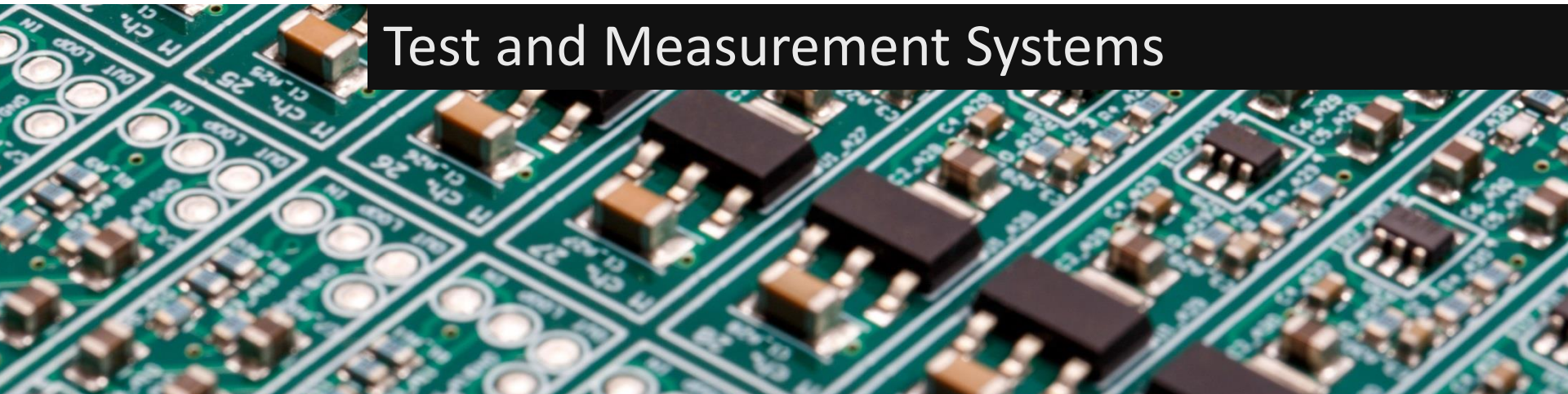
ADDQ

Question Everything!™



Test and Measurement Systems

Presentation

- **Mattias Ericsson**
  - LabVIEW developer ~15years
  - 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*

AddQ CONSULTING

The Art of Archiceture

- What is a Artictecture?
- Conciderations
- Different Strategies
- Conclusions

Add Q CONSULTING

Architecture

- ” The **software architecture** of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships between them.”

“Tell us what is important. Architecture is about the important stuff. Whatever that is.” – Ralph Johnson

