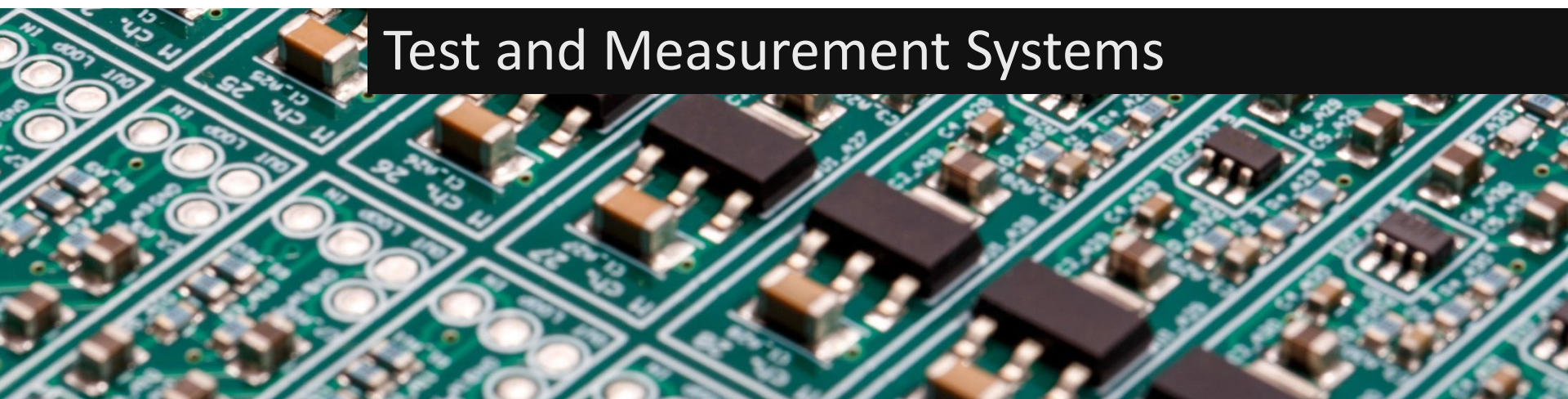




ADDQ

Question Everything!TM



Test and Measurement Systems

Presentation

- **Mattias Ericsson**

- LabVIEW developer ~17years

- CLA

- LabVIEW Partner Program

- QRM

- www.addq.se/qrm

- G# Framework

- Free, open source tool

- LabVIEW Add-On of the Year for Community 2011

- www.ni.com/labviewtools

- www.addq.se/gsharp



To Platform...

Or

Not to Platform

Platform



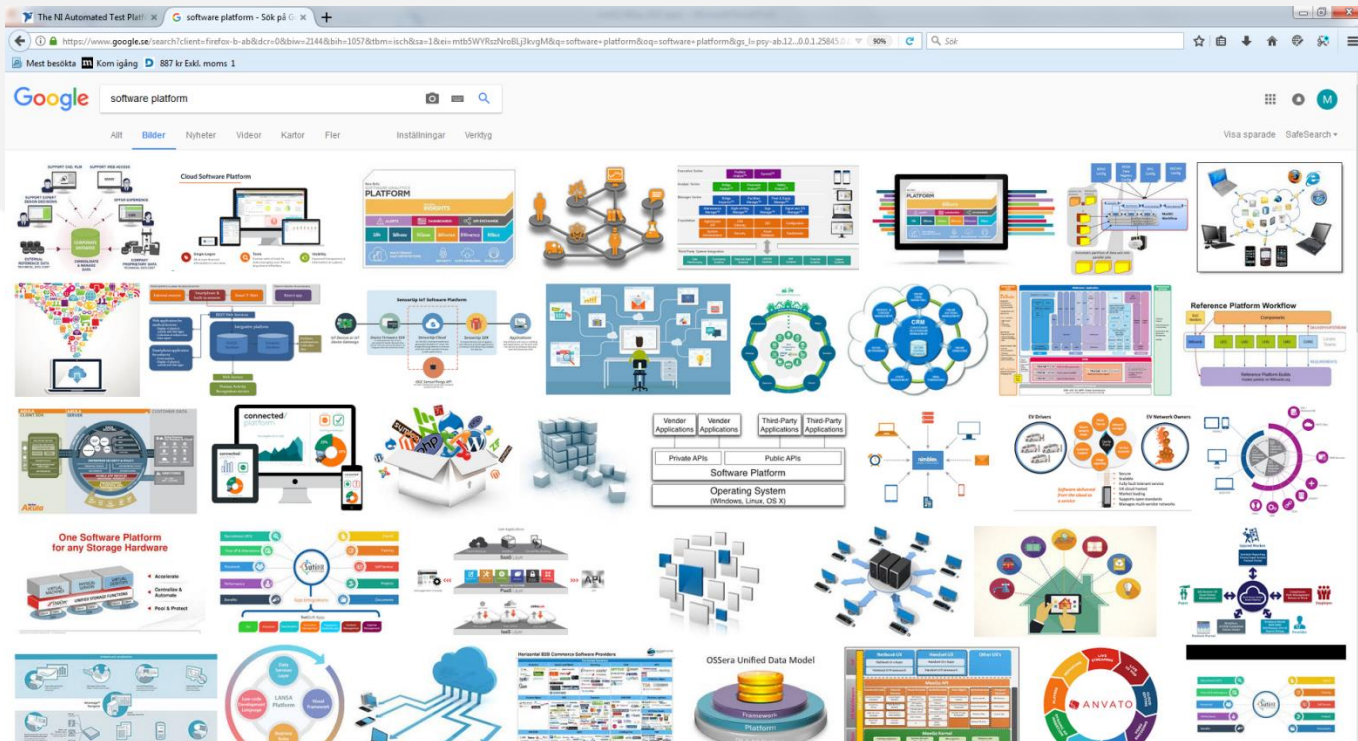
From Wiki:

