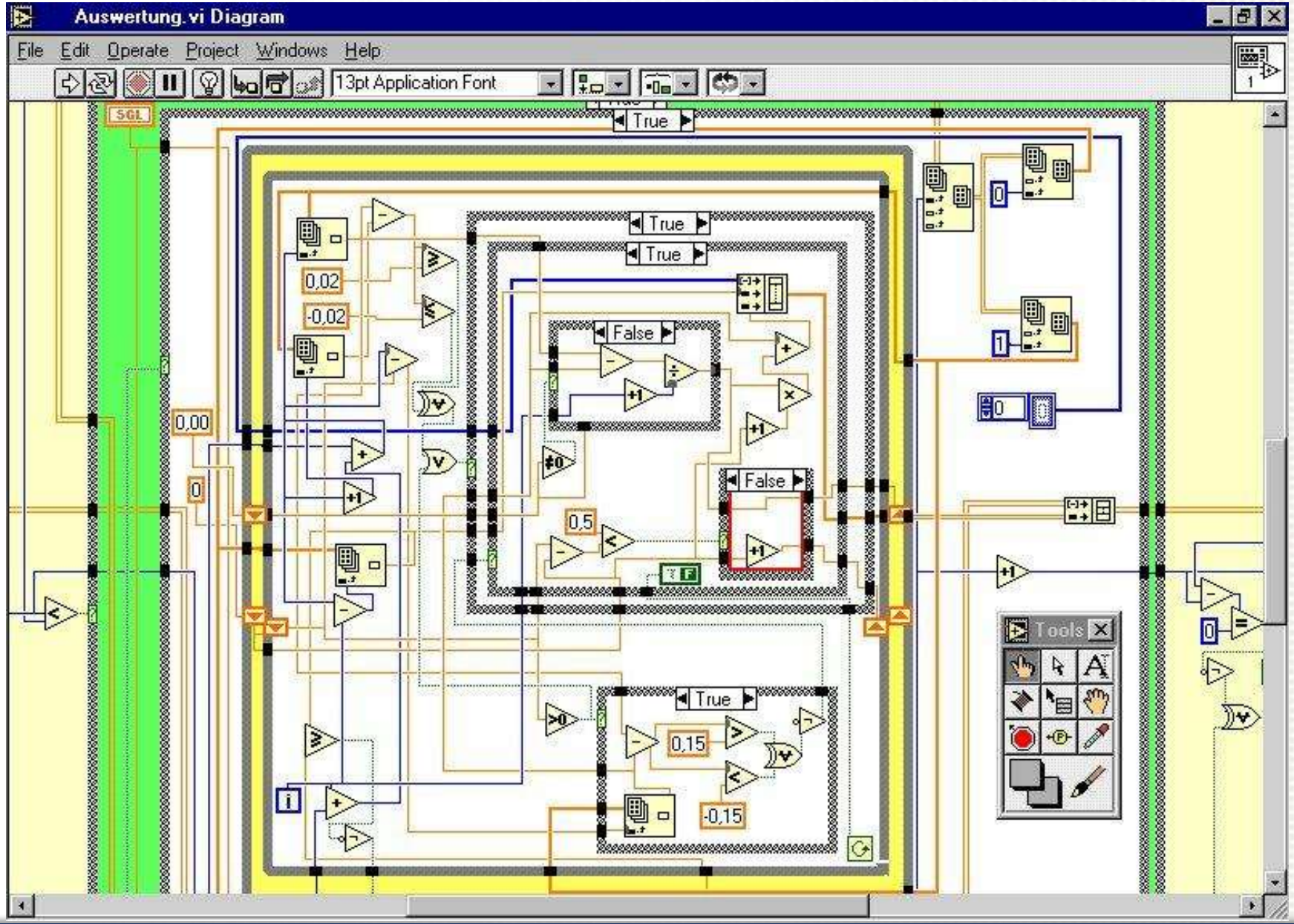
```
10 FOR i = 1 TO 10
20     PRINT i; " squared = "; i * i
30 NEXT i
40 PRINT "Program Completed."
50 END
```

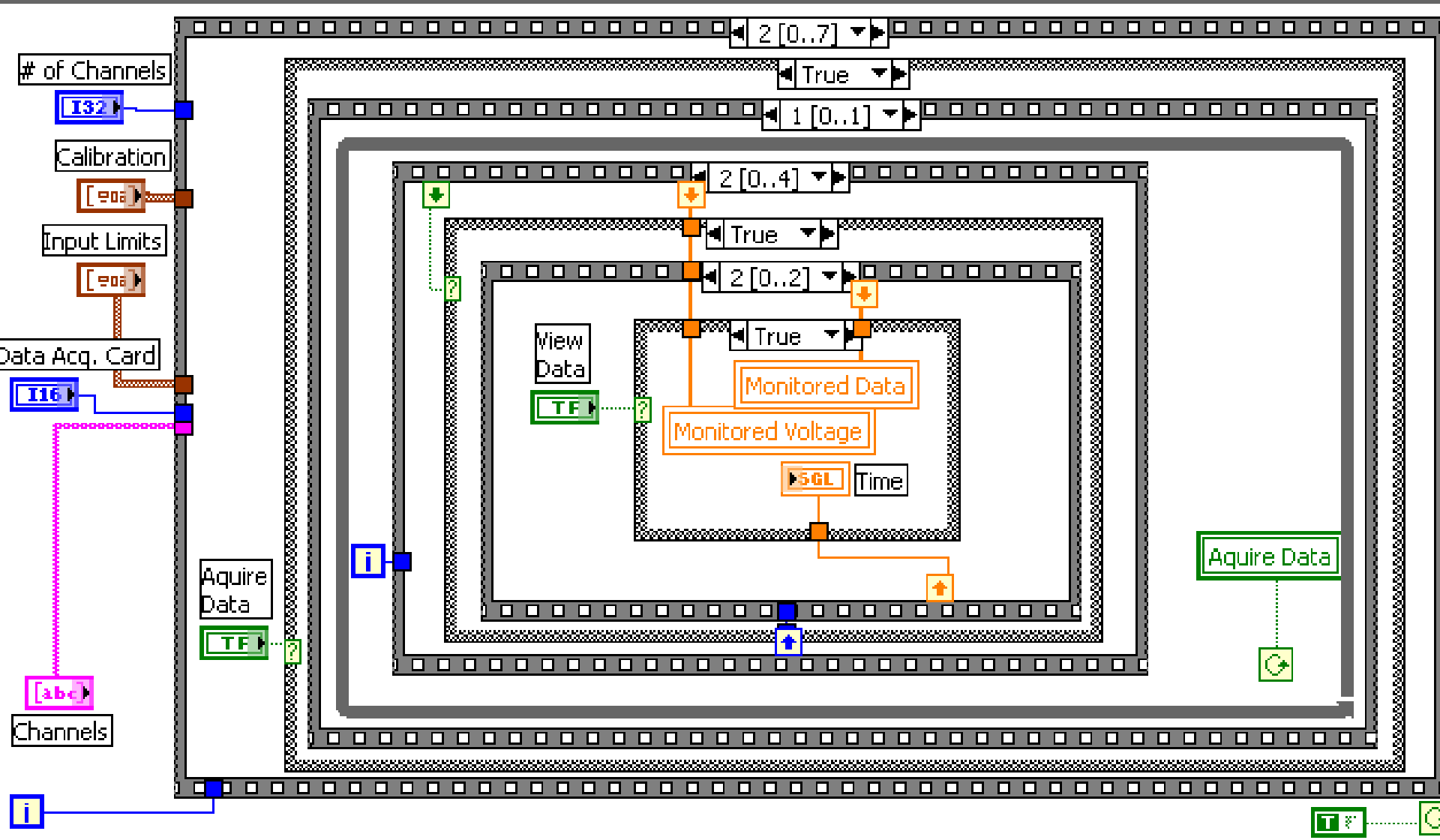
But LabVIEW has some beautifull examples