AddQ CONSULTING

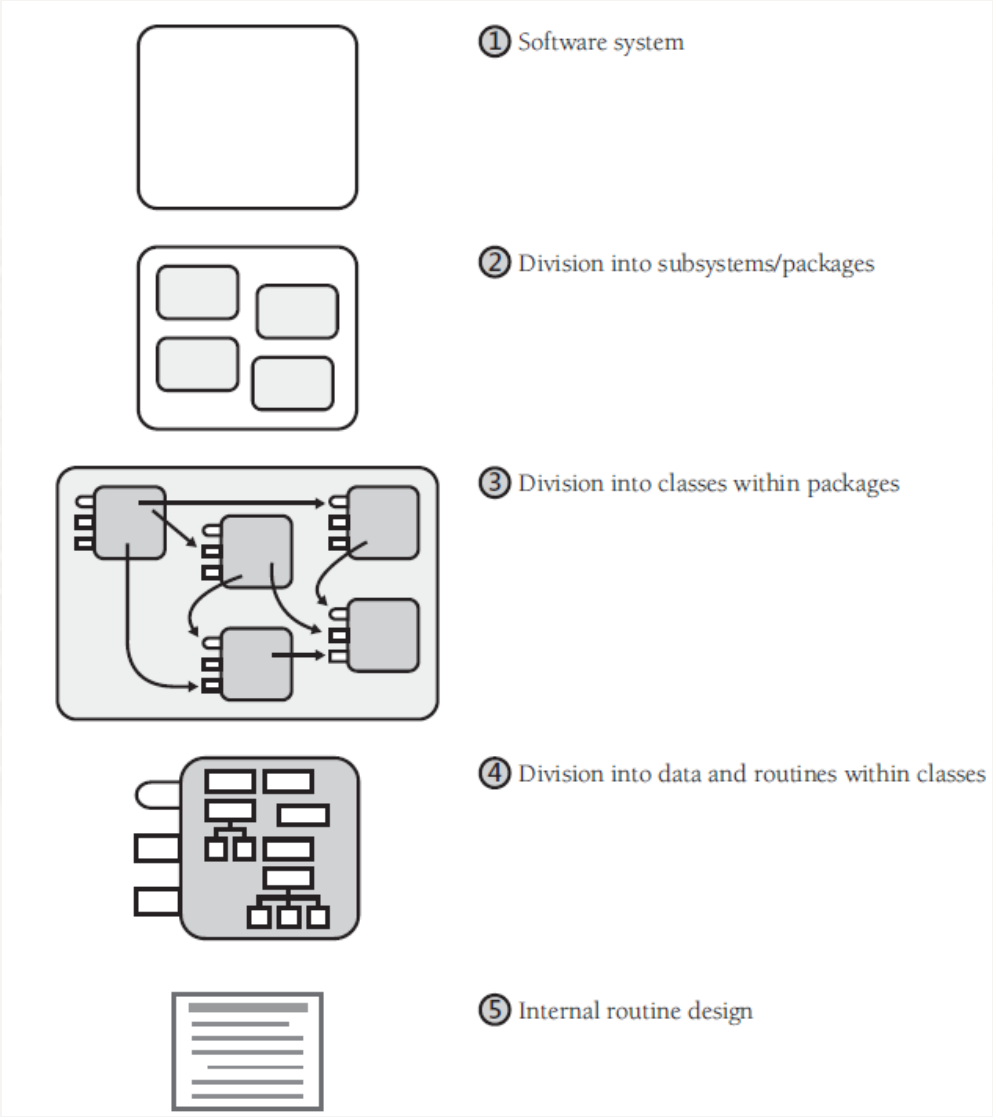
Architecture – some aspects

- **Ralph Johnson** - "*The expert developers working on that project have a shared understanding of the system design. That shared understanding is called architecture*."
- **Jim Coplien** - "*Architecture is when you design your abstract base classes*."
- **Martin Fowler –** "*Two common elements: Highest-level breakdown of a system into parts and the decisions that are hard to change*"
- **Eoin Woods** – "*The set of design decisions which, if made incorrectly, will cause your project to be cancelled*."

