



Aalto University
School of Electrical
Engineering

Experiences in Adoption of MBT to Continuous Process Control Applications

Jukka Peltola, Pekka Aarnio, Kari Koskinen
Aalto University (jukka.peltola@aalto.fi)

Tekes/ReUse & ITEA2/Diamonds
Mika Karaila (Metso Automation)
Stephan Schulz (Conformiq)

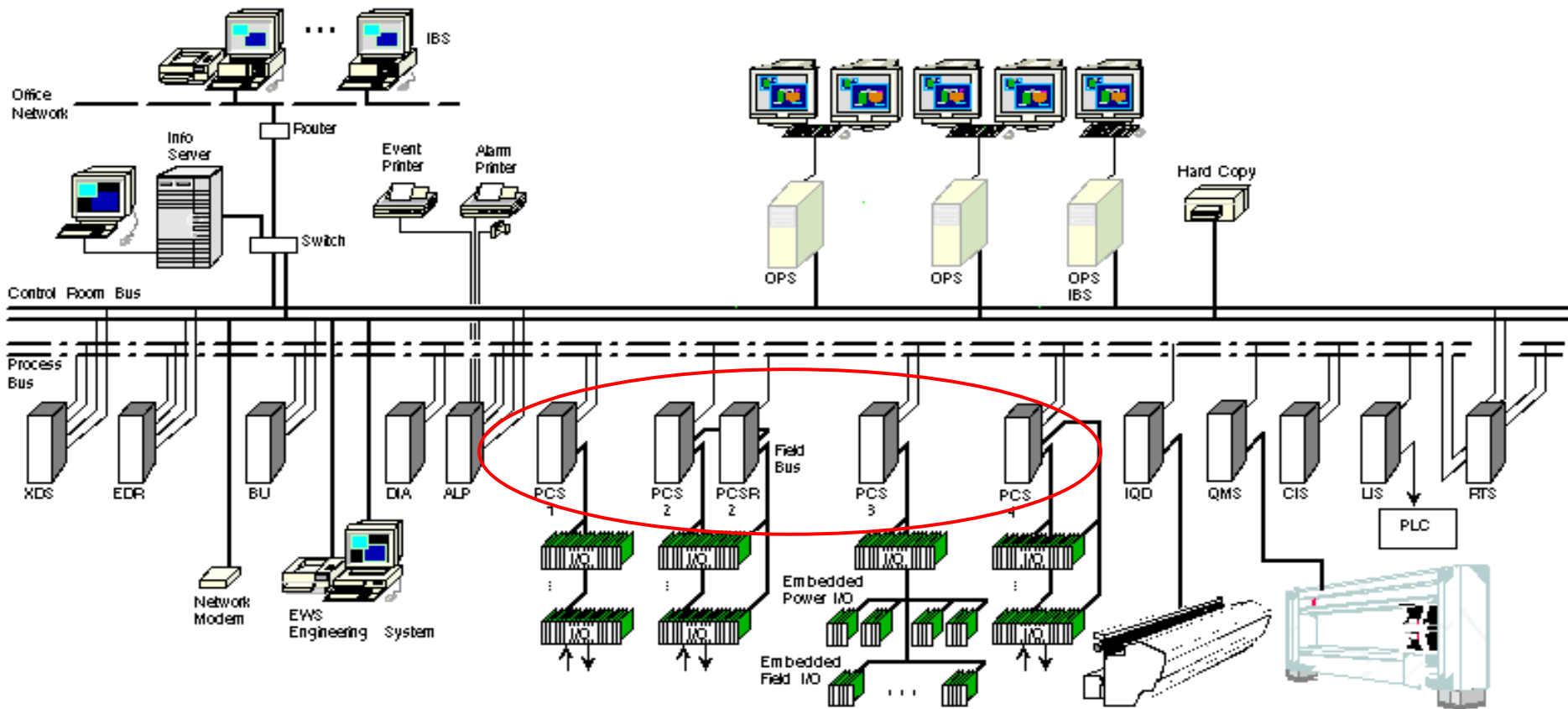
MBT User Conference 2012
25 – 27 September 2012 - Tallinn, Estonia

Contents

- Goal: Improve Factory Acceptance Testing of Control Loops
 - Reduce manual work
 - Higher test quality (coverage)
- Business Case:
 - Process Phase: Factory Acceptance Testing of a Distributed Control System
 - SUT: Control Loop (simple but many)
- Solution Approach:
 - Functional Model Based Testing
 - System modeling with state machines
 - With model reuse

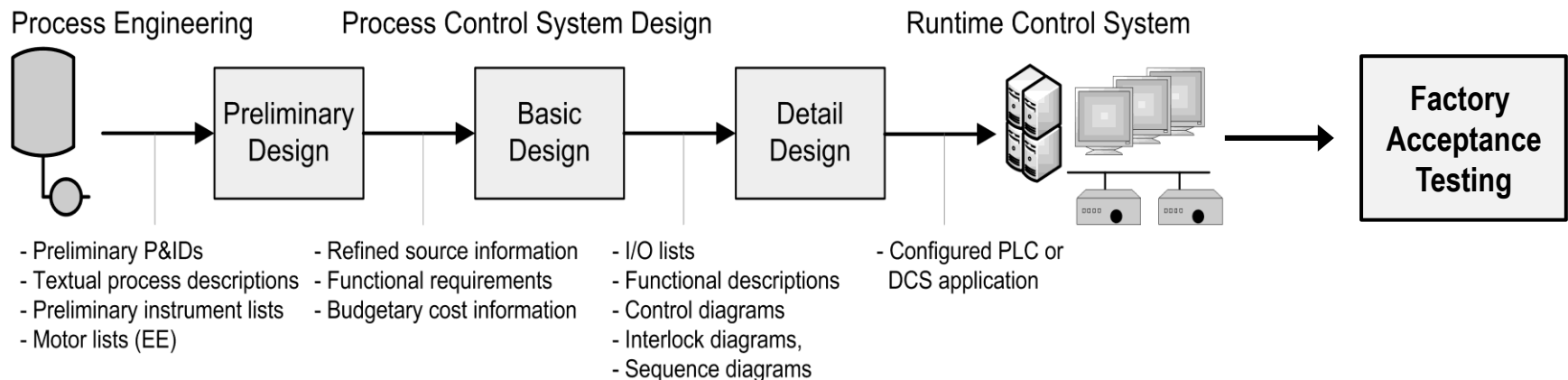
Application Context: Typical Distributed Control System