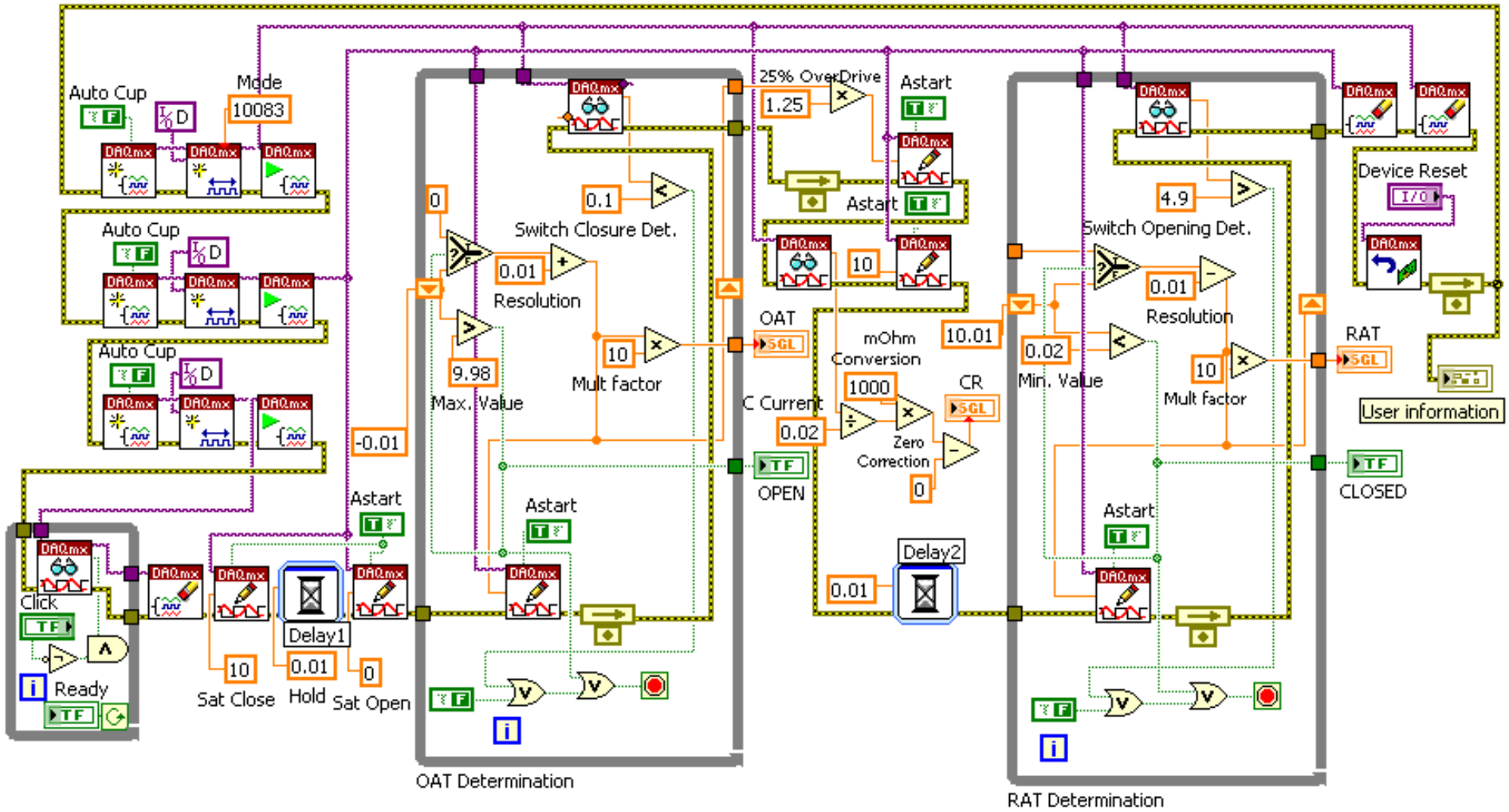
- Examples of spaghetti code

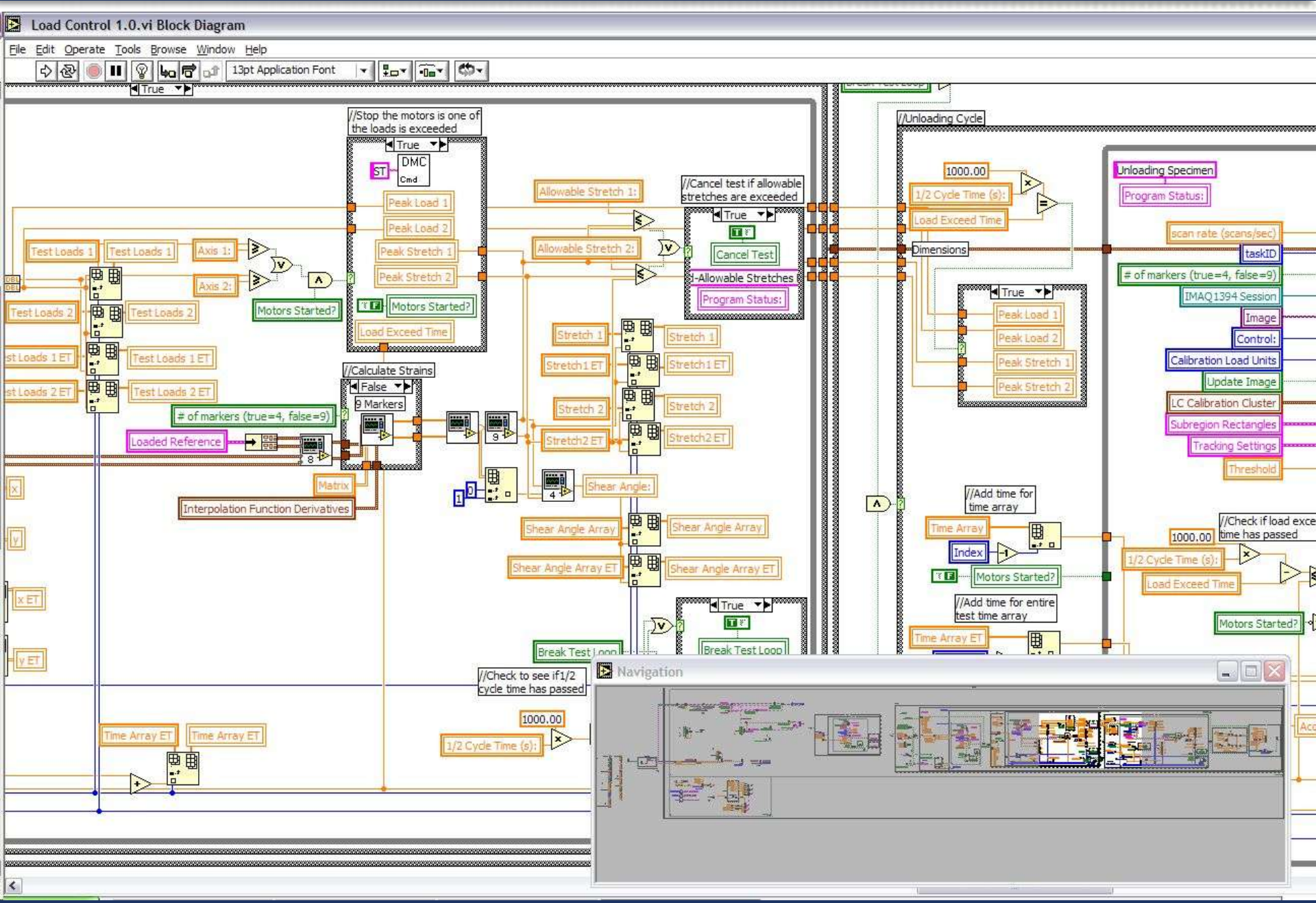


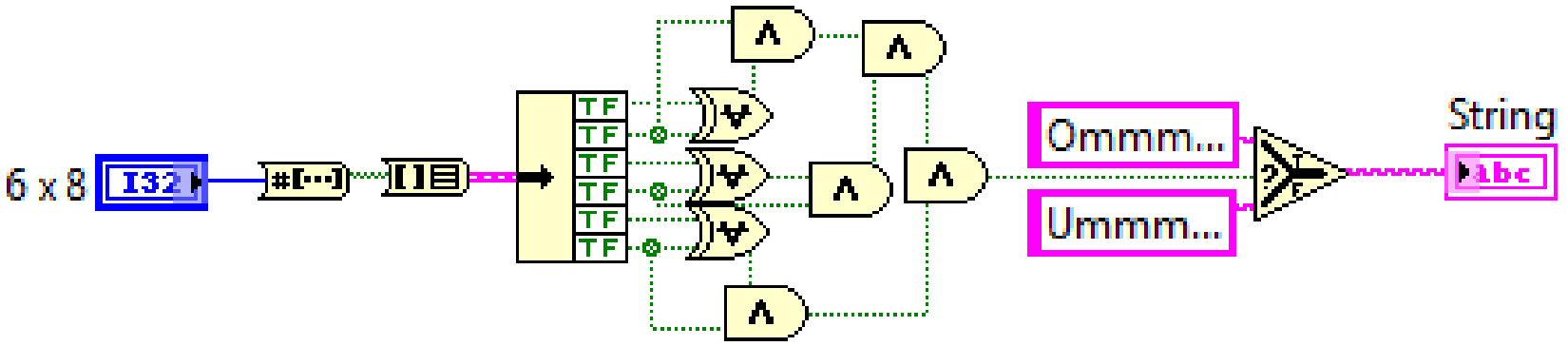




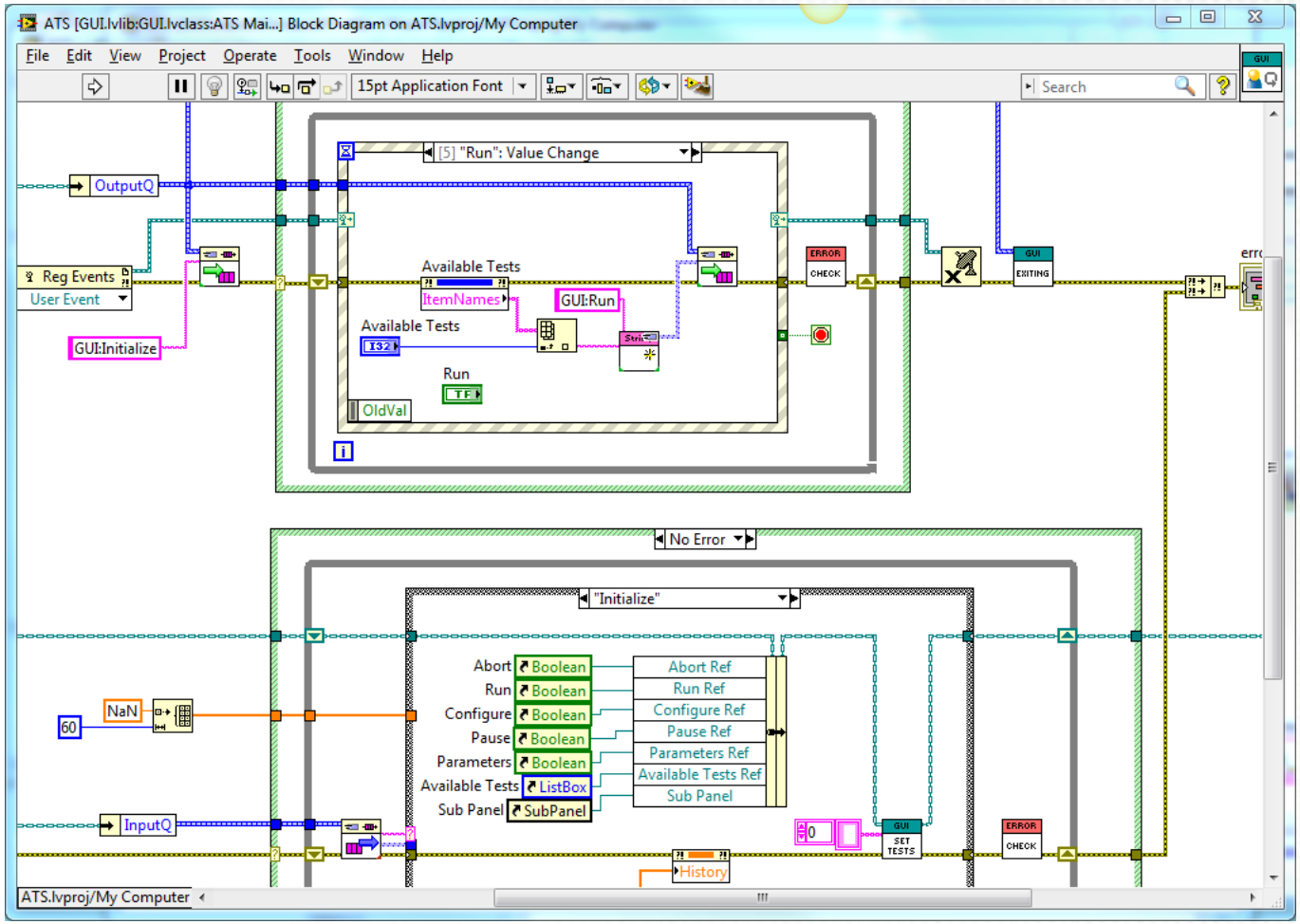




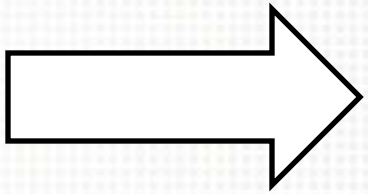




What is a good example?

