Safety Software Engineering

Peter Sieber, HIMA Paul Hildebrandt GmbH

VP Norms & Standards

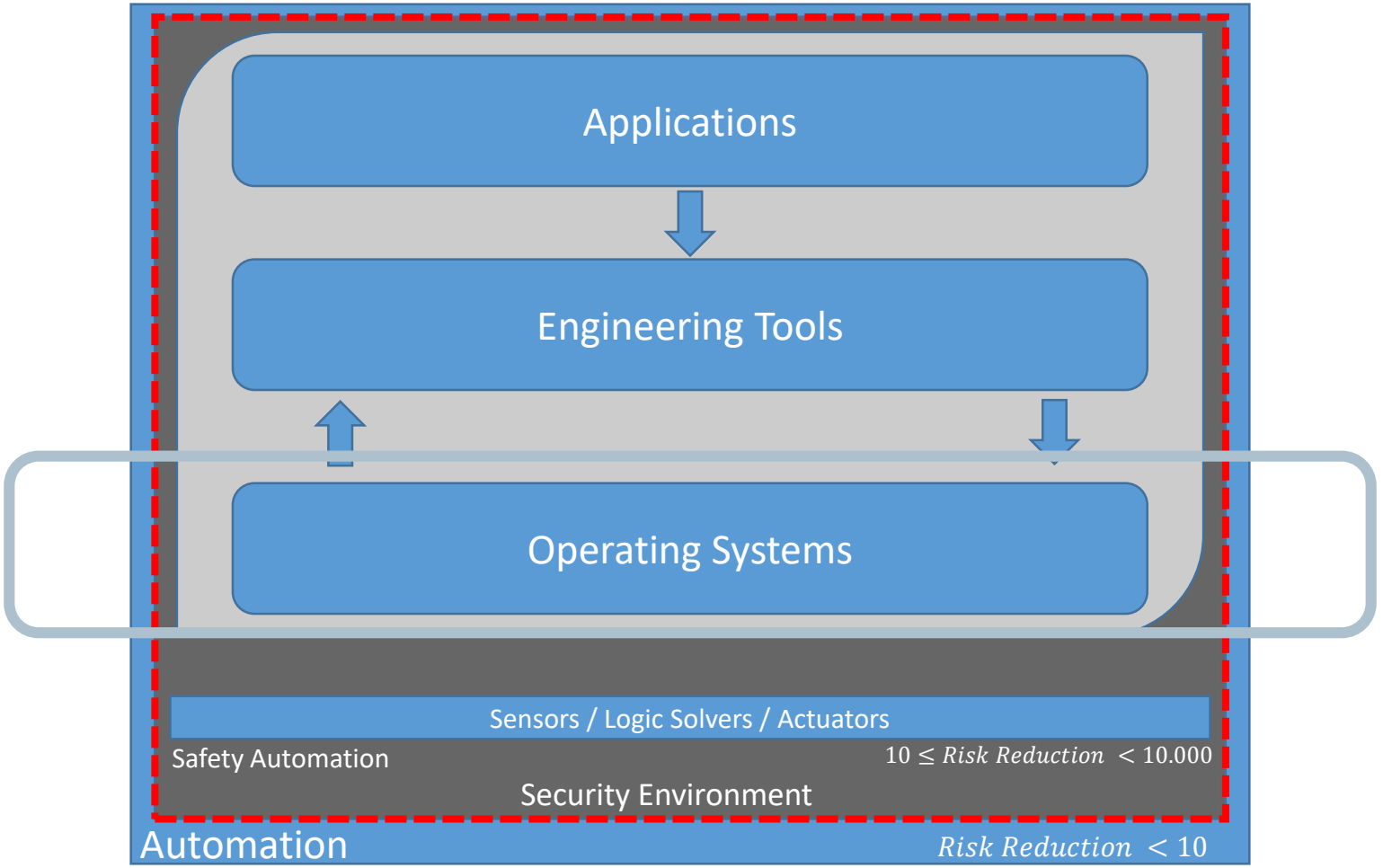
VP Region China

Safety Case
Symposium 2019
Singapore
Mar 26 - 27, 2019

Agenda

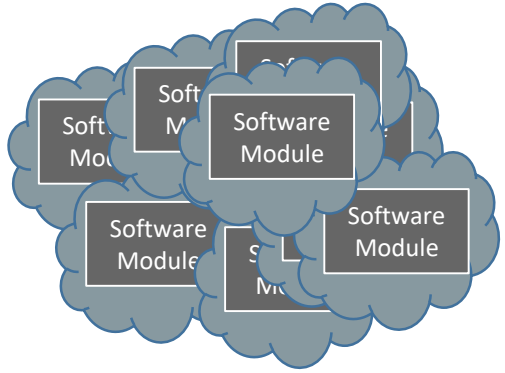
1. Software Development
2. Engineering Tools
3. Safety Applications
4. Application Program Functional Design
5. Instancing of pre-tested Modules
6. Validation Planning and Testing
7. Testing of Applications
8. Conclusion

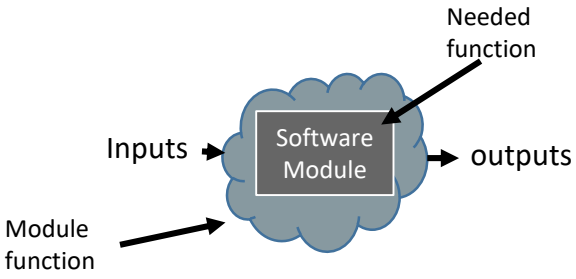
High-End Software?



Common practice of “Software development”

- Do a functional System specification
- Define the System Architecture
- Evaluate market available Modules
- Select & purchase Modules
- Select needed module functions

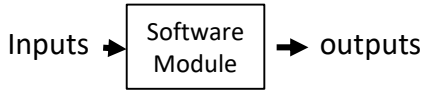


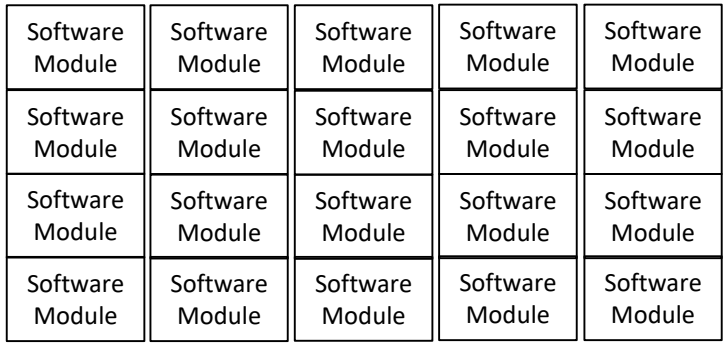


- Specification of modules as available
- Quality of the modules as available
- Modules having unused features on board
- “Hidden” functions creating additional risk
- Overall complexity increased

HIMA process to “develop” Software

- Do a functional System specification
- Define the System Architecture
- Design the architecture test specification
- Design the module Specification
- Design the module test specification





- Specification of all modules as per FSM
- Modules having ***no*** hidden features on board (We know what we get!)
- Module design fully under control
- Complexity minimized

Software Quality and how to cope with it

Common practice

- Define functional environment
- Start testing
- Document and ***process*** recognized errors
- Test till statistical benchmark is reached

Software is released if

$$\frac{\sum_{errors\ recognized\ per\ week}}{\sum_{all\ errors\ recognized}} < Benchmark$$

Reliability Objective:

Reach a statistical Benchmark for systematic Errors (!)

Patches in weeks (days!?!)