- 5-10 Process Control Stations (PCS)
- Within stations 1.000s-10.000s of Control Loops
- Control Loop software modules are instantiated from a repository of design templates



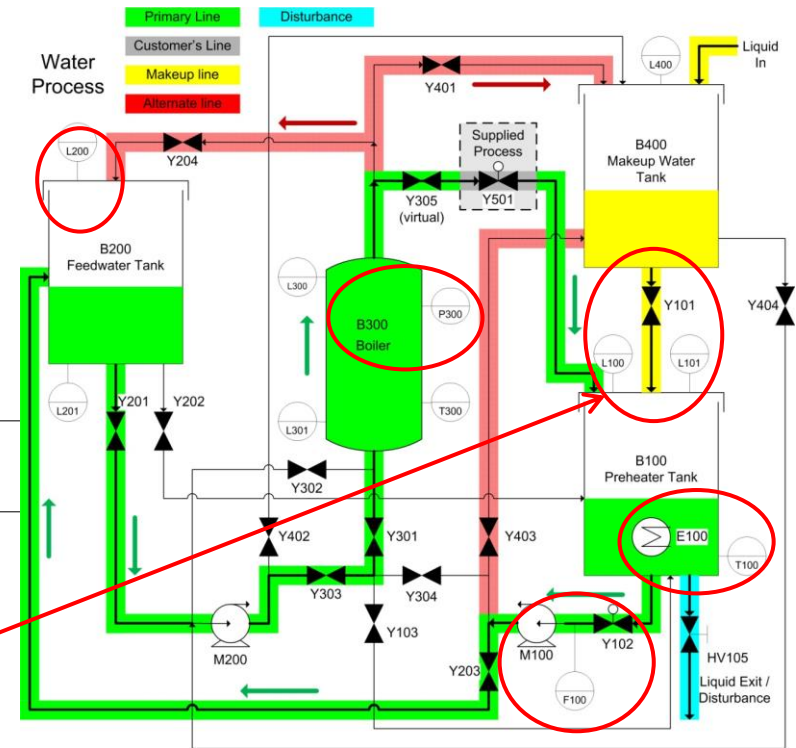
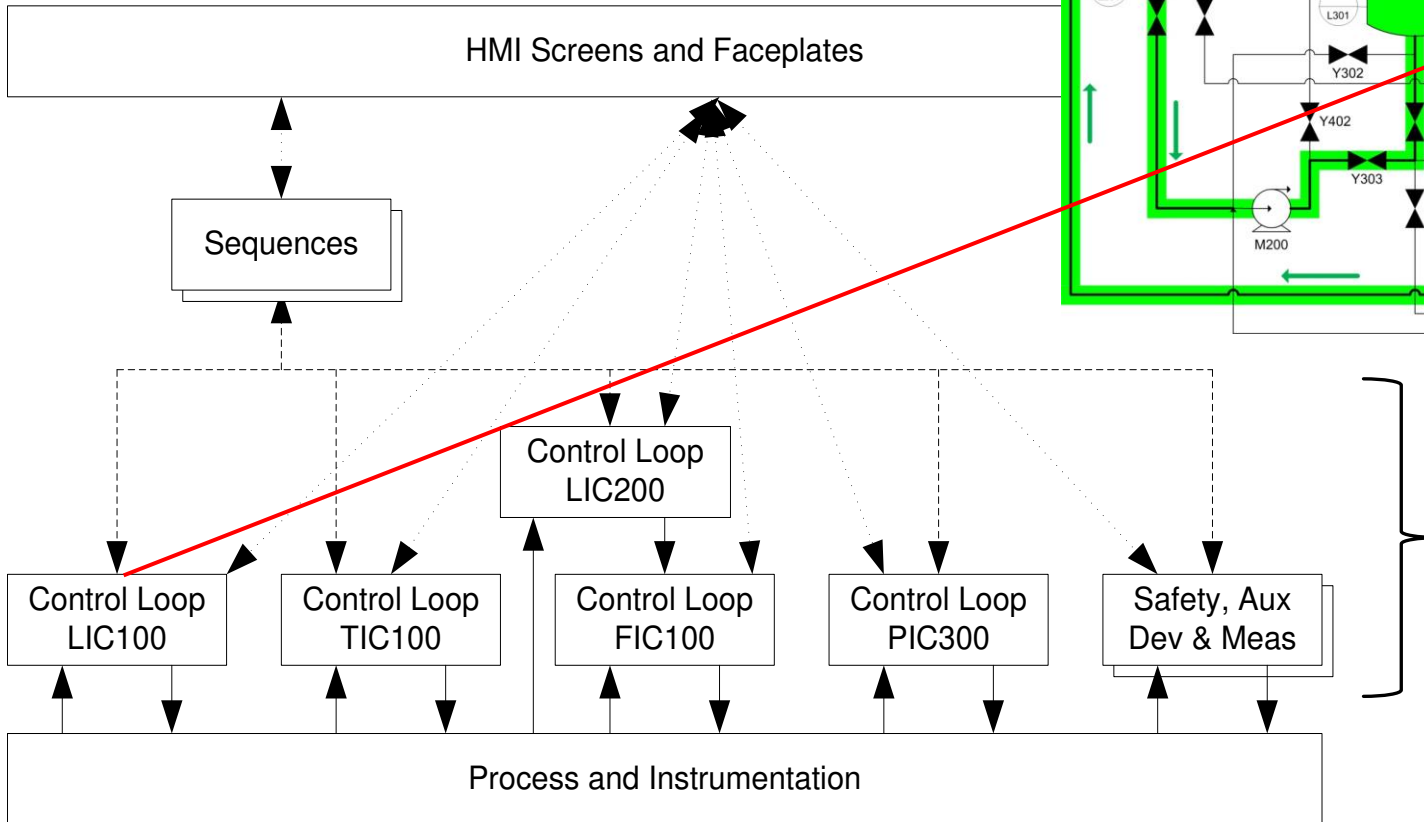
Factory Acceptance Testing in a Plant Delivery Process