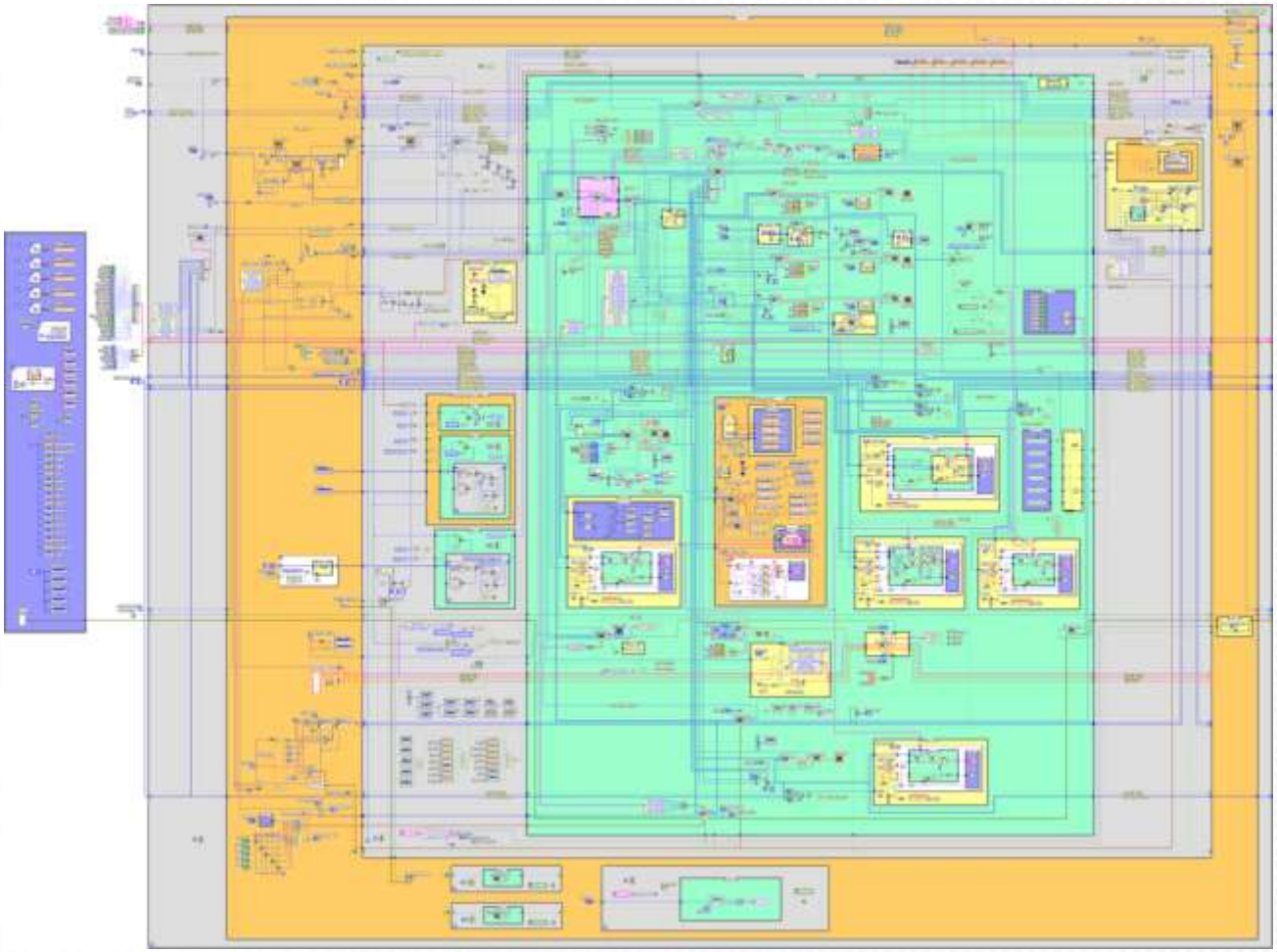


How to organize spaghetti?



5 Questions to ask yourself

1. Should I start refactoring?
2. Where to start?
3. How to start?
4. What to change first?
5. How to test?



My favorite example, but let's look deeper

http://www.biosemi.com/download_actiview.htm

1. Should I start refactoring?

A quick poll;

- Who would start from scratch and rebuild the application?
- Who would refactor this example into something better?

Should I start refactoring?

Take into consideration:

- Do I understand the behavior of the application?
- Is the “spaghetti” tangled in a large while loop or in sub-vi’s?
- Do the sub VI’s have a high cohesion?
- Does the code contain very specific and complex IP?
- Is the application sensitive to performance changes?

Our example

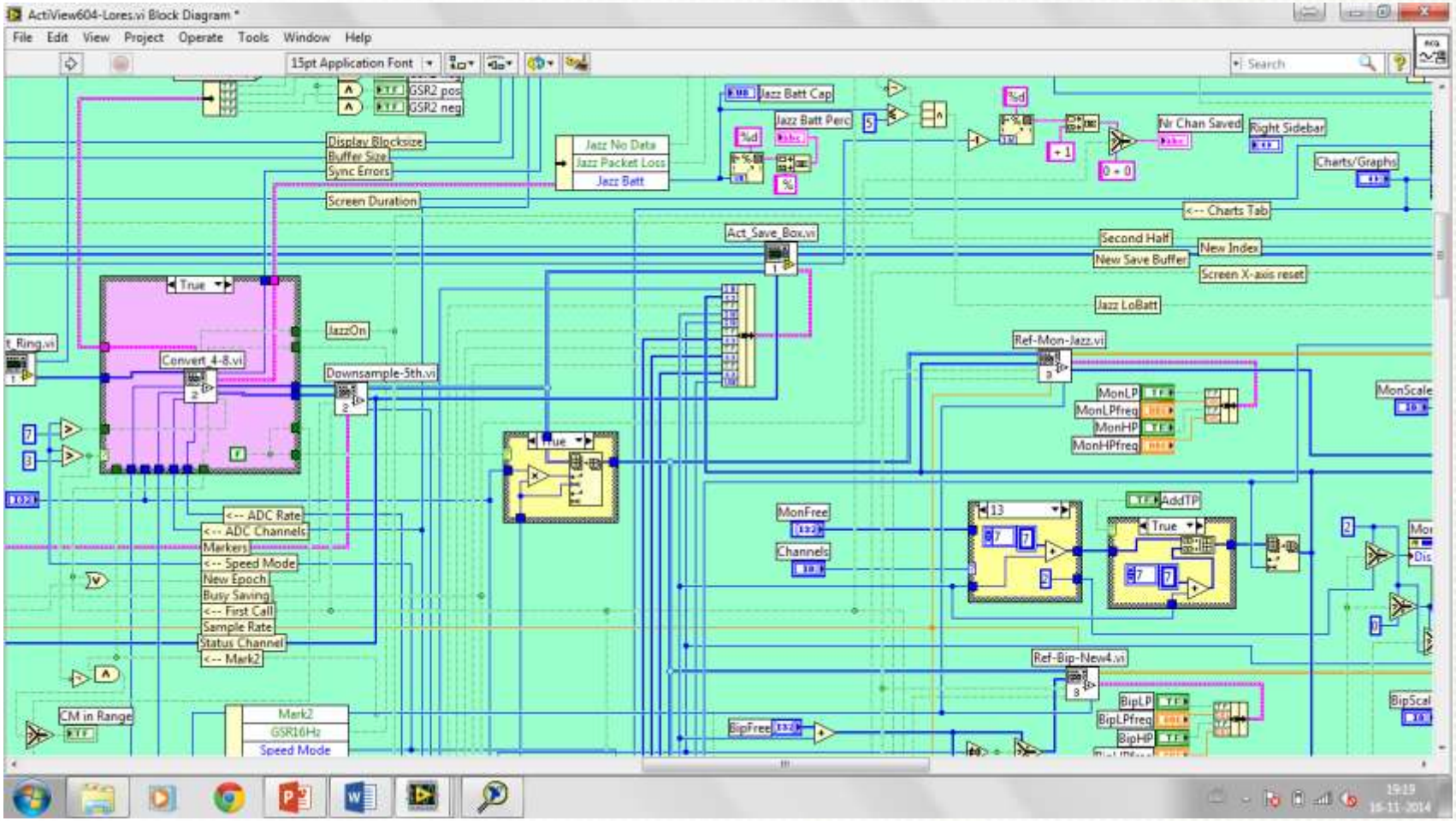
- Do I understand the behavior of the application? Partially
- Is the “spaghetti” tangled in a large while loop or in sub-vi’s? while-loop
- Do the sub VI’s have a high cohesion? Yes
- Does the code contain very specific and complex IP? Yes
- Is the application sensitive to performance changes? No

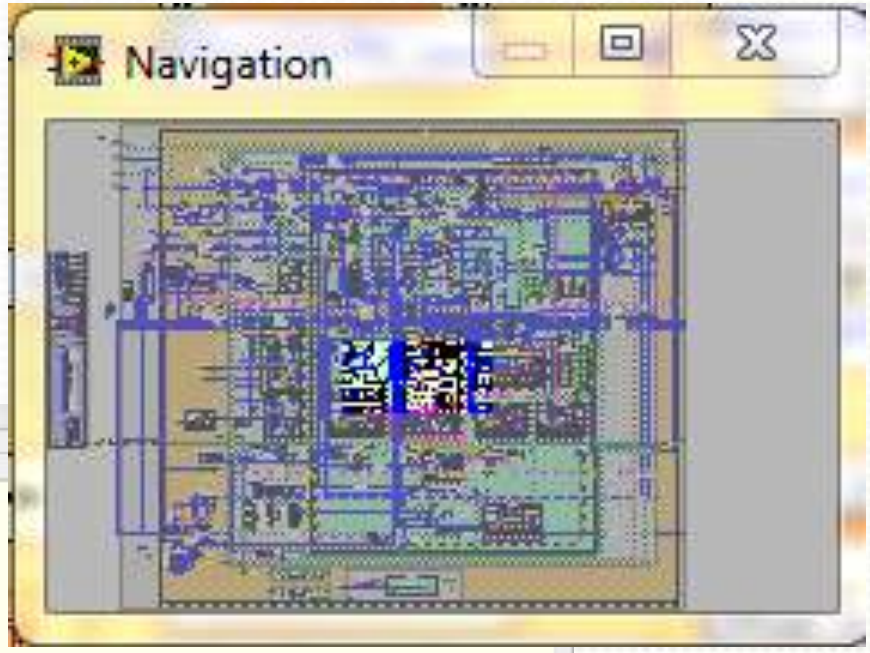
Should I start refactoring?