- *“a raised level surface on which people or things can stand.”*
- *“a shoe with very thick soles.”*



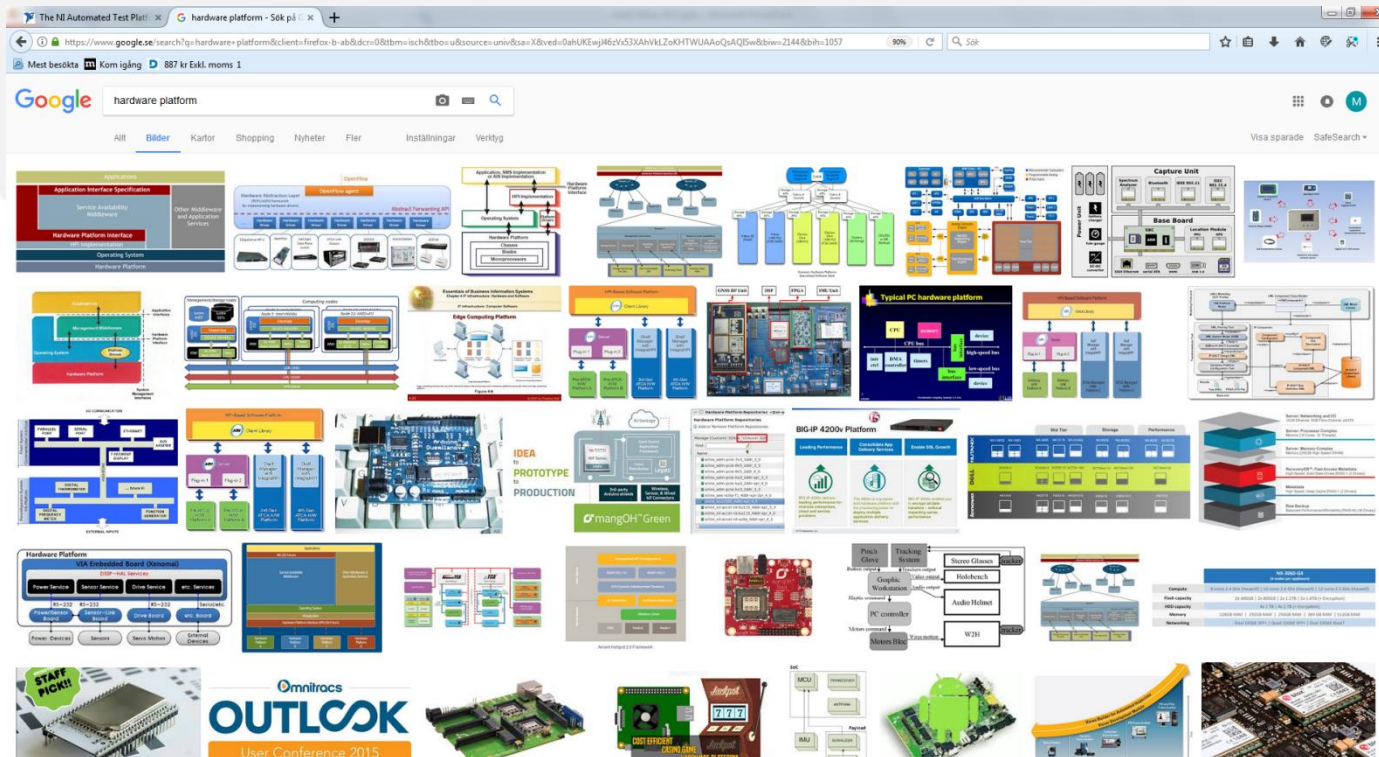
Software Platform - Definition

- *A software environment to write applications and run them*

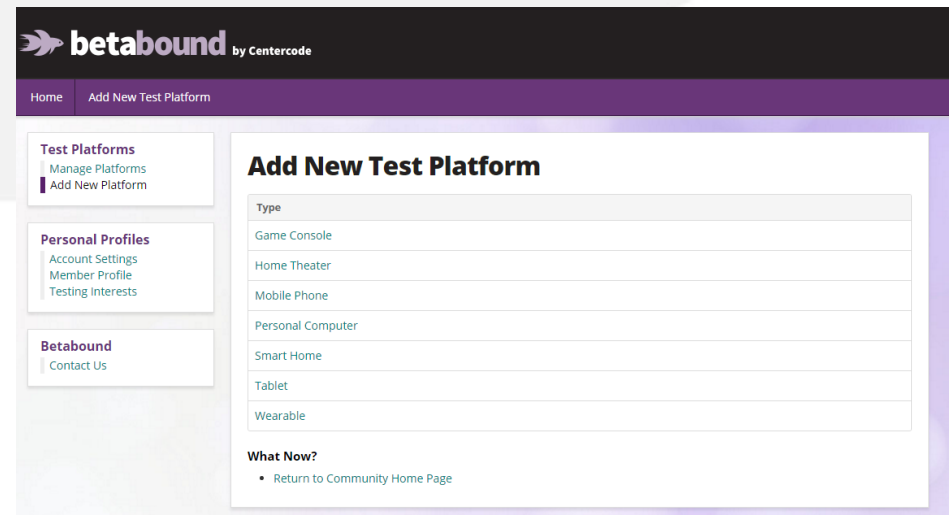


Hardware Platform - Definition