Design vs Architecture

- What is the *difference*?
  - Architecture **is** design, but not all design is architecture.
  - Many design decisions are left unbound by the architecture.
  - Architecture is often referred as **strategic** design

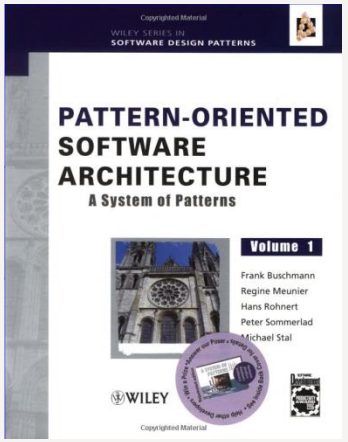
Design Levels



AddQ CONSULTING

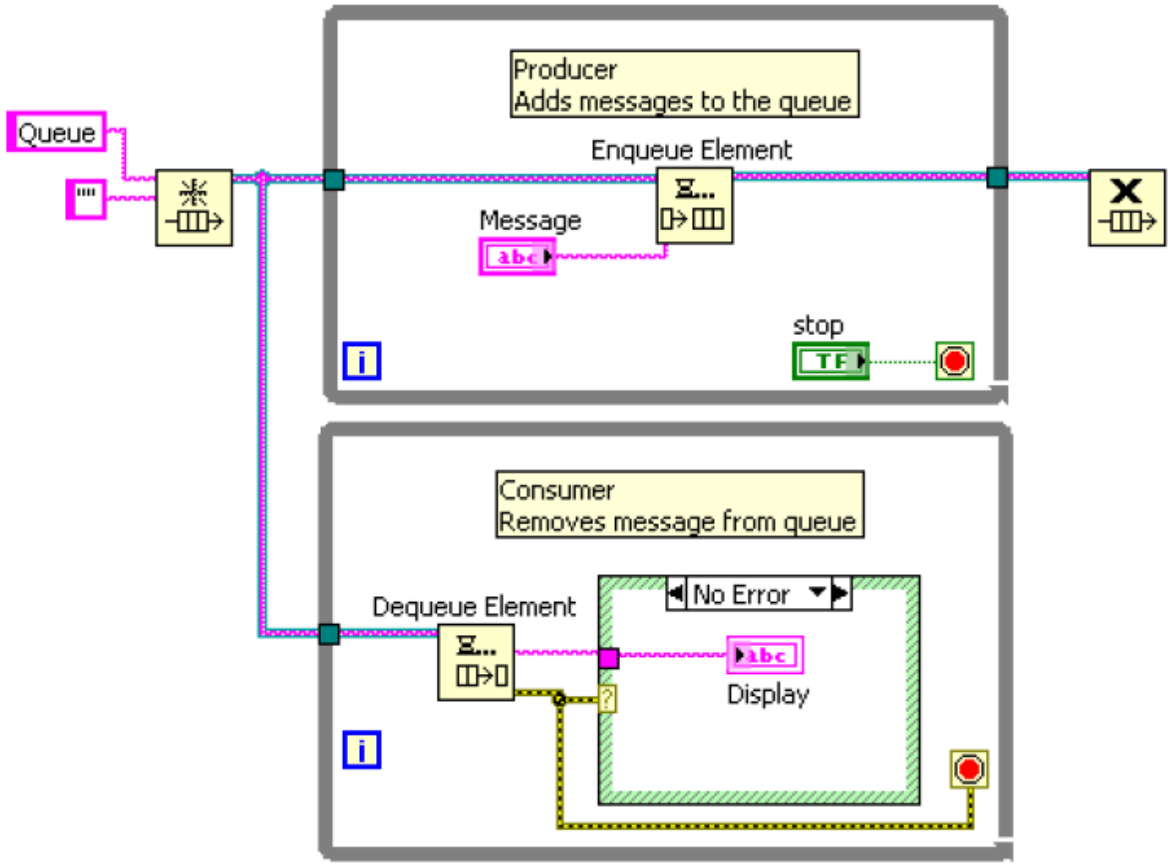
Architectural Patterns

- Expresses a fundamental structural organization scheme for a software system
- Examples:
  - Model-View-Controller (MVC)
  - Layers
  - LabVIEW: Producer-Consumer (aka Master-Queue)