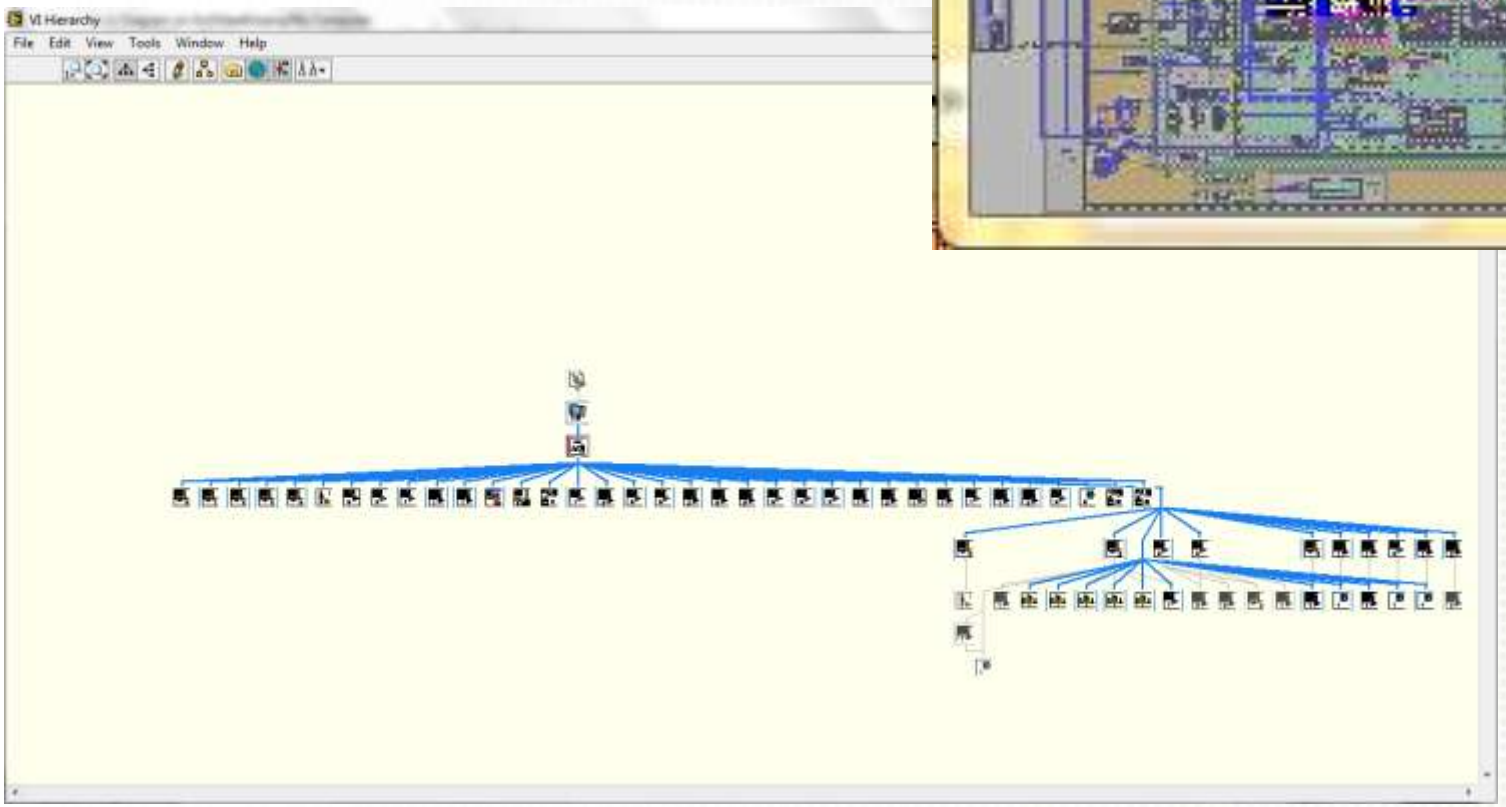
A note of caution:

- The not invented here syndrom! We often disregard what we have not build ourselves!
- Too strong style preferences, this is very subjective
- Refactoring can feel less rewarding, but could be more effective.

Where to start?







Different options available:

#Bottom – up approach:

Start creating subVi's of cohesive code.

Pro's

- Code becomes more readable as you go
- You can quickly identify re-use elements

Con's

- What is wise in selecting fragments to convert?
- How to choose a good interface?
- How to deal with dependencies suchs as locals and UI elements?

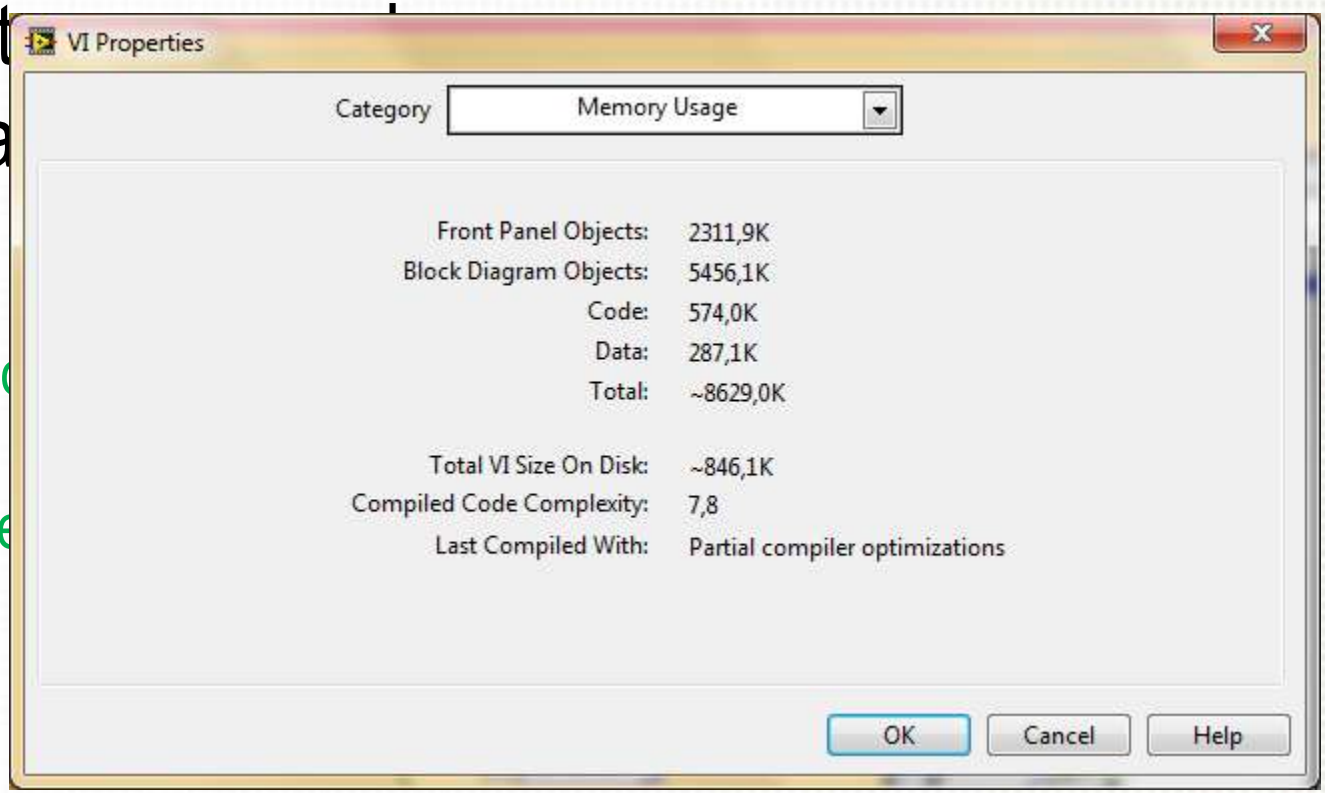
Different options available:

#Top - bott

Refa

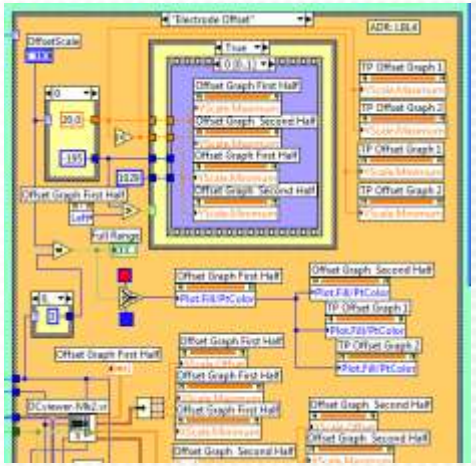
Pro's

- Don't need t
  interfaces.
- Can impleme
  insights
- All code still



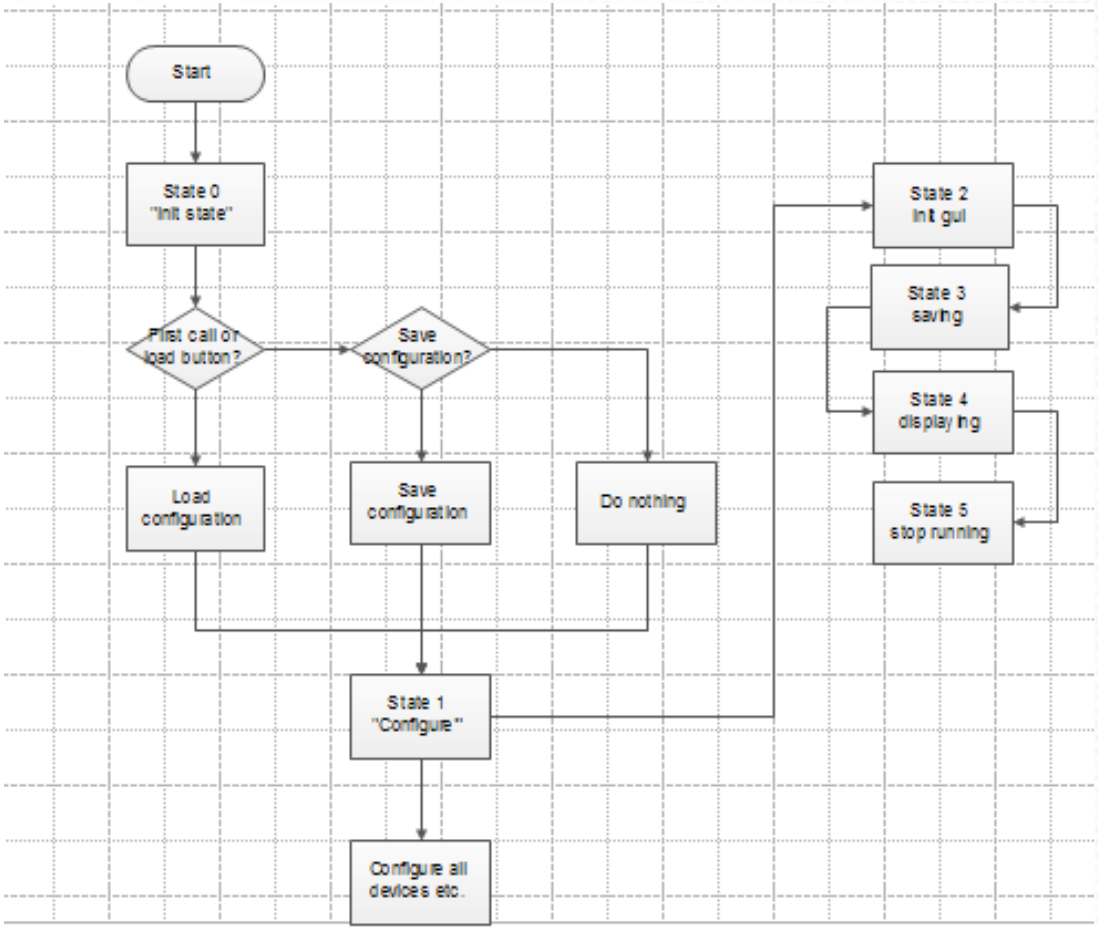
State machine with many states

- Small piece of code in all states (which normally would be in subVi's)
- Especially handy for UI intensive app's