- A set of compatible hardware on which software applications can be run.*



Test Platforms



Choosing a Test Platform

- One size doesn't fit all!



- Building a successful platform is more about making the right trade-offs than it is about best technology.

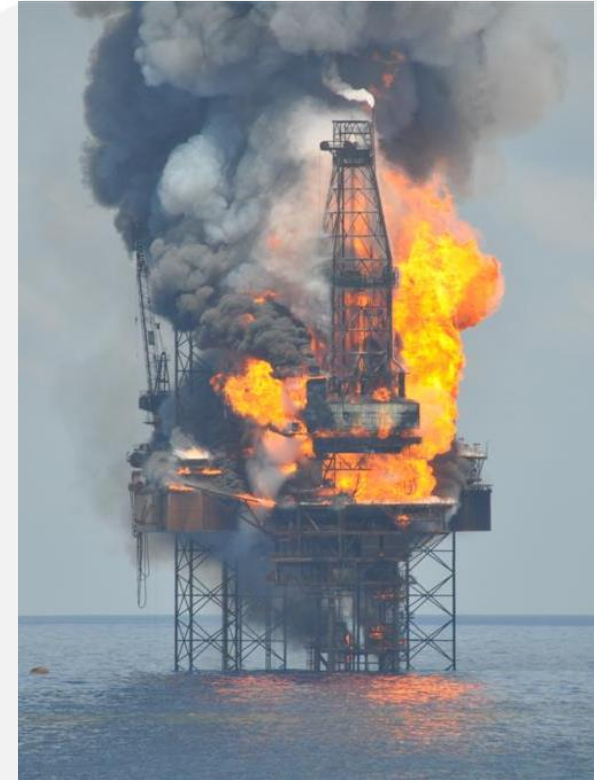
Why a Hardware Test Platform?