- Last QA step before shipping and Site Acceptance Testing
- HW & SW installed and tested without physical process devices
 - HW: Process Control Stations, Cross connections, I/Os, Buses, ...
 - SW: measurement & actuation modules, control modules, alarms, interlocks, sequences, monitoring)
- Each control loop is tested manually by a test technician using the system's debugging interfaces and Basic Design documents
 - 1000s of loops → very labor intensive ("~100 testers working for weeks")
- Potential for various forms of test automation



About Control Loops

- Handle a group of functions: measurement – control – device actuation, alarms, interlocks
- Cyclic software applications
- Instantiated from design templates consisting of proprietary Function Blocks
 - Configured with parameters and feature switches



Example SUT: Functional Description of LIC100 Control Loop

3.1.1 LIC100, Preheater Tank (B100) Surface Level Control

Measurement

L100 is used as level measurement.

Control

Surface level of the Tank is controlled using a limit controller. Desired level should be maintained within +/-5 mm or less around setpoint.

Actuation

An on/off valve *Makeup Water Valve* (Y101) is used as the controlling device.

User Interface

A display in operating station is needed.

Alarms

Warnings and alarms of surface level (LL, L, H, HH) should be produced.

Interlocks

Overflow of the tank should be prevented by using the level measurement's (L100) HH alarm and, as a backup, a separate overflow switch (L101). Either of the two should close all inlet streams, including the customer line Y305.

Signals from other functions

-

Information to other loops

LL-alarm is used in Preheater Pump's (M100) interlock.

LL-alarm is used in Heater's (E100) interlock.

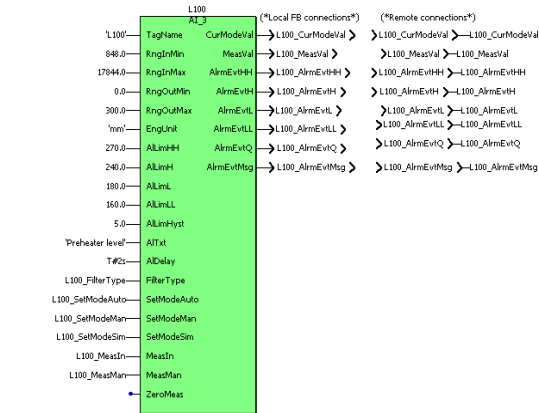
+ design template specification
➔ source material for testing

Anatomy of the LIC100 Control Loop Implementation

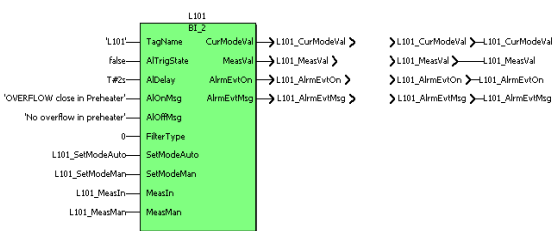
■ Level control for a tank (LIC100PROG).

- Level measurement (L100),
- Hysteresis control (LIC100),
- On/off valve control (Y101),
- Overflow switch (L101) and
- Device interlocks (Y101INT) for valve.

Level measurement

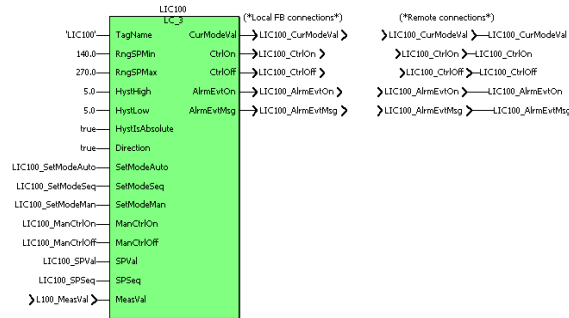


(*Preheater B100 Overflow Switch*)



Overflow switch

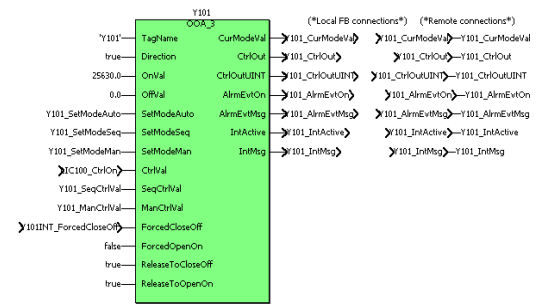
(*Surface Level Control*)



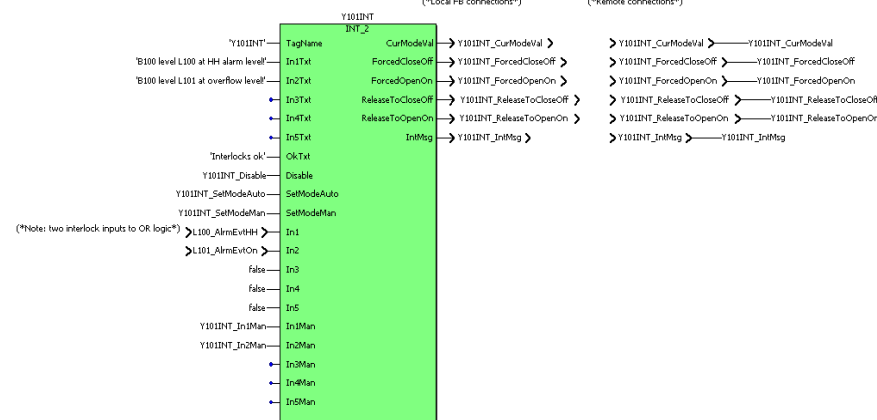
Hysteresis control

Valve control

(*Valve Control*)