AddQ CONSULTING

Producer / Consumer

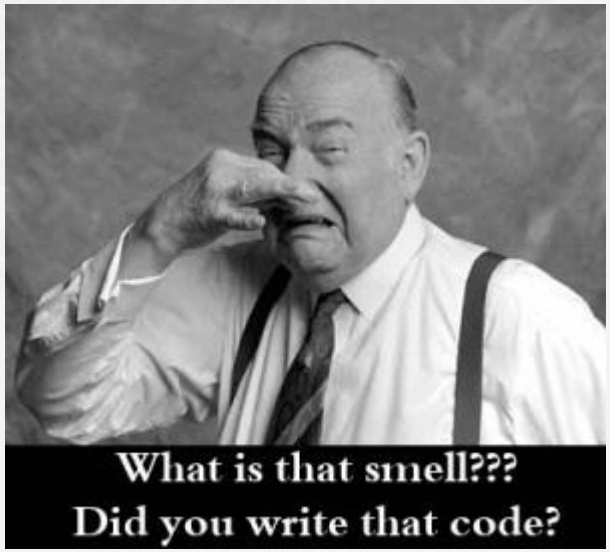


AddQ CONSULTING

Symptoms of Poor Design

The Odors of Rotting Software

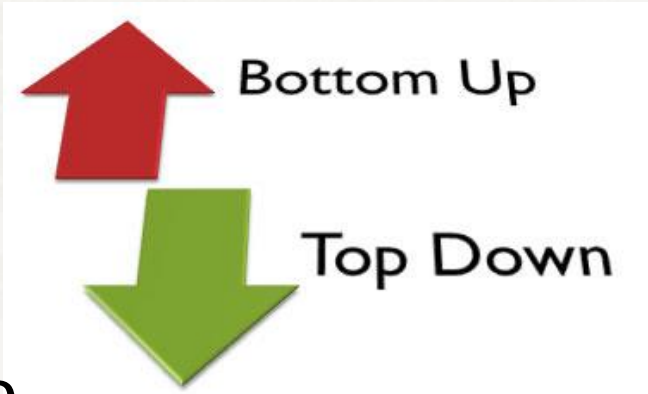
- Hard to change
- Easy to break
- Hard to reuse
- Hard to do the right thing
- Complexity and overdesign
- Repeating structures that could be unified under a single abstraction
- Hard to read and understand



AddQ CONSULTING

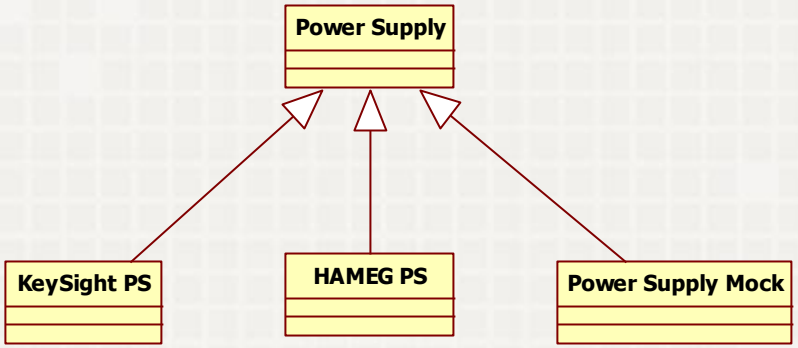
Design Approches

- Top-Down
  - Your brain isn't force to deal with too many details at once
  - Decomposition strategy
- Bottom-up
  - Sometimes the top-down is so abstract it is hard to get started.
  - When identifying several low-level responsibilities, start from the top again!
  - Composition strategy



AddQ CONSULTING

Hardware Abstract Layer (HAL)



To HAL or not to HAL is the question!