- Cost effective
 - Reuse resources between different test objects
- Software modules that operates on the hardware
- Duplication of stations



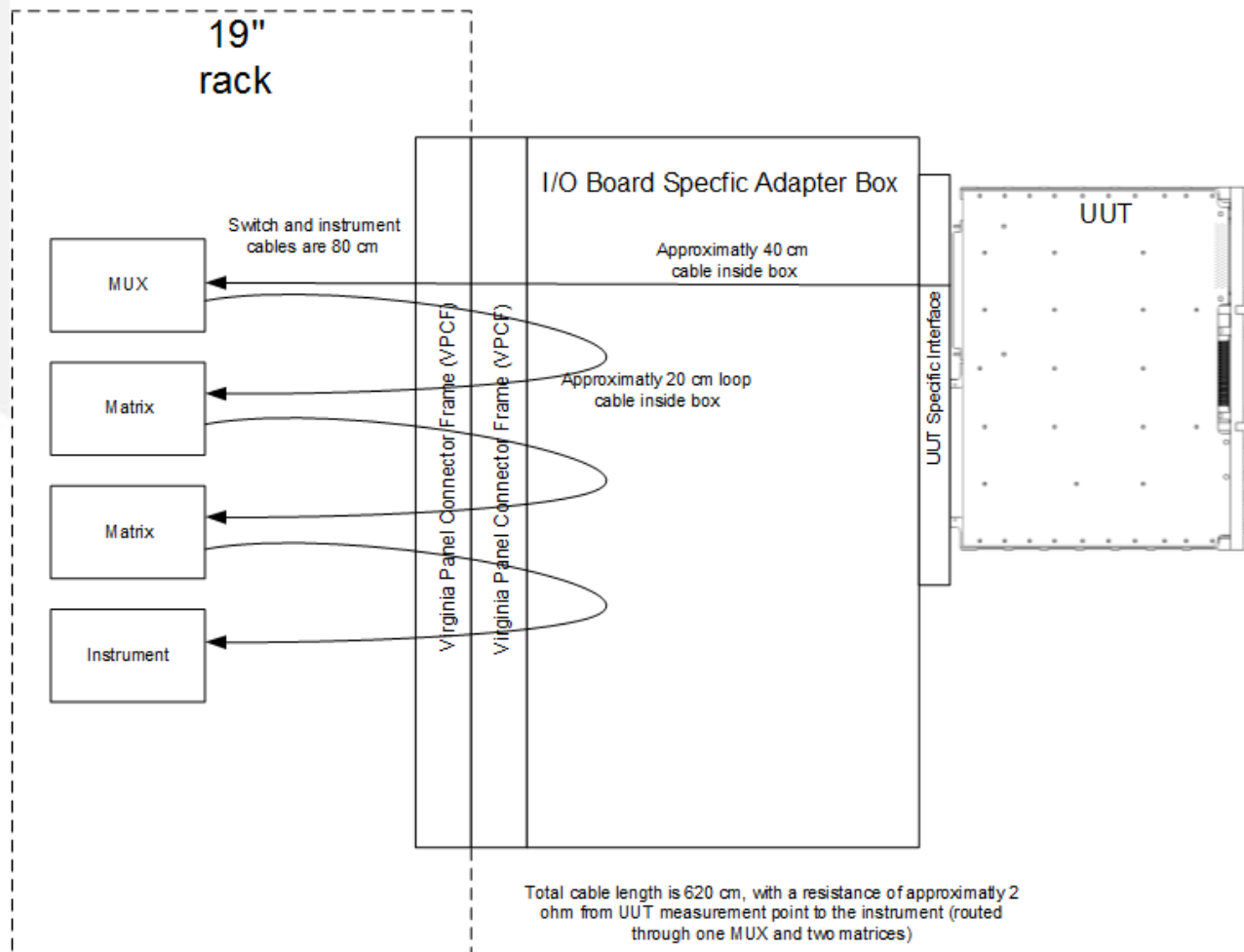
Hardware Platform Problems

- Hardware Interface
 - Connectors cost money
 - More resources than you need?
 - Longer signal paths
- Predict Future Needs
 - Difficult
 - Resources may never be used