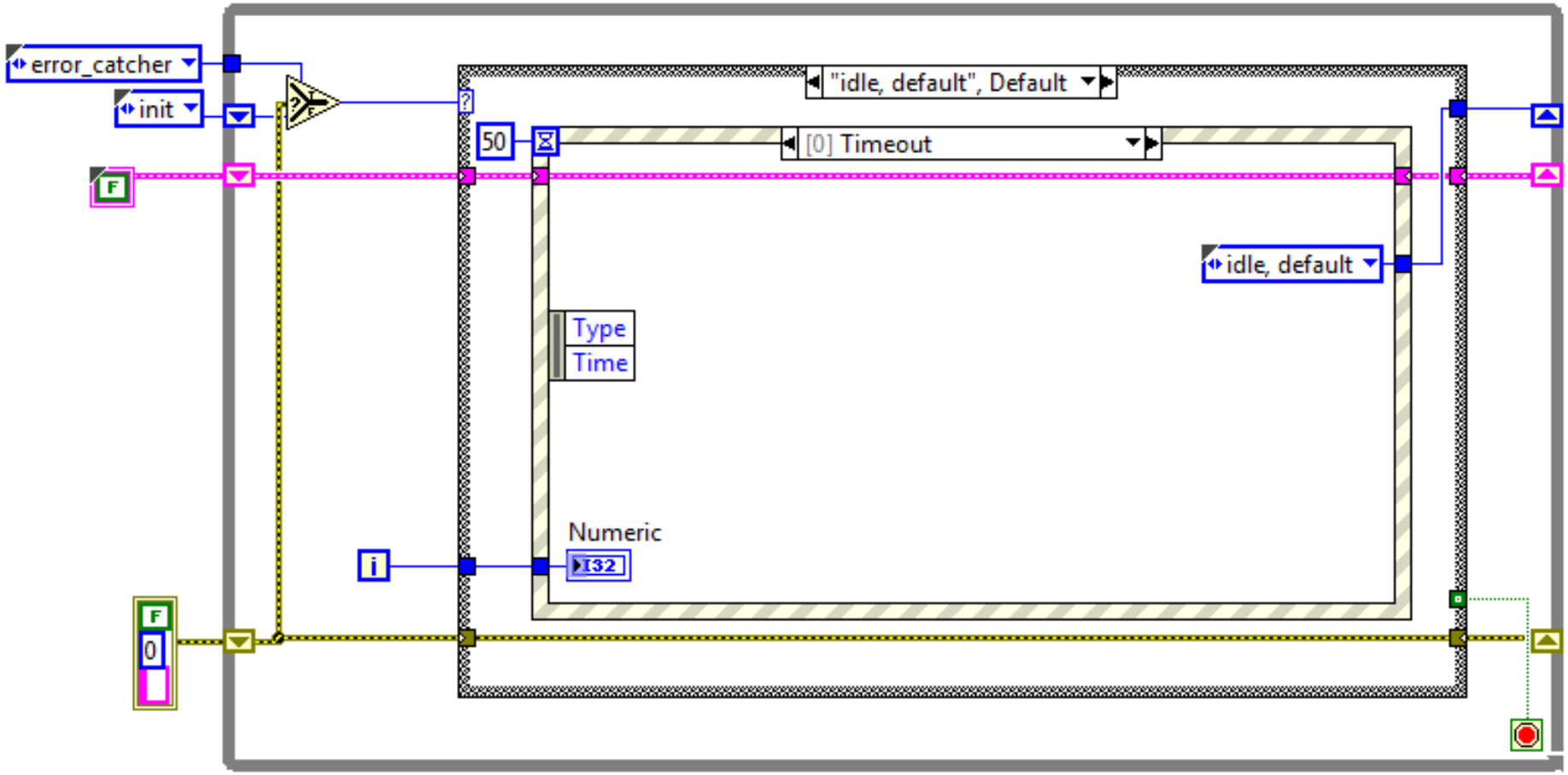


ADR: LBL4

Create state diagram

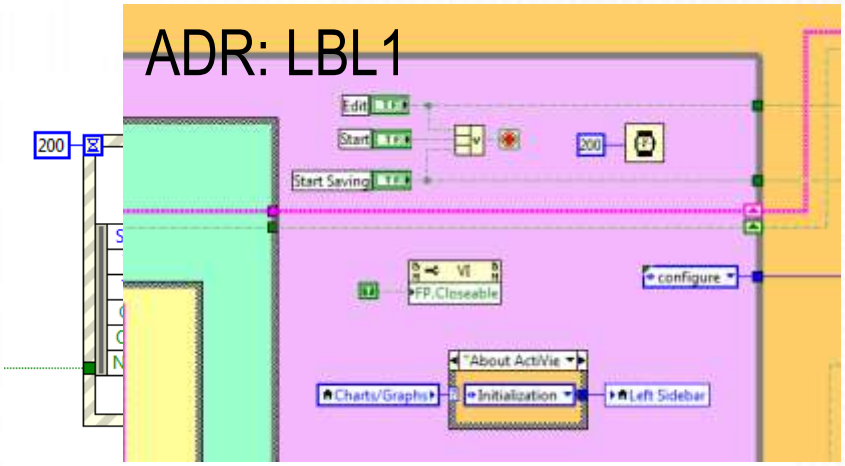


Template statemachine

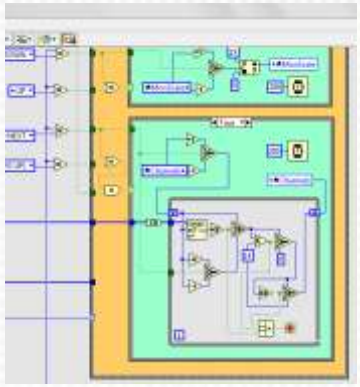


Event handler

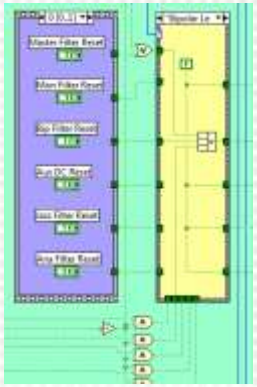
- Optimize structure in state selection:
- Improve performance because of counters!
- Convert polling structures



ADR: LBL3



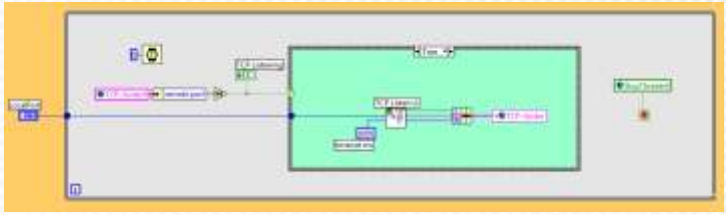
ADR: LBL2



#Task based

- Extract tasks and create separate engines.
  - Instrument communication
  - File streaming
  - UI handling (carefull!)
  - Network communication (e.g. TCP)

ADR: LBL5

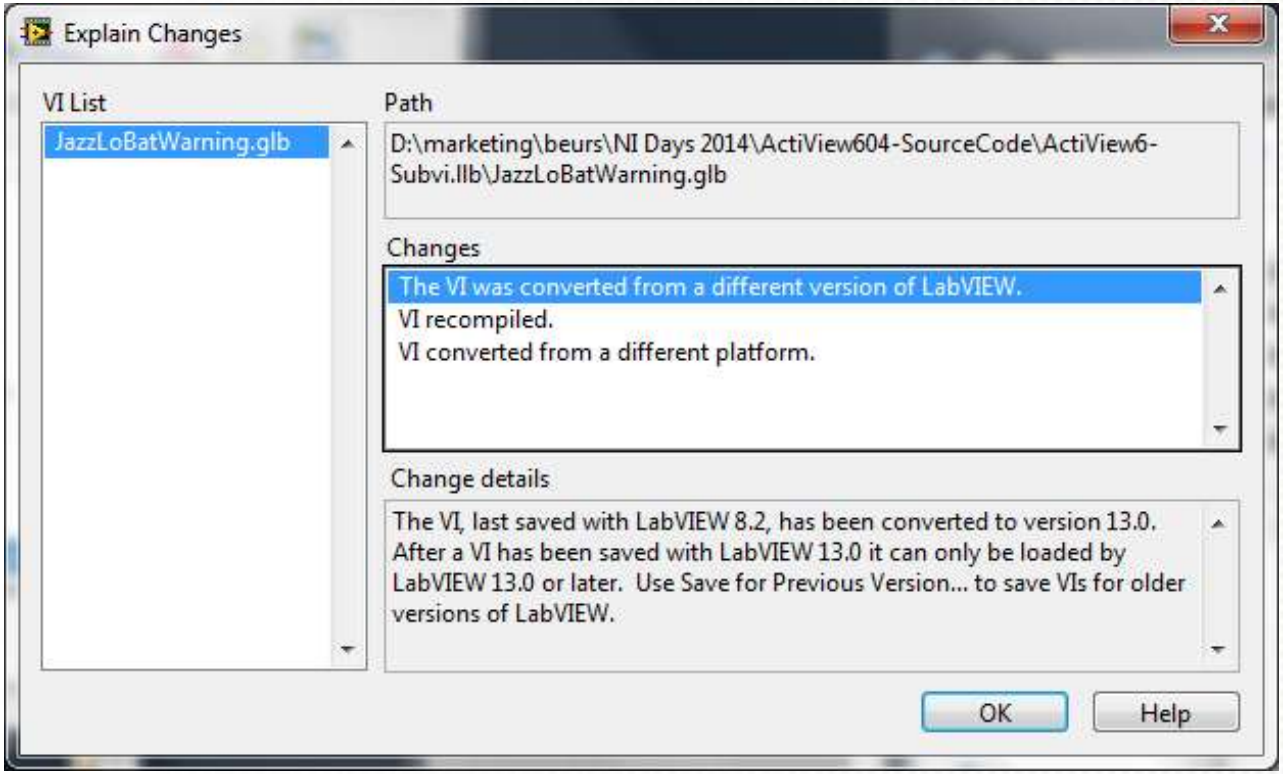


How to start?

- Make sure you have the right tools & setup:
  - Start with a source-code control (scc) version
  - Optionally with VMWare for operation system management
  - Create a LabVIEW project to organize VI's and find dependencies.
  - Convert llb's to folders (scc)

How to start?

- What version of LabVIEW?



How to start?