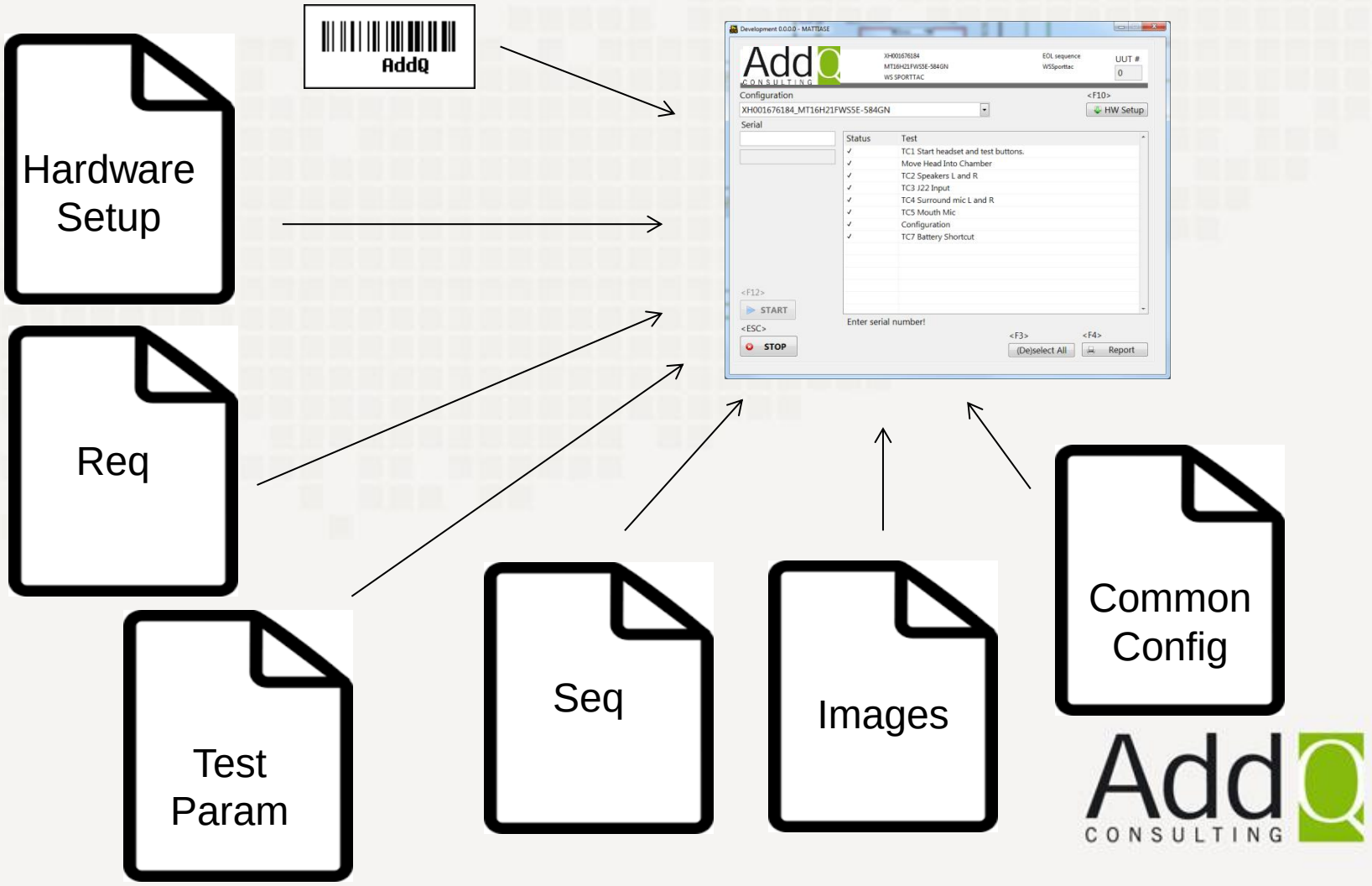
AddQ CONSULTING

Important Considerations

- Receiver of the system
  - Maintainer
  - Operator
- Station setups
  - Different brands of instruments
  - Different instrumentation
- Product support
  - Variants
  - Coexisting versions

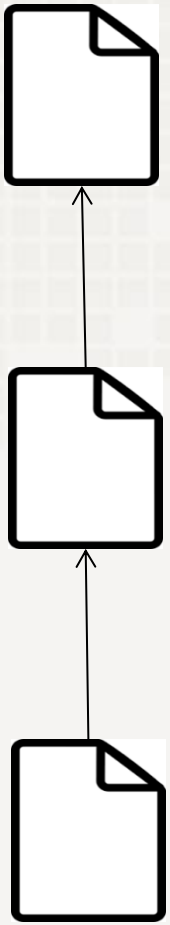
AddQ CONSULTING

System Configuration



Configuration Dependency

- Station Configuration
  - Hardware setup
  - COM ports etc.
- Product Configuration
  - Requirement and Sequences
  - Images
  - Different fixtures
    - Channel maps
  - Firmware
- Common Configuration
  - Platform setup

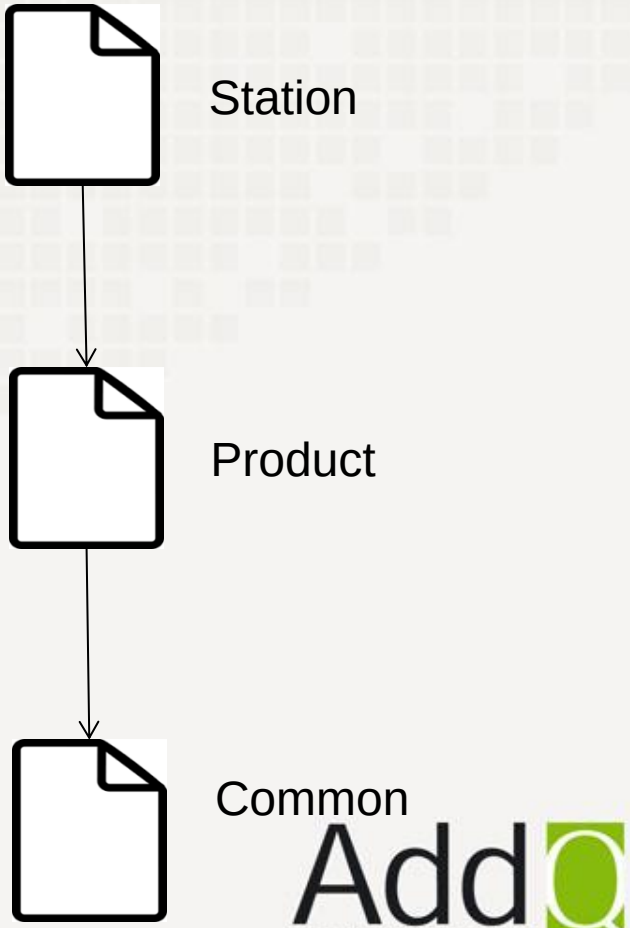


AddQ CONSULTING

Configuration Override

- INI-file
  - Search hierarchy
    1. File name
    2. Section
    3. Key
- XML
  - Compare Microsoft .NET

Always use relative paths!