Hardware Test Platform

Problem Example

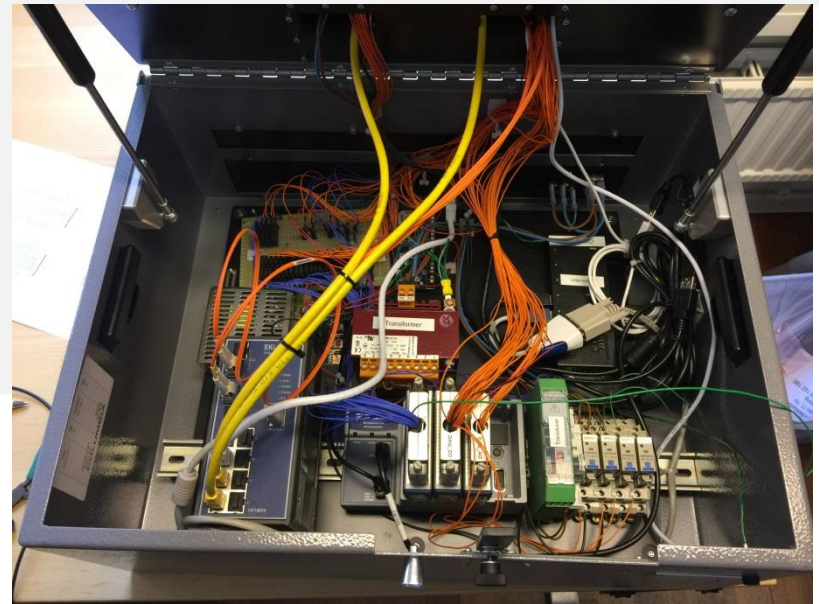
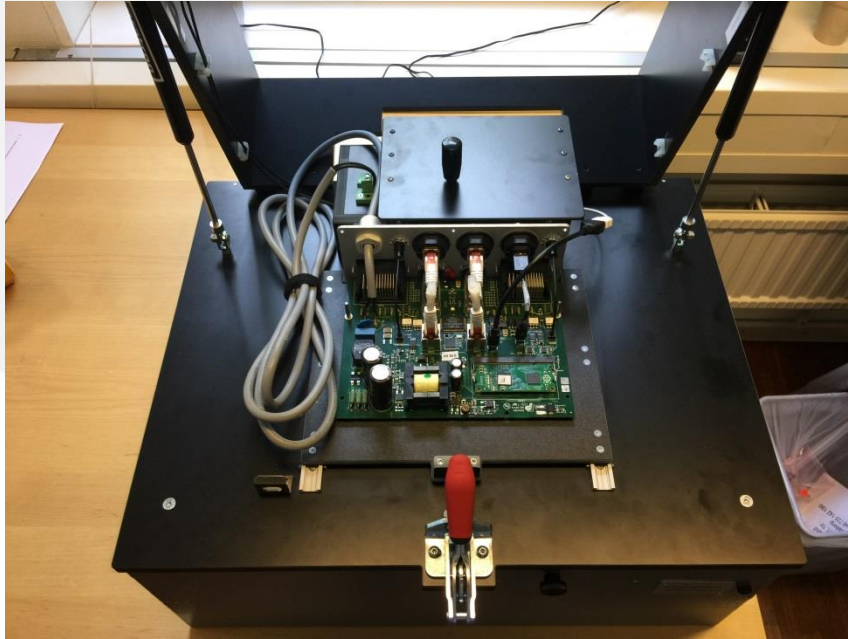


Why Not to Hardware Platform

- Hardware interface
 - Simpler
 - Short cables
 - Signal adaptation close to test object
- Resources
 - Only the needed
- No need to predict future!



Example No Hardware Platform



Why a Software Test Platform?

- Reuse of code
 - Debugged
- Faster development process
 - Less new code
 - Easier to introduce developers
- Extendable
 - Plugin
- Documentation



Software Platform Problems

- Increases complexity
- Endless configuration
- Hard to know how to extend
- Difficult to maintain



Test Platform Example

NI

The screenshot shows the NI Automated Test Platform website in Swedish. The browser address bar displays www.ni.com/automatedtest/platform/. The page features the National Instruments logo and a navigation menu with links for INNOVATIONER, BUTIK, SUPPORT, and ANVÄNDARGRUPPER. A user account section on the right shows 'MY ACCOUNT' for 'Hello Mattias | Log out'. The main content area is titled 'The NI Automated Test Platform' and includes a sidebar with a 'Shop' menu. The sidebar menu items are: NI Automated Test, Build Better Automated Test Systems, NI Automated Test Approach, Application Areas, Products and Support (highlighted), Fundamentals of Building a Test System, and PXI Advisor. The main content area contains three featured sections: LabVIEW, NI PXI, and TestStand, each with a description and a representative image.