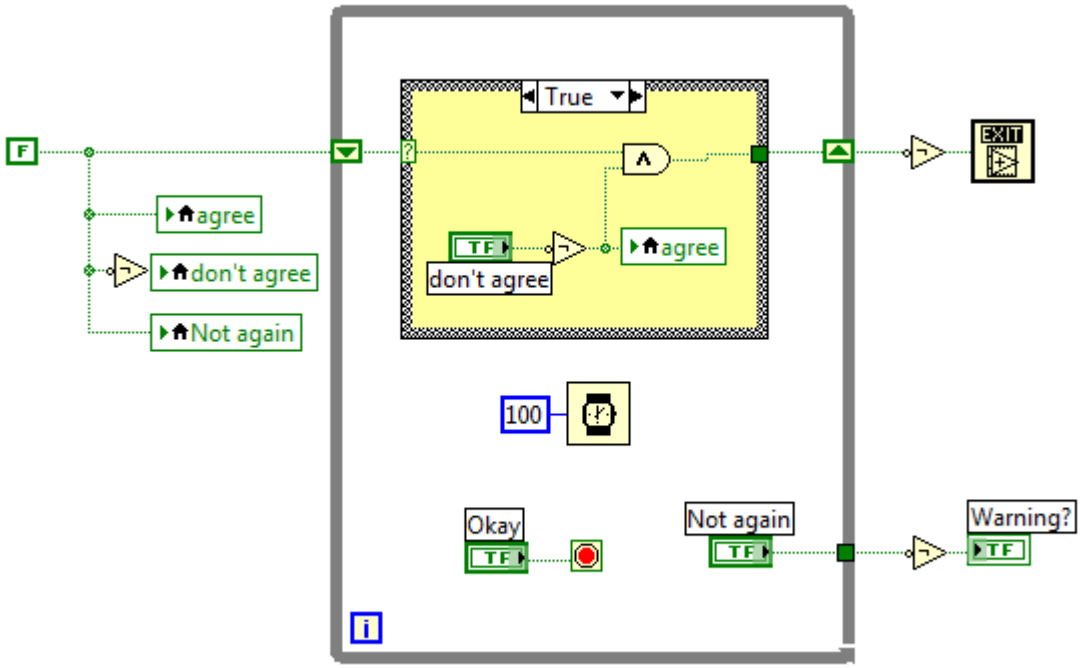
- Make sure you have the right tools & setup:
  - Commit every majer change to scc so you can revert.
  - Test, test, test…
  - Compare & diff van be very usefull

What to change first?

1. Change to selected platform (DEV, OS)
2. Improve structures without changing functions
3. Split-up code into state's with cohesion
4. Create subVI's for states where size is limiting
5. Create subVI's for re-use and replace where necessary.
6. Build unit tests

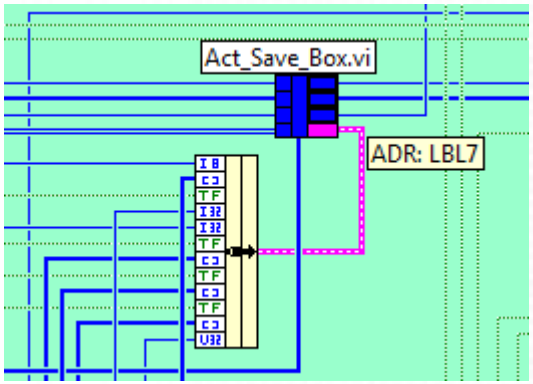
You can always update independend VI's

ADR: LBL9



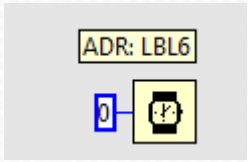
What to change first?

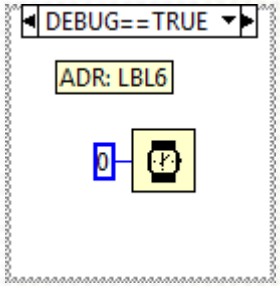
- 8. Remove code that is unneccesary
- 9. Break & rebuild connectors and interfaces



ADR: LBL7

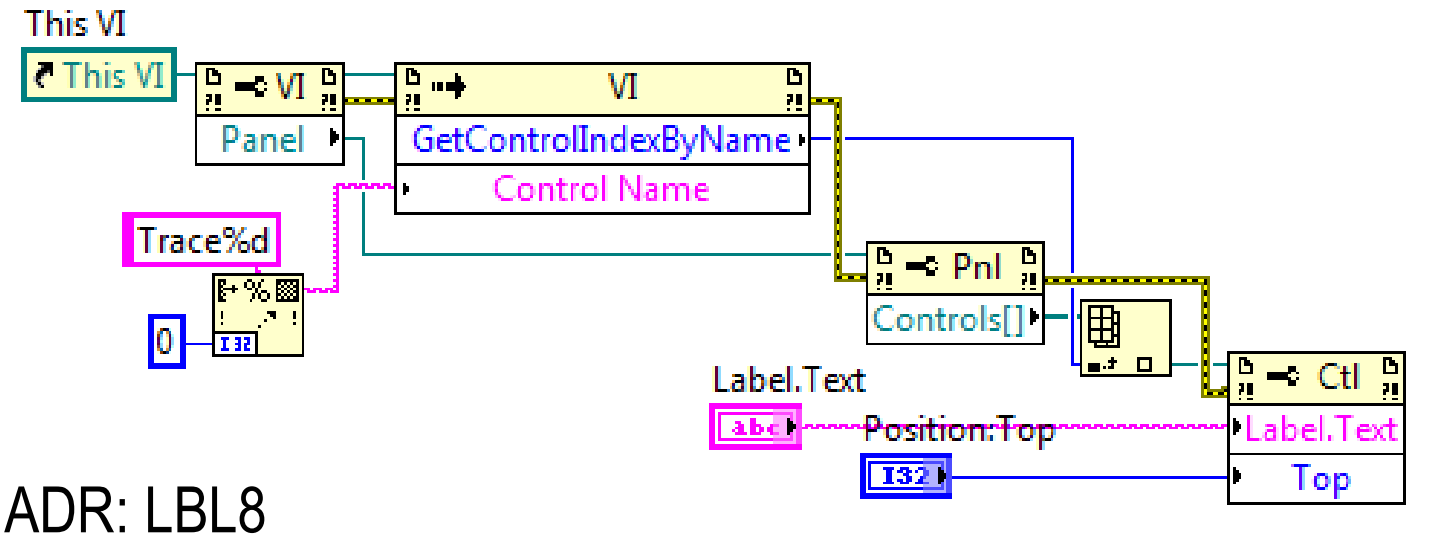
ADR: LBL6





What to change first?

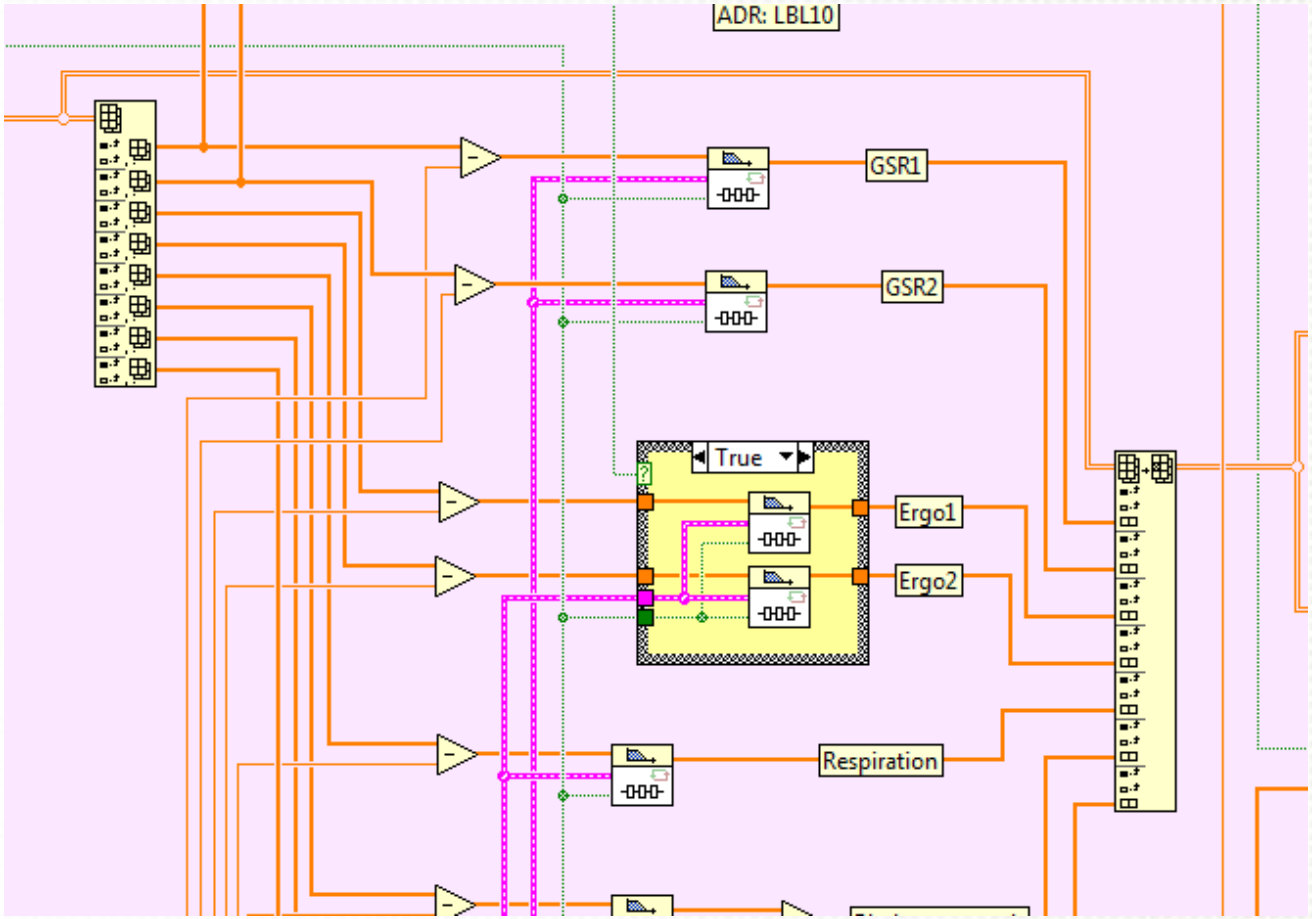
- 10. Optimize code for maintainability & scalability