(*Device Interlocks*)



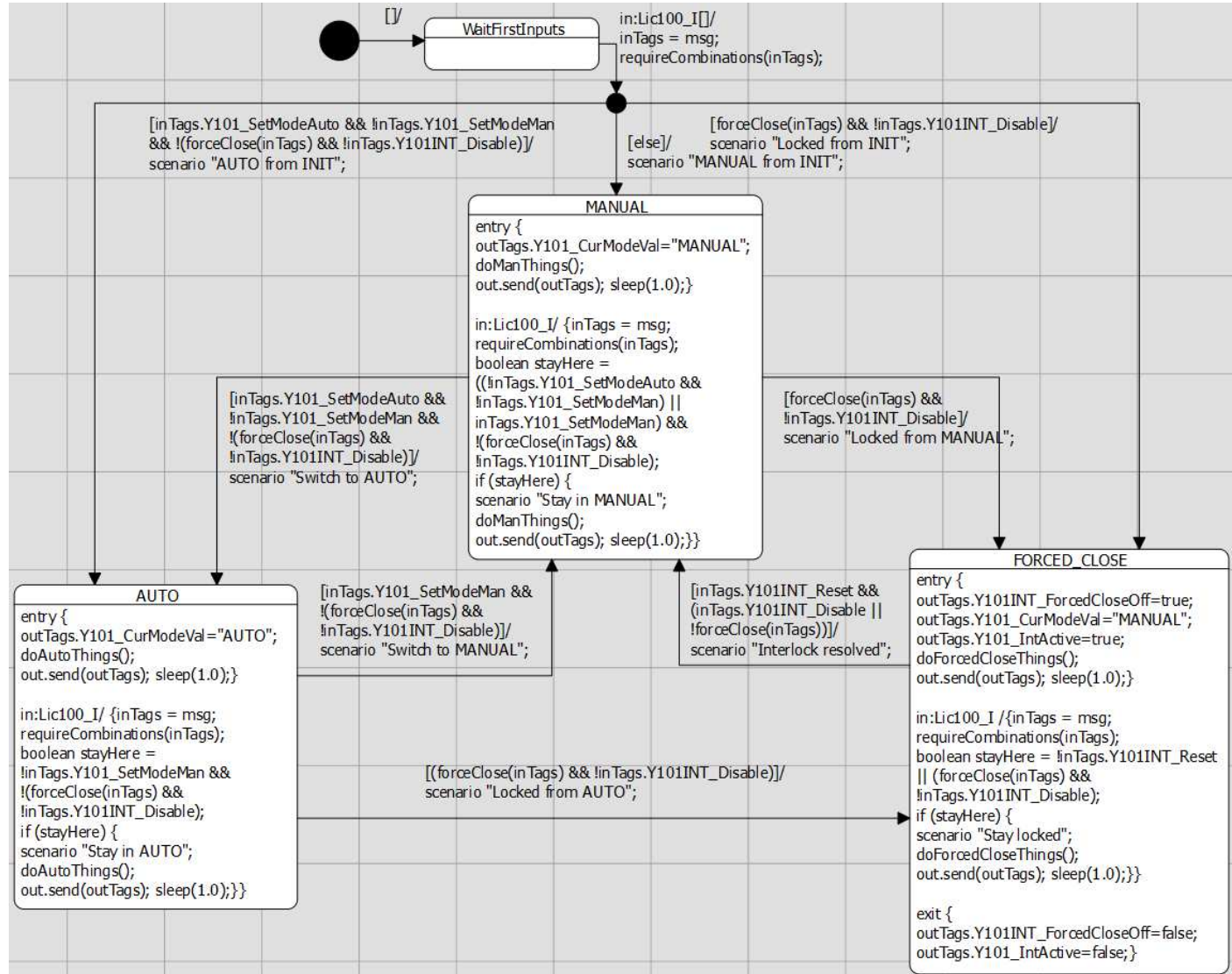
Device interlock

Testing Goals

- Validate discrete functionality
 - Alarms,
 - Interlocks,
 - Mode changes (AUTO, MAN, SEQ, LOCAL, FORCED ON/OFF)
- Discover execution cycle level anomalies
 - A wrong output from a control loop lasting one execution cycle
 - Most bugs appear during system discontinuities, e.g. when releasing interlocks back to normal operation.
 - E.g. Auto -> Forced control -> Auto etc...
 - May appear with design templates containing certain function blocks (FB), if not configured right.
 - Proportional-Integral-Derivative control FB
 - Motor control FB
 - Magnetic valve control FB

Test Model for Control Loop LIC100

- A subset of loop's functionality modeled for testing
- Abstraction level is low
→ Model looks complicated, but test adapter is straight forward.
- Cyclic SUT Execution:
 - 1) read inputs,
 - 2) execute function blocks,
 - 3) write outputs
- System interface:
 - one input port and message with 8 variables
 - one output port and message with 9 variables
- System states: AUTO, MANUAL, FORCED_CLOSE



Example of a Successful Test for Control Loop LIC100 (Filling Valve)