Shop

- NI Automated Test
- Build Better Automated Test Systems
- NI Automated Test Approach
- Application Areas
- Products and Support**
- Fundamentals of Building a Test System
- PXI Advisor

The NI Automated Test Platform

National Instruments offers a complete automated test platform. All your needs from design to manufacturing are addressed with a combination of productive software, industry-leading instrumentation, and a full portfolio of global services.

[Speak to an NI engineer today](#)

LabVIEW

LabVIEW is the leading system design software for automated test that helps you develop powerful test software

NI PXI

PXI is the leading modular instrumentation platform used to build compact, high-performance automated

TestStand

NI TestStand is a powerful ready-to-run test management environment that helps you develop automated test and

<http://www.ni.com/automatedtest/platform/>

Framework

- Class libraries
- Architectural Idea
- Object-oriented architecture
 - Abstract base classes
 - High-level methods
- Extensibility
 - Application specific code plugin
- Control of Flow
 - "Hollywood Principle"



Hollywood Principle

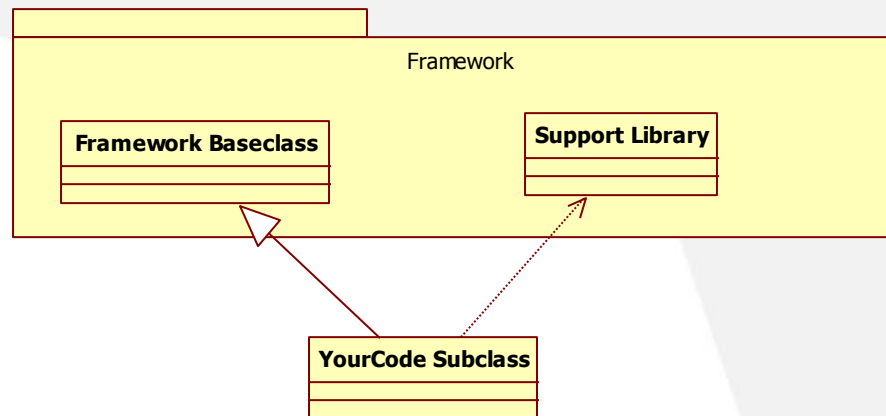
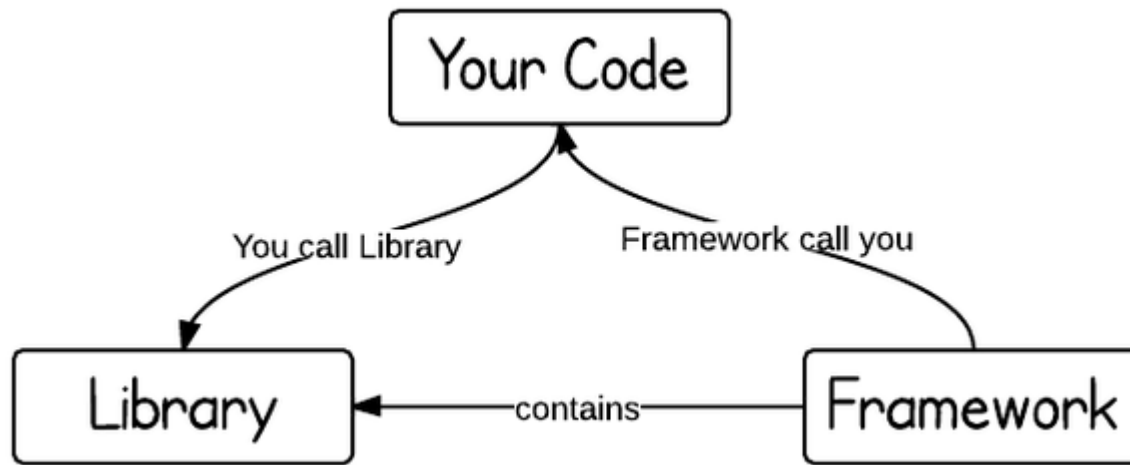
Don't call us, we'll call you



- Distinguishes a framework from a "library"
- Inversion of Control

When using a framework, the application-specific code written by the programmer gets called by the framework.