ADR: LBL8

Apply new techniques

ADR: LBL10



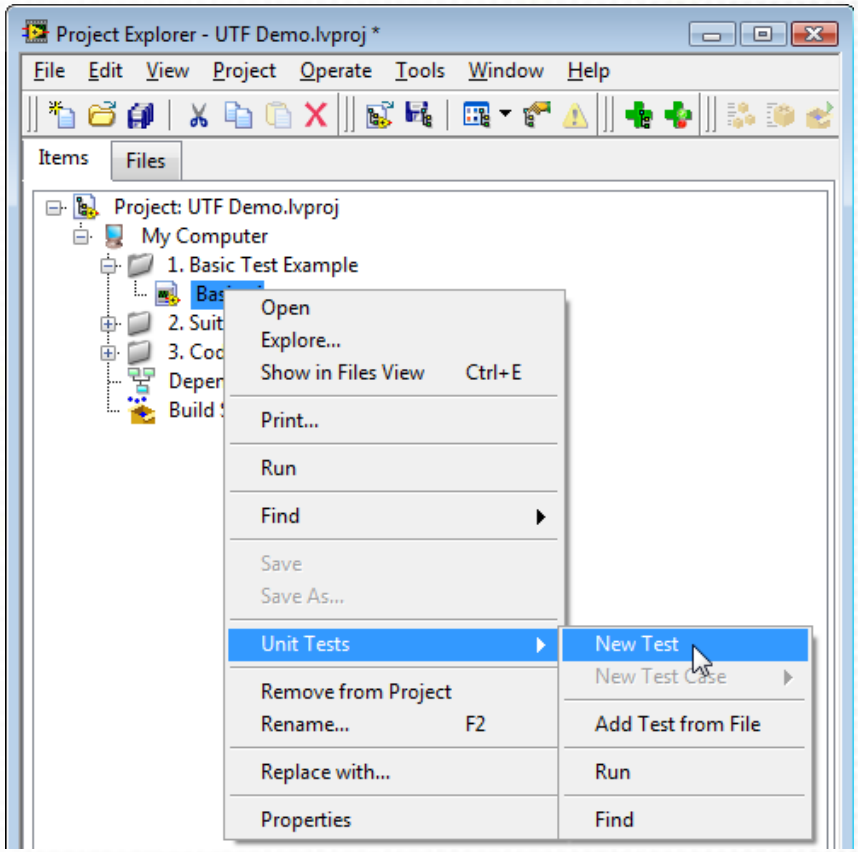
And Last

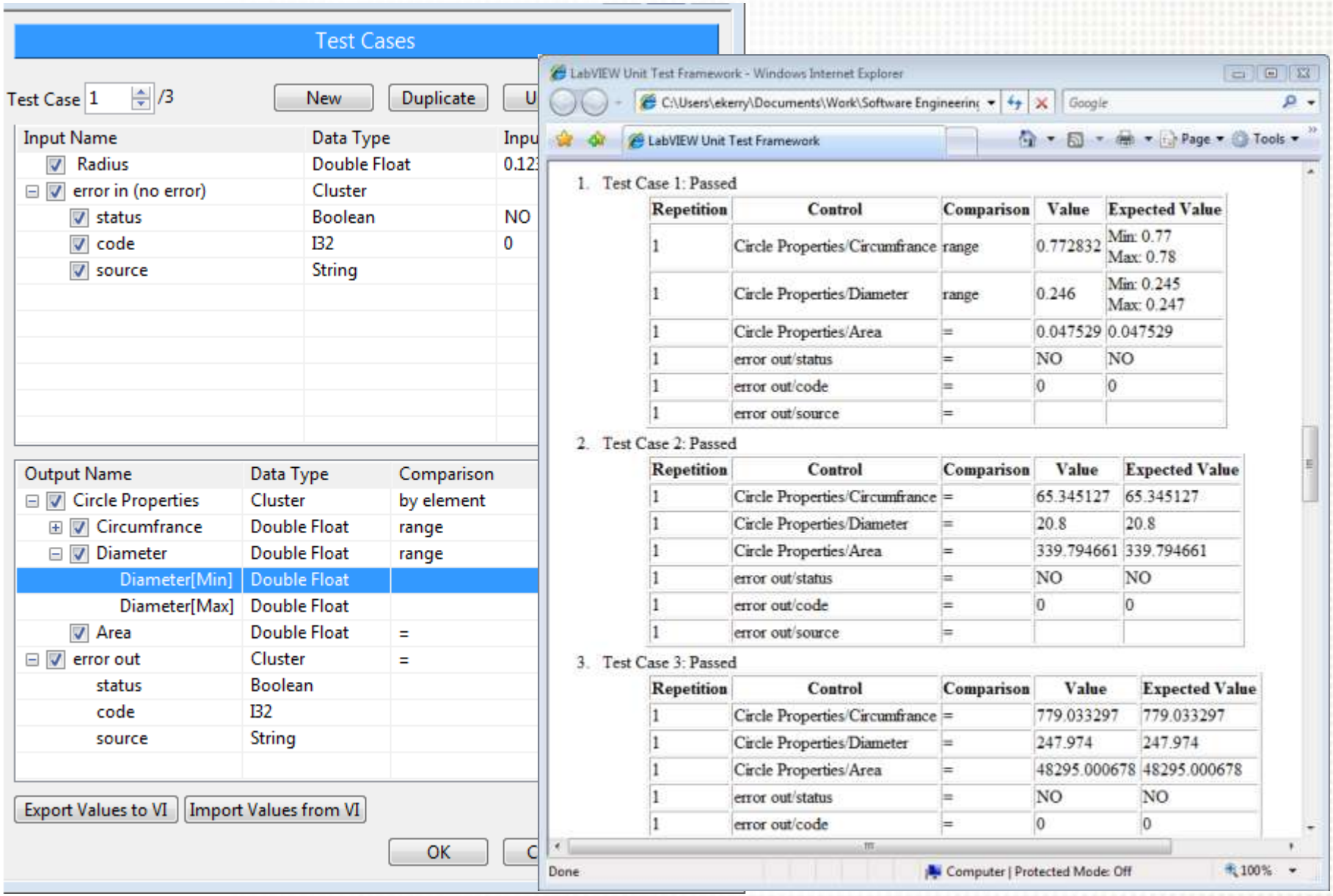
- 11. Rename global variables
- 12. Change implementation of alghorithms
- 13. Improvements in speed and UI

What & How to test

- ADR: LBL11

http://www.ni.com/white-paper/8082/en/





1. Test Case 1: Passed

Repetition	Control	Comparison	Value	Expected Value
1	Circle Properties/Circumfrance	range	0.772832	Min: 0.77 Max: 0.78
1	Circle Properties/Diameter	range	0.246	Min: 0.245 Max: 0.247
1	Circle Properties/Area	=	0.047529	0.047529
1	error out/status	=	NO	NO
1	error out/code	=	0	0
1	error out/source	=		

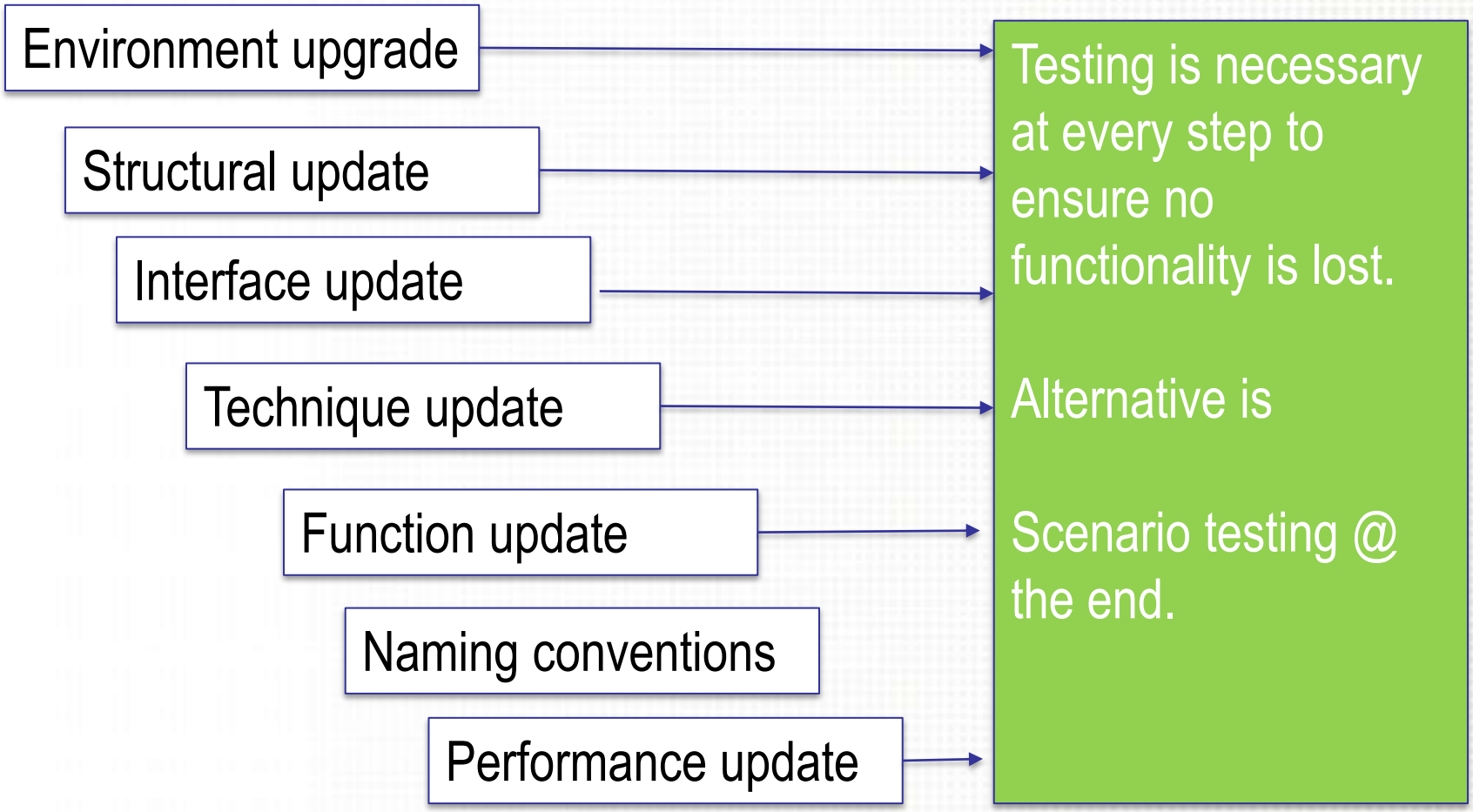
2. Test Case 2: Passed

Repetition	Control	Comparison	Value	Expected Value
1	Circle Properties/Circumfrance	=	65.345127	65.345127
1	Circle Properties/Diameter	=	20.8	20.8
1	Circle Properties/Area	=	339.794661	339.794661
1	error out/status	=	NO	NO
1	error out/code	=	0	0
1	error out/source	=		

3. Test Case 3: Passed

Repetition	Control	Comparison	Value	Expected Value
1	Circle Properties/Circumfrance	=	779.033297	779.033297
1	Circle Properties/Diameter	=	247.974	247.974
1	Circle Properties/Area	=	48295.000678	48295.000678
1	error out/status	=	NO	NO
1	error out/code	=	0	0