- Test Case 12 shows a bug
 - Step 1: interlock cause exists, but interlock feature is disabled. Loop control output is true/open. Loop stays in AUTO mode.
 - Step 2: interlock cause still exists, but now interlock feature is enabled. The loop's control output (Y101_CtrlOut) is forced to false/close, although controller (LIC100_CtrlOn) tries to keep it true/open. **Failure: The loop shows AUTO mode, although the interlock should have dropped it to MANUAL (= safe state).**
- (In principle, remaining in AUTO mode may be dangerous. For example, when process technician goes to solve the case and removes the interlock cause, automation could 'suddenly' reactivate the device.)
- Luckily, this failure in mode output disappeared in next execution cycle. Thus, the bug can be classified as a 'single cycle anomaly', which, however, may have other undesired consequences.
- A real bug. Not fabricated for demoing.

```
*** Starting TestCase num 12 ***
Locked from AUTO
Harness setUp()

TC12/Step 1
  in
    Harness sendCQRecord_Lic100_I()
    Checking sent data:
      L100_MeasIn: 0
      L101_MeasIn: false
      LIC100_SPVal: 0
      Y101_SetModeAuto: true
      Y101_SetModeMan: false
      Y101_ManCtrlVal: true
      Y101INT_Disable: true
      Y101INT_Reset: true
    Wrote ns=5;s=OpcSimTA.TB_RunTest

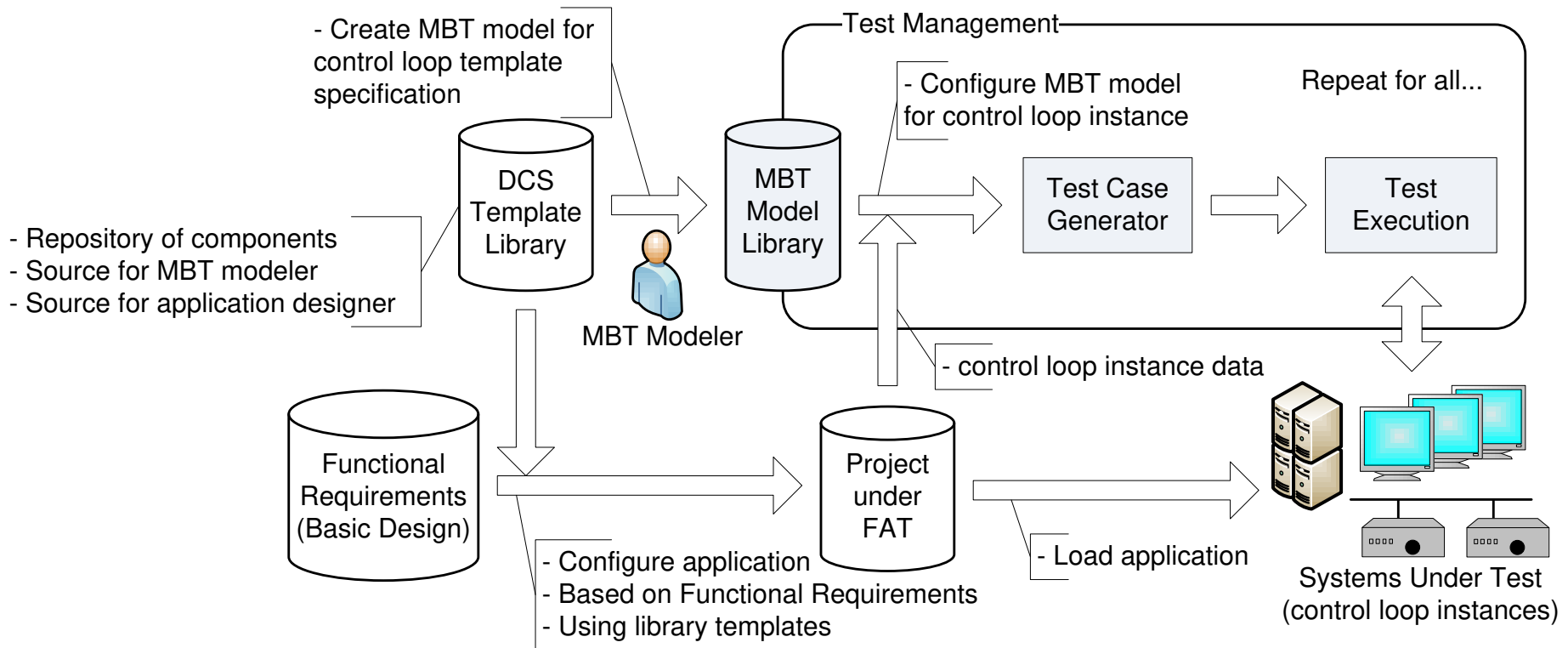
  out
    Harness receiveCQRecord_Lic100_Q()
    Checking received data: (SUT vs. Requirement)
      L100_MeasVal OK (0.0=0.0)
      L100_AlrmEvtHH OK (false=false)
      L101_MeasVal OK (false=false)
      L101_AlrmEvtOn OK (true=true)
      LIC100_CtrlOn OK (true=true)
      Y101_CurModeVal OK (AUTO=AUTO)
      Y101_CtrlOut OK (true=true)
      Y101_IntActive OK (false=false)
      Y101INT_ForcedCloseOff OK (false=false)

TC12/Step 2
  in
    Harness sendCQRecord_Lic100_I()
    Checking sent data:
      L100_MeasIn: 0
      L101_MeasIn: false
      LIC100_SPVal: 0
      Y101_SetModeAuto: false
      Y101_SetModeMan: false
      Y101_ManCtrlVal: true
      Y101INT_Disable: false
      Y101INT_Reset: true
    Wrote ns=5;s=OpcSimTA.TB_RunTest

  out
    Harness receiveCQRecord_Lic100_Q()
    Checking received data: (SUT vs. Requirement)
      L100_MeasVal OK (0.0=0.0)
      L100_AlrmEvtHH OK (false=false)
      L101_MeasVal OK (false=false)
      L101_AlrmEvtOn OK (true=true)
      LIC100_CtrlOn OK (true=true)
      Y101_CurModeVal FAIL! (AUTO<>MANUAL)
      Y101_CtrlOut OK (false=false)
      Y101_IntActive OK (true=true)
      Y101INT_ForcedCloseOff OK (true=true)
    Locked from AUTO (TC: 12) Harness tearDown()
    Last test case, disconnecting SUT.
    Disconnecting TestAdapter
```