AddQ CONSULTING

Example of Simplification

Problem:

- TestStand
  - Sequences
  - Variants requires own sequence
- LabVIEW
  - Test Code State Machine – sometimes product specific
- INI for requirements
- INI for test cases
- Station specific
  - Channel maps
- No override for local changes

Very hard for maintainer to know what to do!

AddQ CONSULTING

Example of Simplification

Solution:

- LabVIEW
  - Test Code – generic simple steps
  - One platform for all station, products and variants.
- XML
  - Sequence
  - Requirements and parameters
  - Test steps
- Configuration override
  1. Station
  2. Product
  3. Common

Easy for maintainer to manage!

AddQ CONSULTING

XML

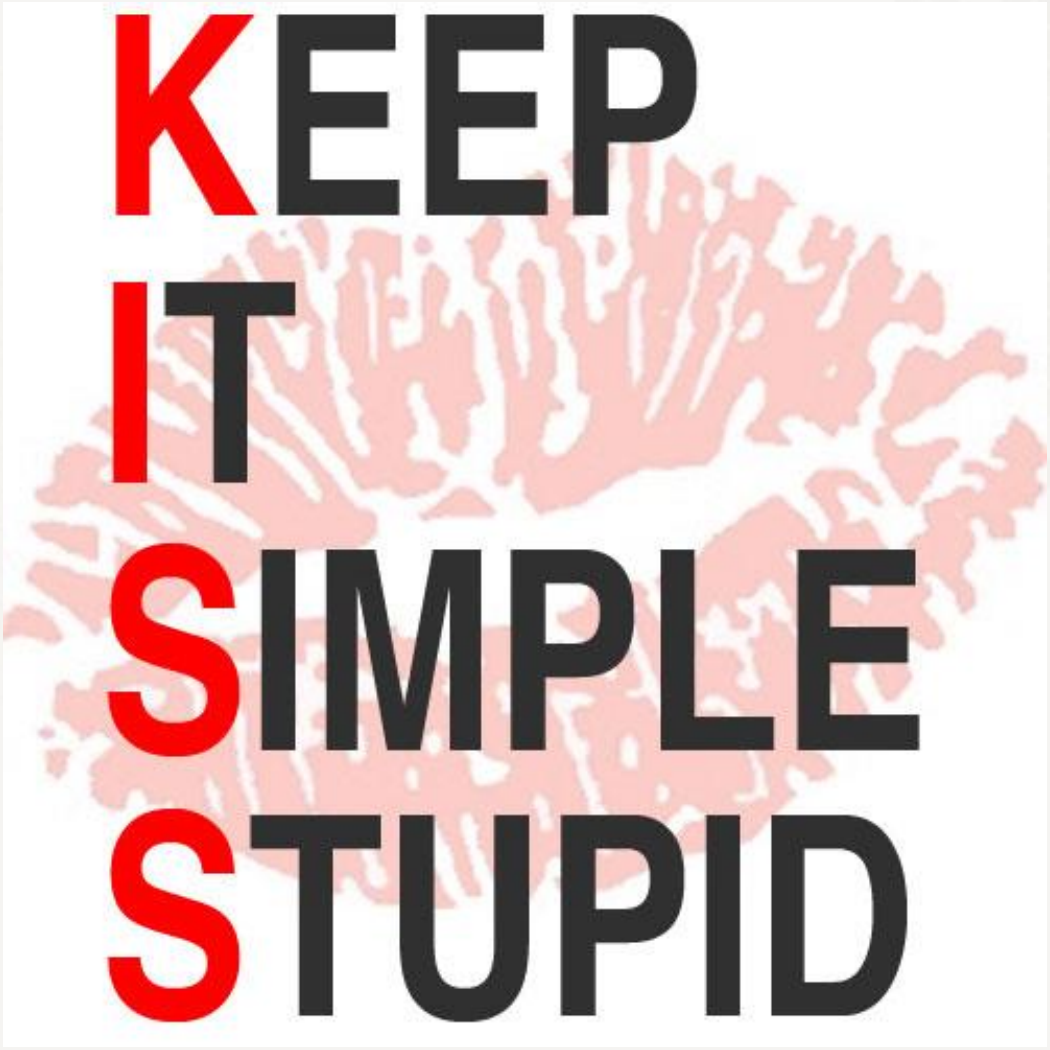
```
<?xml version="1.0" encoding="UTF-8"?>
<testsuite name="PCBA test" schemaVersion="1.0" xmlns="http://www.addq.se" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="ht
    <uut autogenerateserial="false">
    <setup>
        <power>
            <output>
                <powerchannel channelname="MainSupply" voltage="14.45" currentlimit="3" enab
            </output>
        </power>
        <signalroute>
            <checkroute logic="AND">
                <condition name="EmergencyStop"/>
                <evaluatetrue info="Please reset emergency stop!" type="ok"/>
            </checkroute>
            <route name="PowerEnable" set="true" waittime="100"/>
            <route name="MainPowerSoftStart1" set="true" waittime="1200"/>
        </signalroute>
    </setup>
    <test name="TC4.1" stopiffail="true">
        <tdesc>Input current</tdesc>
        <dmm>
            <input>
                <aichannel channelname="CurrentShunt">
                    <analysis method="amplitude" unit="A" comparator="GELE" reqmin="2.20" re
                </aichannel>
                <aichannel channelname="Sense">
                    <analysis method="amplitude" unit="V" comparator="GELE" reqmin="13.9" re
                </aichannel>
            </input>
        </dmm>
    </test>
    <test name="TC4.2">
        <tdesc>Over voltage turn off</tdesc>
        <power>
            <output>
                <powerchannel channelname="MainSupply" voltage="18.0" currentlimit="3" enabl
            </output>
        </power>
        <dmm>
            <input>
                <aichannel channelname="CurrentShunt">
                    <analysis method="amplitude" unit="mA" comparator="GELE" reqmin="0" reqm
                </aichannel>
                <aichannel channelname="Sense">
                    <analysis method="amplitude" unit="V" comparator="GELE" reqmin="17.9" re
                </aichannel>
            </input>
        </dmm>
        <power>
            <output>
                <powerchannel channelname="MainSupply" voltage="14.45" currentlimit="3" enab
            </output>
        </power>
```

```
[Common]
Lock=DAQ1:DO,300
ReversePolarity=DAQ1:DO,301
MainPowerSoftStart1=DAQ1:DO,305
MainPowerSoftStart2=DAQ1:DO,306
DualPowerSupply=DAQ1:DO,304
EmergencyStop=DAQ1:DI,303::0
#Inverted
FixtureClosed=DAQ1:DI,302::0
#Inverted
FixtureID=DMM:Resistance,Auto,N/A,320

[UUT/1]
MainSupply=PS1
PowerEnable=DAQ1:DO,400
# Port and channel
CurrentShunt=DMM:Voltage,0,DC,101
#Mode,range,bank,channel
OvertempResistor=DAQ1:DO,000
LED1=LEDAn:1
LED2=LEDAn:2
LED3=LEDAn:3
LED4=LEDAn:4
LED5=LEDAn:5
LED6=LEDAn:6
Sense=DMM:Voltage,Auto,DC,102
I-sense1=DMM:Voltage,Auto,DC,103
I-sense2=DMM:Voltage,Auto,DC,104
&&Common

[UUT/2]
MainSupply=PS1
PowerEnable=DAQ1:DO,401
# Port and channel
CurrentShunt=DMM:Voltage,0,DC,105
#Mode,range,bank,channel
OvertempResistor=DAQ1:DO,001
LED1=LEDAn:7
LED2=LEDAn:8
LED3=LEDAn:9
LED4=LEDAn:10
LED5=LEDAn:11
LED6=LEDAn:12
Sense=DMM:Voltage,Auto,DC,106
I-sense1=DMM:Voltage,Auto,DC,107
I-sense2=DMM:Voltage,Auto,DC,108
&&Common
```