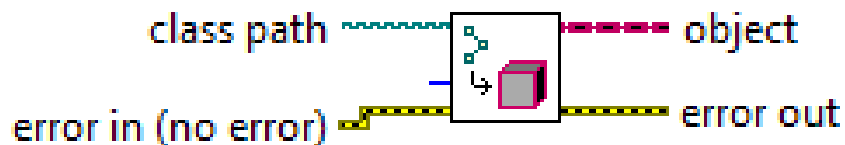
Hollywood Principle



Dependency Injection

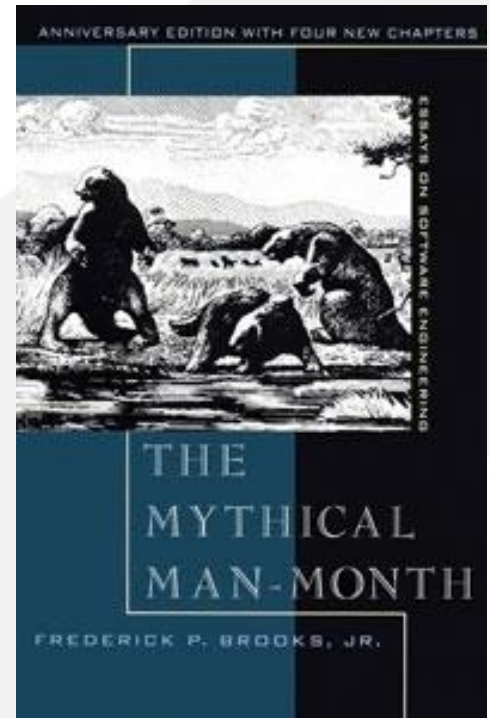
- Loose coupling
- Specify sub classes by path
 - INI-file
- HAL – Hardware Abstract Layer
- Mock Injection
- But... Exe-files are a bit tricky!

Get LV Class Default Value.vi



Second System Effect

- Introduced in 1975
- Frederick P. Brooks Jr (b. 1931)
- Aspects developing IBM OS/360
- The Second System
 - most dangerous system you will ever design
 - tend to incorporate all of the additions you originally did not add to the first system
 - over-engineering



Software Platform Experience

- 2003 Test Platform 1 (LV7, TestStand, GOOP2)
 - Endless configuration...
- 2008 Test Platform 2 (LV8, TestStand, GOOP3)
 - Endless extension by subclassing...
 - Second system effect
- 2010 Test Platform 3 (LV2009, TestStand, G#)
 - Complex, overdesigned
 - Dependency Injection
- 2015 QATS (LV2012, G#)
 - Simplified
 - XML test configuration
 - Loose coupling

QATS

Lessons Learned Software Platforms

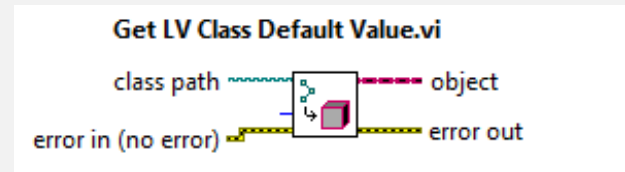
- Maintenance
 - Non-programmers
- Loose coupling
- Dependency Injection
 - Hardware vs Mocks (HAL)
 - Test Configuration
 - Result Export
- Design for most common scenario
- Keep it simple!



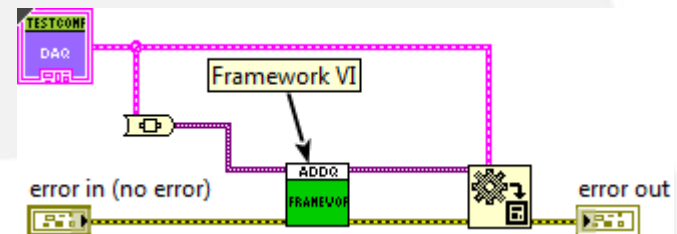


LabVIEW Framework Tips

- Object-orientation
 - Extension by abstract classes
 - Dependency Injection by path
 - Abstraction