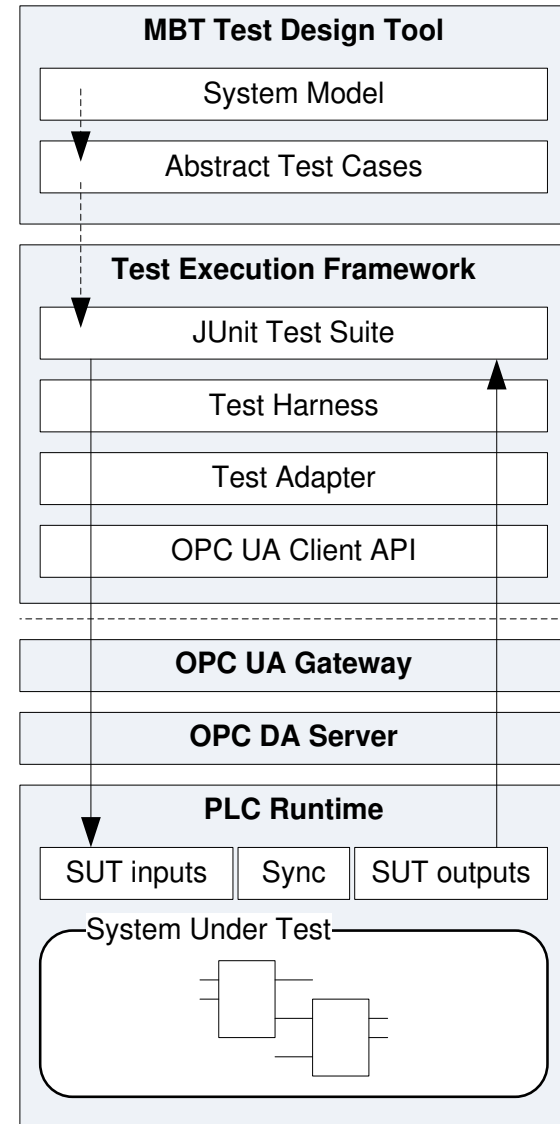
MBT for Factory Acceptance Testing (Model Reuse Concept)

- MBT modeling is done offline, based on Template library ('domain engineering')
- Library MBT model and tests are configured with control loop instance parameters ('application engineering').



Integrating MBT with Automatic Test Execution

- MBT Test Design Tool
 - Conformiq Designer
- Test Execution Framework
 - Generated test suite as JUnit code
 - Test suite imported in Eclipse Java project
 - Harness code linked to test adapter with OPC UA
- OPC Middleware
 - UA Gateway
 - DA Server from PLC manufacturer
- PLC Runtime
 - SUT interface is a set of global PLC variables accessed via PLC's OPC server
 - SUT is a PLC or DCS program or a set of programs under one cyclic execution task.
 - Execution cycle of the SUT is synchronized with test case steps (1 to 1).



Conclusions and Summary

- We have executed tests on process control loops and shown that the approach finds realistic bugs, often hard to find using manual methods.
 - Control loops in a Multiprog PLC (Aalto's process control loop library)
 - Control loops in MetsoDNA (Metso's template library)
- It is easy to conclude that the effort of modeling each control loop instance separately is non-feasible for FAT of 1000s of control loops.
- Control system's reuse approach (design templates) can be followed also by MBT : Model only once on template level and configure the model for (similar groups of) control loop instances
 - Basically, configuration information is retrieved from engineering system and used to generate parts of the MBT model's code (containing configuration parameters)
 - MBT model may also focus on some specific aspect/functionality of multiple templates. E.g. several templates share similar state behavior
- Final stage for customizing tests to control module instances is during test execution at test adaptation layer (data driven)
- Further steps:
 - Demonstrating MBT model configuration and/or data driven test execution
 - More complex control loop case to study the potential and limits of applicability
 - Allocating simple mass testing to keyword & data driven test automation

Thank you!
Questions?