Architectural Strategies



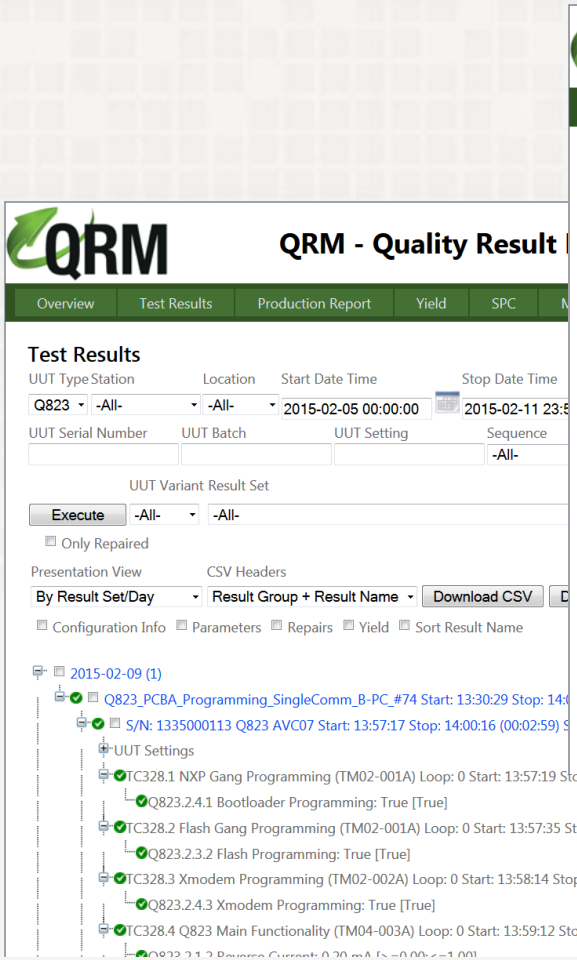
AddQ CONSULTING

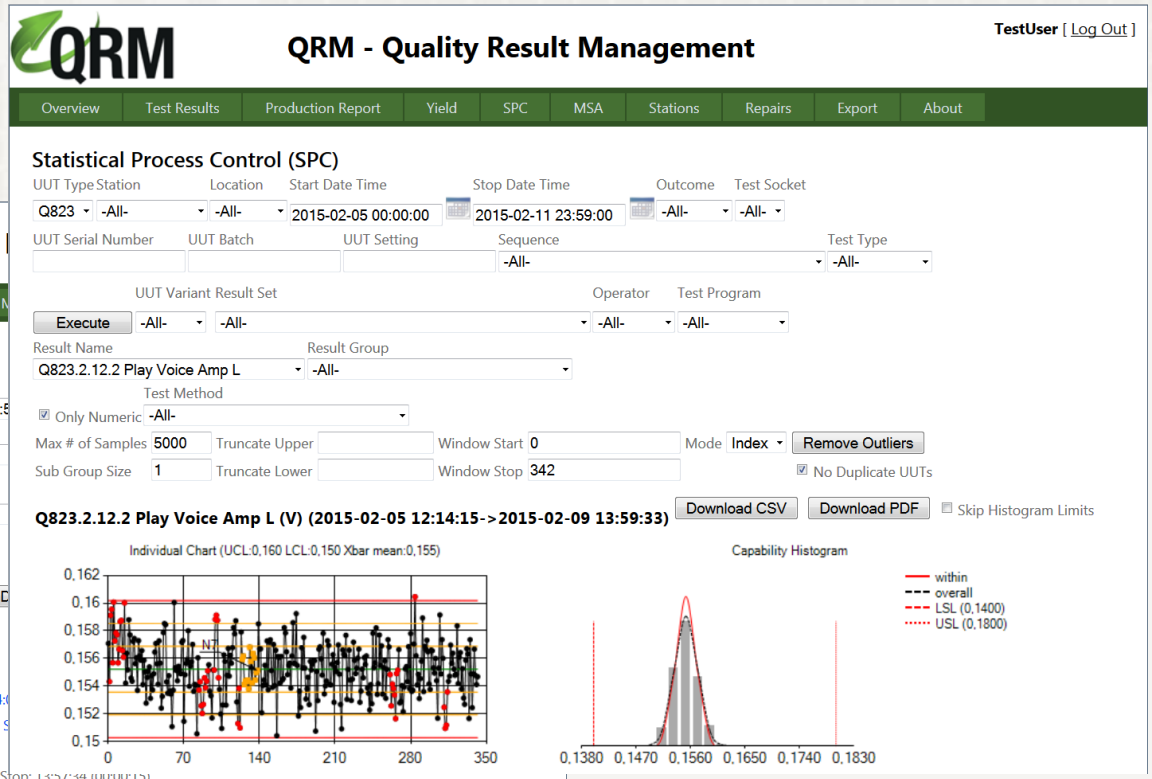
Architectural Strategies

- **What** to test?
  - Test object
  - Test equipment
- **Who** is going to maintain?
- Product and Variants
  - What's next to be tested?
- Serial Numbers
  - Generation
  - Scanning
- Documentation
  - Block diagrams e.g. UML
  - Schemes and Manuals
- Results
  - Measurements
  - Traceability