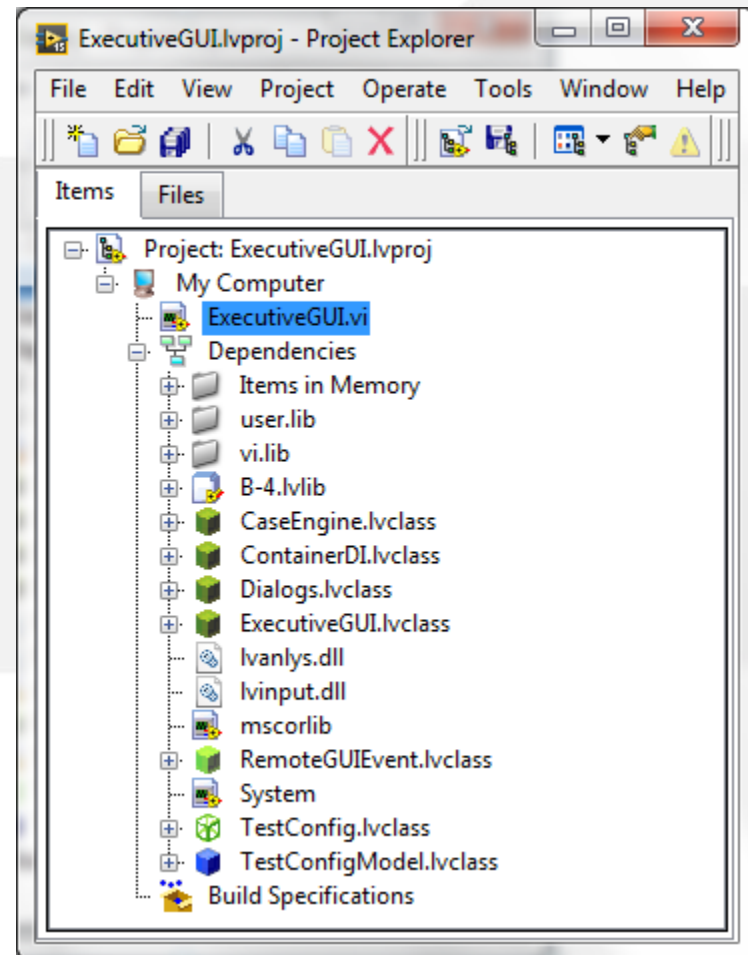
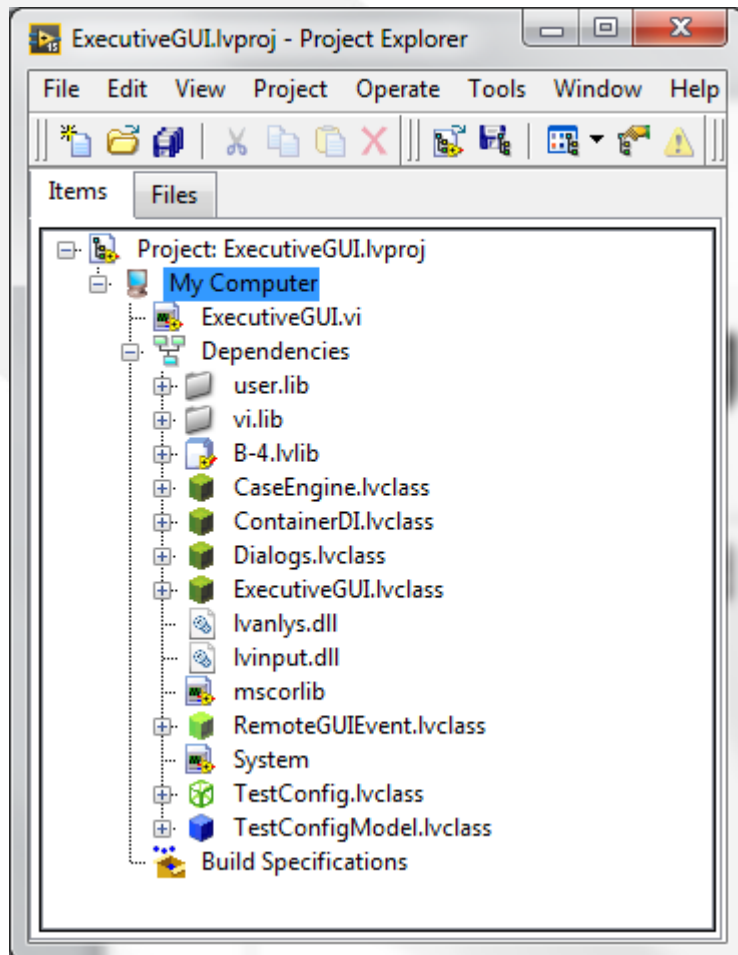


- Information Pipelines
 - Flatten to Variants



- Taking care of dependencies!
 - Look up for type defs

Checking Dependencies



To Platform...

Or

Not to Platform