elec.aalto.fi/

www.metsoautomation.com/

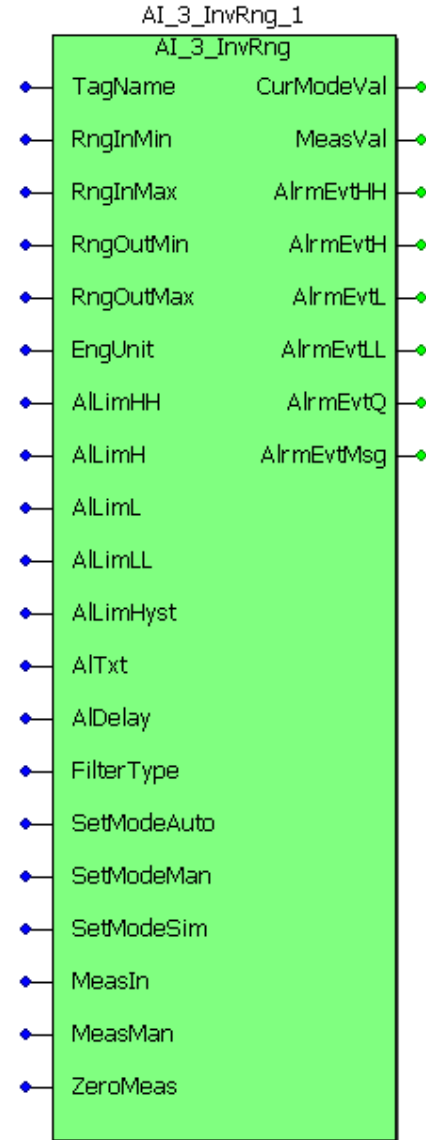
conformiq.com/

tekes.fi/en/



Test scope: Function Block, Case FB type AI_3

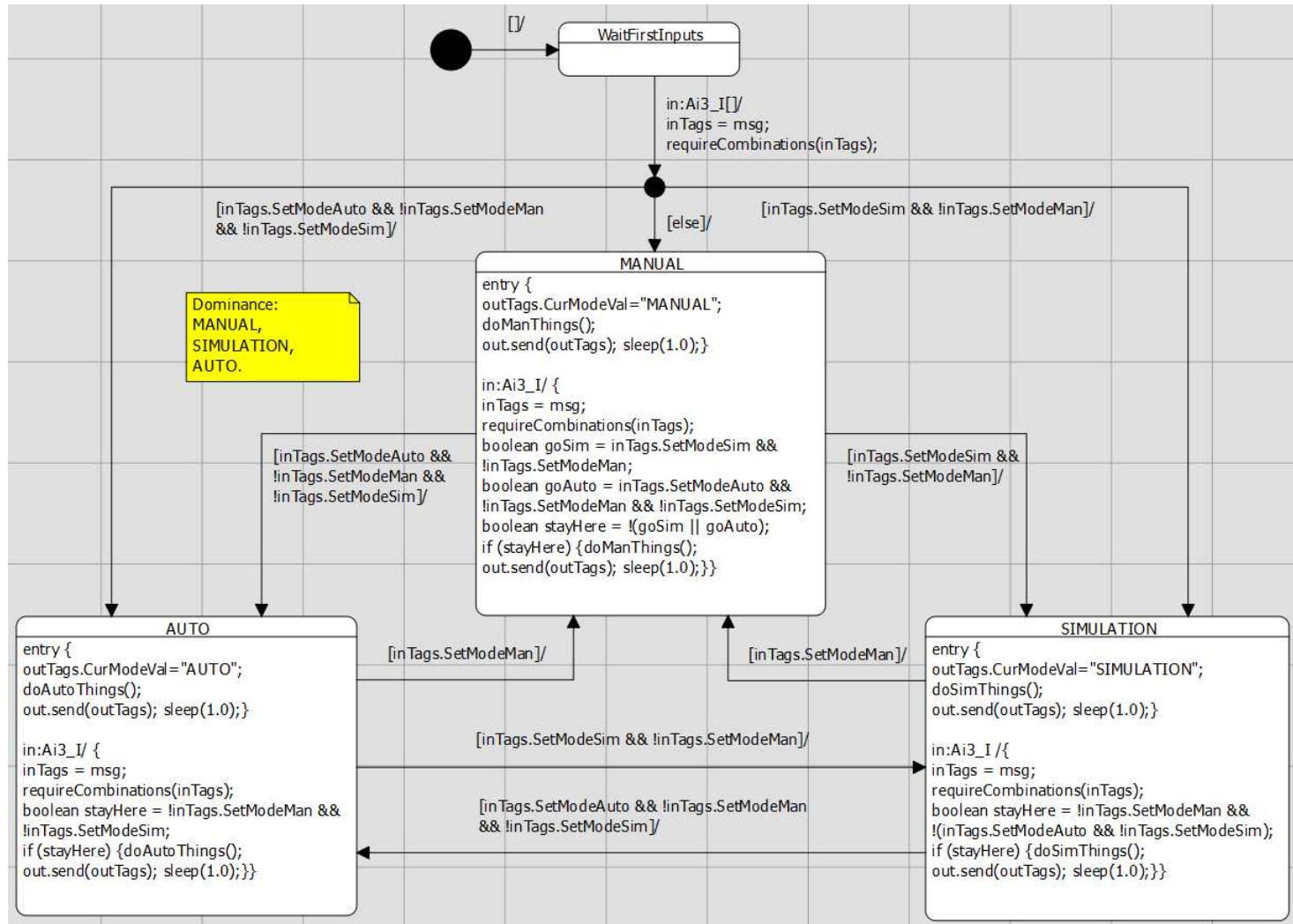
- Analog input FB type has the purpose of receiving a measurement value from a specified global variable (process interface) and providing it as a (named) scaled signal to the automation application. This type also supports alarms, filtering, auto/man/sim modes and zeroing.



Port name	Port usage	Port data type	Parameter	Description
TagName	VAR_INPUT	STRING	P	Name of the circuit. E.g. "L100".
RngInMin	VAR_INPUT	REAL	P	Source signal range minimum
RngInMax	VAR_INPUT	REAL	P	Source signal range maximum
RngOutMin	VAR_INPUT	REAL	P	Measurement range minimum in engineering units
RngOutMax	VAR_INPUT	REAL	P	Measurement range maximum in engineering units
EngUnit	VAR_INPUT	STRING	P	Unit of produced measurement value
ALimHH	VAR_INPUT	REAL	P	Alarm limit in engineering units
ALimH	VAR_INPUT	REAL	P	-"-
ALimL	VAR_INPUT	REAL	P	-"-
ALimLL	VAR_INPUT	REAL	P	-"-
ALimHyst	VAR_INPUT	REAL	P	Alarm is set off when value is this much on the safe side of the limit.
AITxt	VAR_INPUT	STRING	P	Text string used to parsing Alarm message output.
AIDelay	VAR_INPUT	TIME	P	Delay for alarm to trigger
FilterType	VAR_INPUT	INT	P	Value selects filtering mechanism, 0: no filtering, 1: reserved, 2: reserved, N: average of N consecutive samples.
SetModeAuto	VAR_INPUT	BOOL		Automatic mode when rizing edge detected
SetModeMan	VAR_INPUT	BOOL		Manual mode when rizing edge detected
SetModeSim	VAR_INPUT	BOOL		Simulation mode when true, else auto or man mode.
MeasIn	VAR_INPUT	UINT		Source analogue input channel for measurement data.
MeasMan	VAR_INPUT	REAL		Measurement value, when in MAN mode
ZeroMeas	VAR_INPUT	BOOL		Force output to zero when true
CurModeVal	VAR_OUTPUT	STRING		Shows current operating mode
MeasVal	VAR_OUTPUT	REAL		Measurement value
AlrmEvtHH	VAR_OUTPUT	BOOL		Alarm is on (MeasVal>=hh)
AlrmEvtH	VAR_OUTPUT	BOOL		Alarm is on
AlrmEvtL	VAR_OUTPUT	BOOL		Alarm is on
AlrmEvtLL	VAR_OUTPUT	BOOL		Alarm is on
AlrmEvtQ	VAR_OUTPUT	BOOL		Alarm for measured signal's quality
AlrmEvtMsg	VAR_OUTPUT	STRING		Alarm message

Test Model for AI_3 (analog measurement composite FB)

- First model "Hello MBT World!"
- Low abstraction, because SUT interface (FB IOs / OPC variables) was the mental starting point.