AddQ CONSULTING

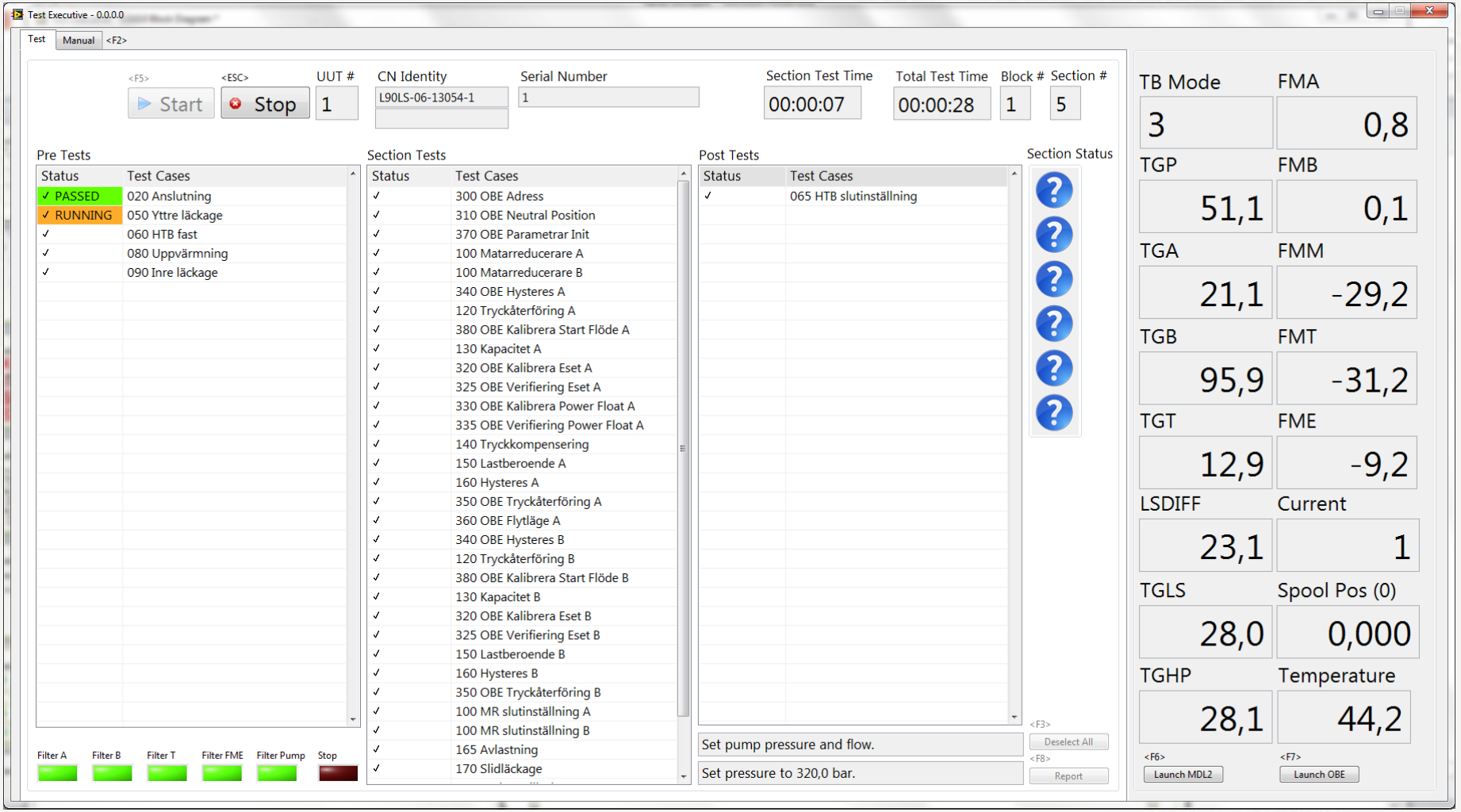
Web Application



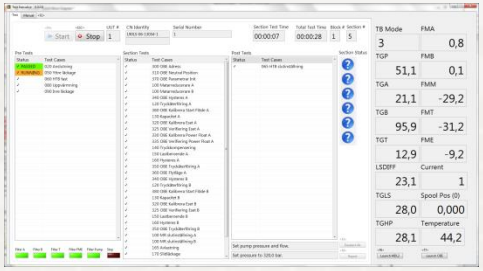


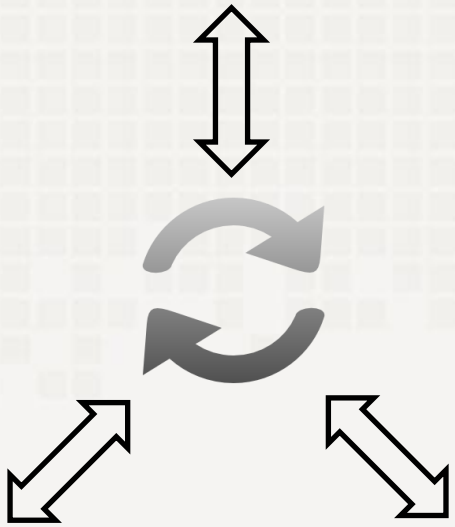
AddQ CONSULTING

Background Meas Process



Background Meas Process





1. Application launch background process.
2. Background process measures and constantly writes to memory.
3. Application regular polls memory for lates meas values.
4. Application closes process when closing down.

+ Easy switch between Test and Manual

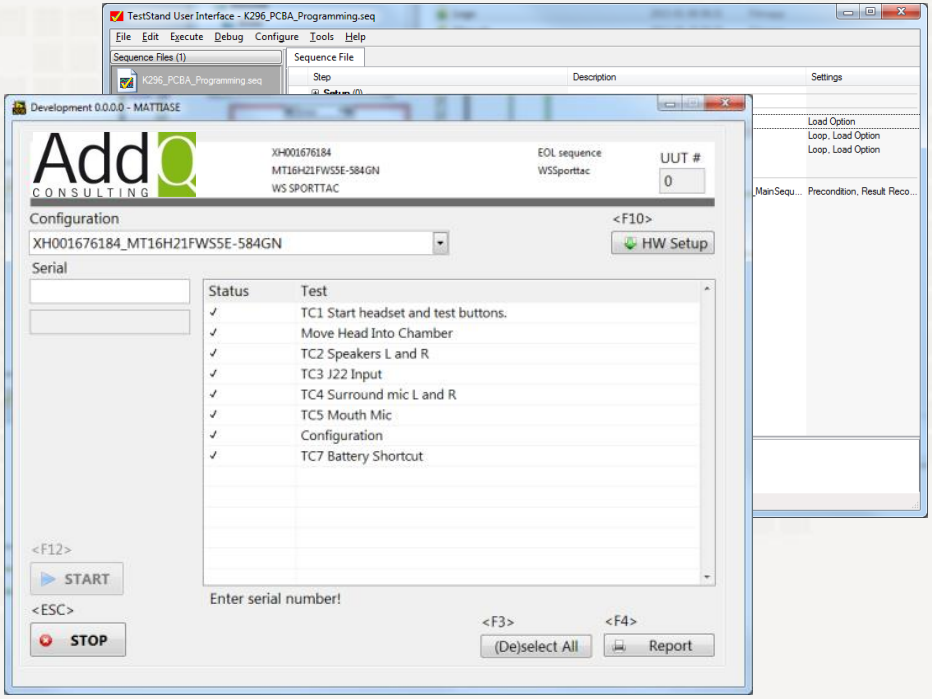
+ Always monitoring

+ Averaging

- Slow sampling

Add Q CONSULTING

”Hit and Run”



1. Application initialize equipment.
2. Sequential testing
    1. Measure
    2. Analyze
3. Application close equipment when closing down.

\+ Easy control flow

\+ Simple

\+ High performance

\- No monitoring

Add Q CONSULTING

Conclusion

- Architecture is not only software design
  - Configuration
  - Maintenance
  - Product and variants
  - Result Management
- No overdesign!
- Documentation
  - Keep up to date!
- KISS – Keep It Simple Stupid!

AddQ CONSULTING