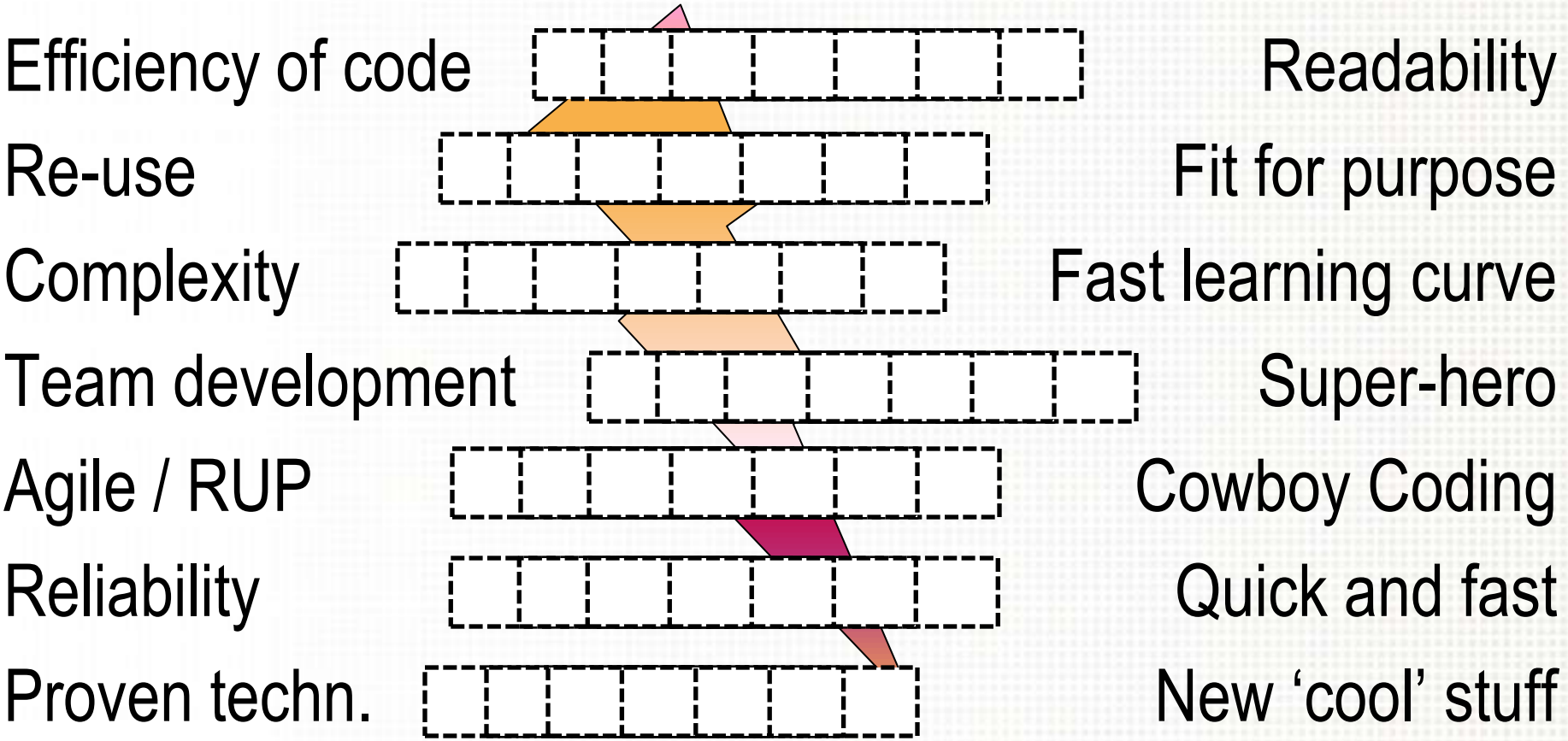
Summary



Questions?

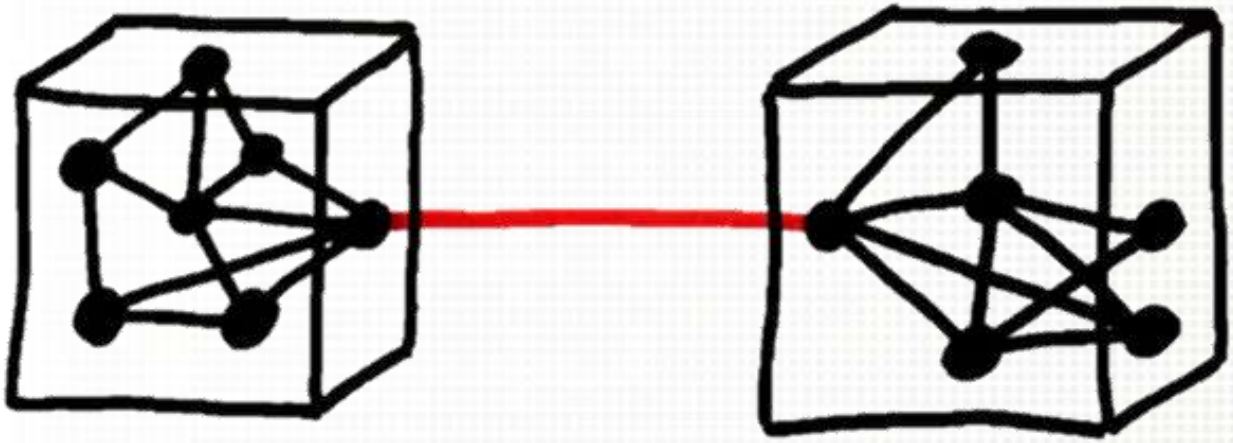
Supportive slides for answering questions…

Attitude – Balancing forces



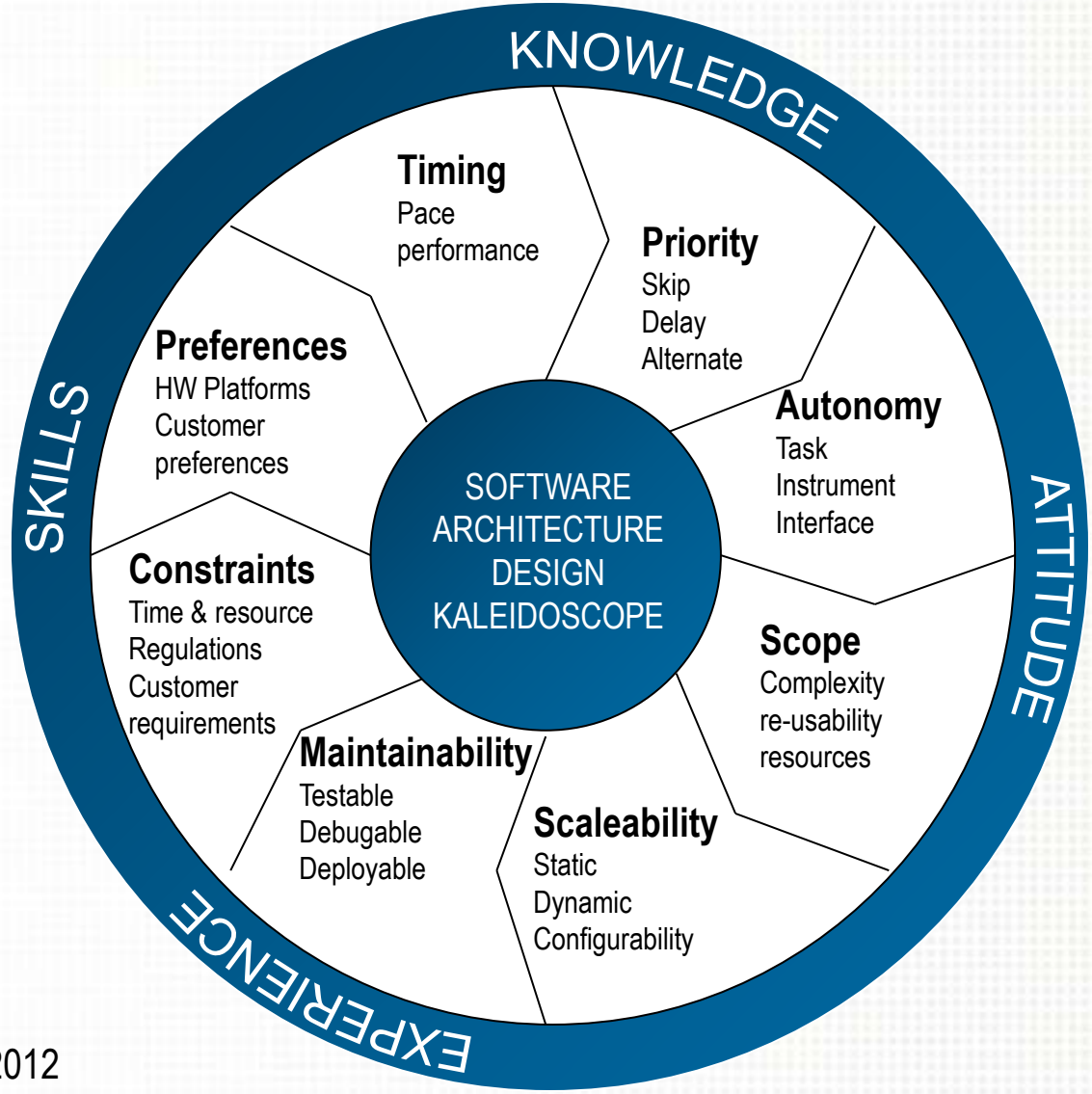
Experience

COUPLING AND COHESION



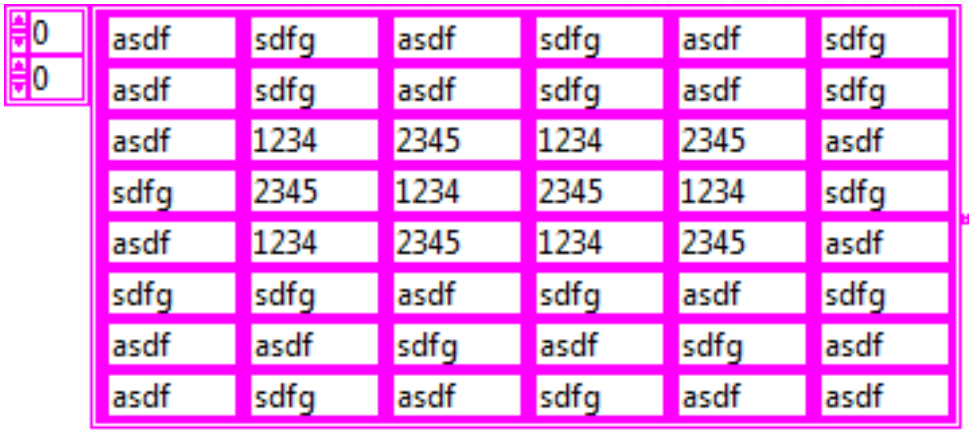
http://www.planetgeek.ch/2011/07/08/presentation-agile-code-design-how-to-keep-your-code-flexible/

DESIGN KALEIDOSCOPE



Question about testing…

- **Find the Location of the First Number in 2D String Array**



0						
0						
	asdf	sdfg	asdf	sdfg	asdf	sdfg
	asdf	sdfg	asdf	sdfg	asdf	sdfg
	asdf	1234	2345	1234	2345	asdf
	sdfg	2345	1234	2345	1234	sdfg
	asdf	1234	2345	1234	2345	asdf
	sdfg	sdfg	asdf	sdfg	asdf	sdfg
	asdf	asdf	sdfg	asdf	sdfg	asdf
	asdf	sdfg	asdf	sdfg	asdf	asdf