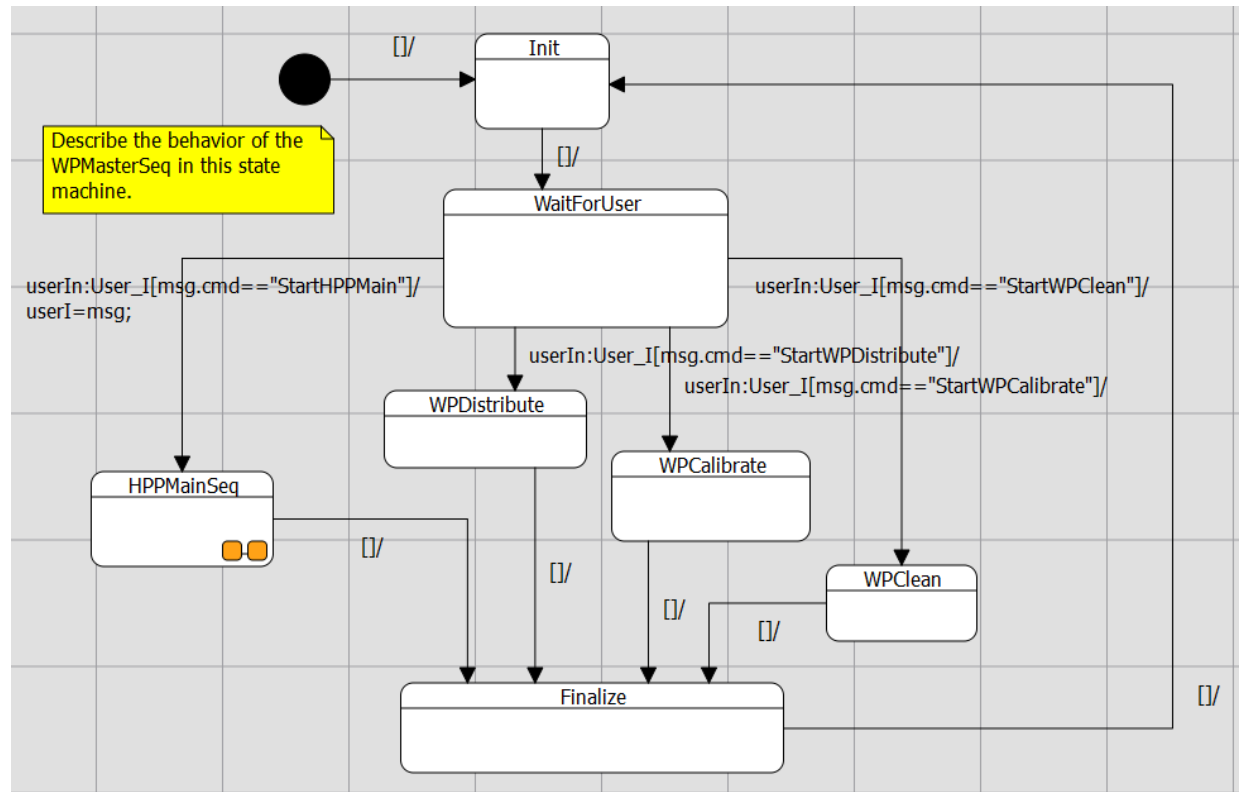
Control Loop LIC100 Interface and Complexity Metrics

- In: number of FB inputs (In=Par+Dyn)
- Par = Parameter inputs
- Dyn = Dynamic inputs (change during runtime)
- Out = number of FB outputs
- Out2 = relevant outputs (e.g. string messages excluded)
- FBs = number of contained Function Blocks
- CONs = number of connections / "wires"
- LOC = lines of code in user blocks implemented with ST
- FBs/IOs = FBs/(Dyn+Out2), a complexity metric of a block

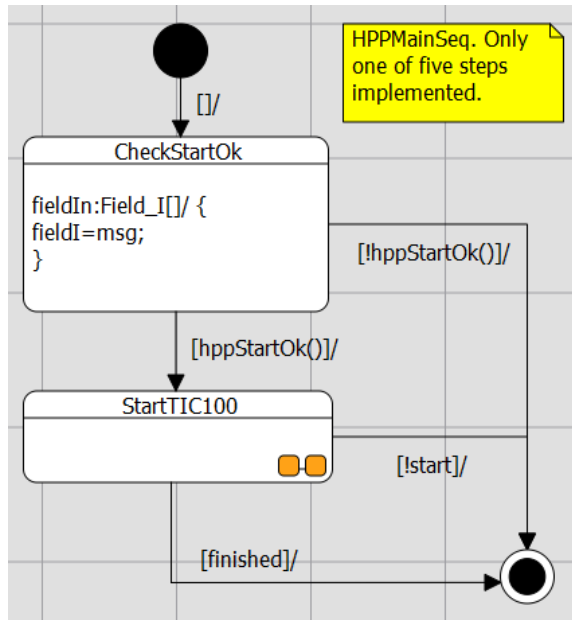
Component		In			Out		Complexity			
Tag Name	POU Type	In	Par	Dyn	Out	Out2	FBs	CONs	LOC	FBs/IOs
L100	AI_3	20	14	6	8	6	45	176	225	3,8
L101	BI_2	10	6	4	4	2	24	78	60	4,0
LIC100	LC_3	15	7	8	5	2	24	72	27	2,4
Y101	OOA_3	14	4	10	7	3	13	55	27	1,0
Y101INT	INT_3	21	7	14	6	4	8	27	0	0,4
LIC100PROG	Composite	80	38	42	30	17	5	140	0	
LIC100PROG	Test Interface			8		9	5			0,3

Test Model for WP Master Sequence

- First model which starts almost purely from thinking of the required behavior, instead of an existing SUT interface.
- → "Test Driven Development"
- Modeler's thinking rises more easily to a more abstract level. That is, above PLC variables.
- If S88 kind of interface existed in modelers thinking, that would probably be used in the model.
- (HPP Main Sequence is embedded in a state.)



Test Model for WP Master Sequence



- Sequence hierarchy: Master, Main, Step
- Trying to build generic models for Main and Step. Then reuse them.
- UML State Chart notation works well in this scenario
- This SUT has no implementation. The model is a draft.