HIMA process

- Define test plan covering 100% of specified function (each line of code)
- Execute Tests
- Document executed Tests

Software is released if

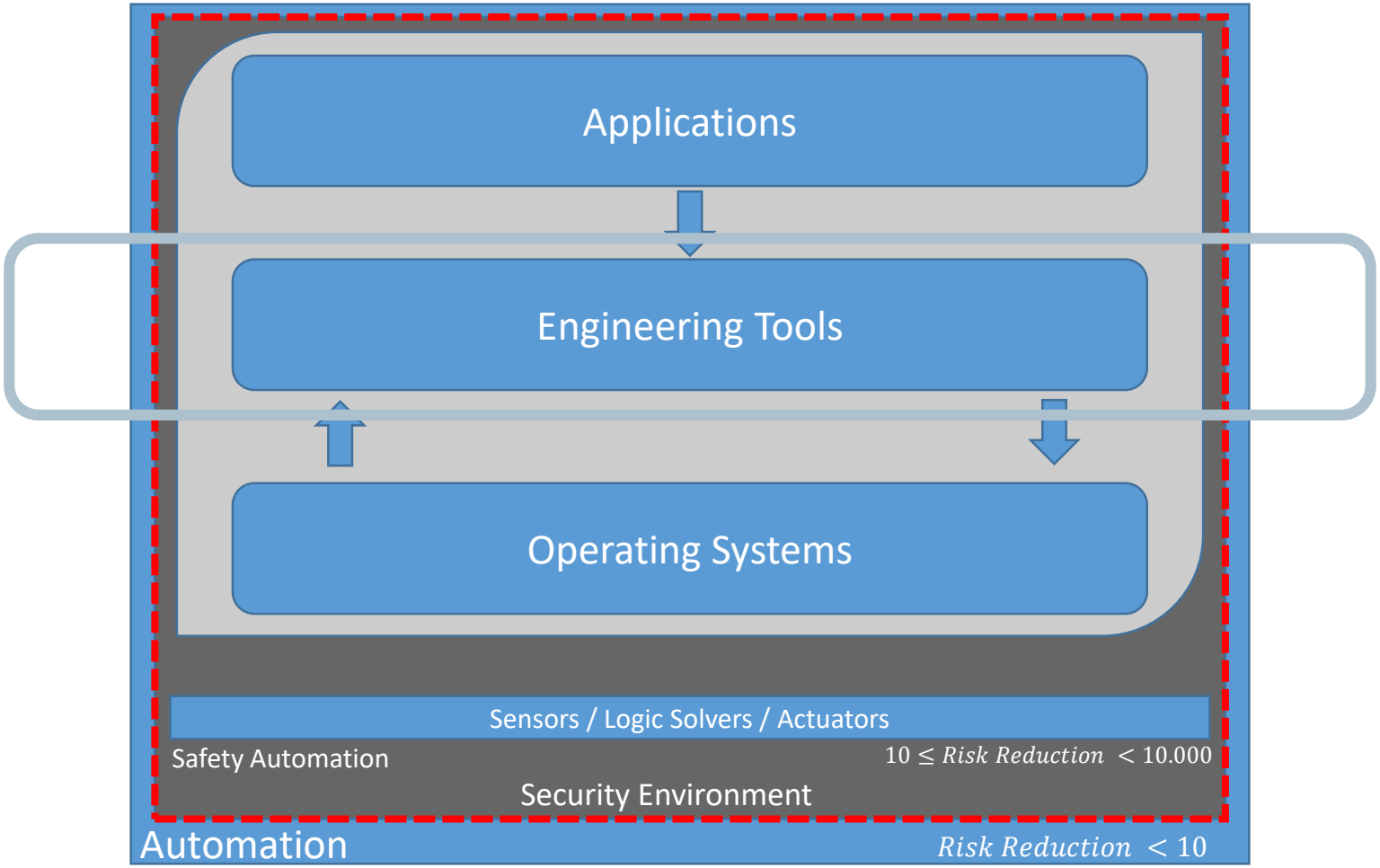
- Test program is executed
- All errors found are ***processed***
- Error fixes got re-tested

Reliability Objective:

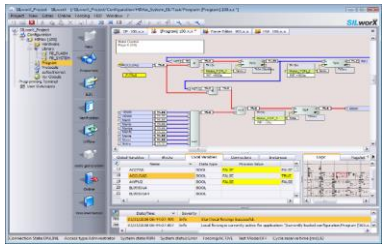
Software free of deviations from spec

Updates in Years

High-End Software Engineering Tool



Engineering Tools for High-End Software



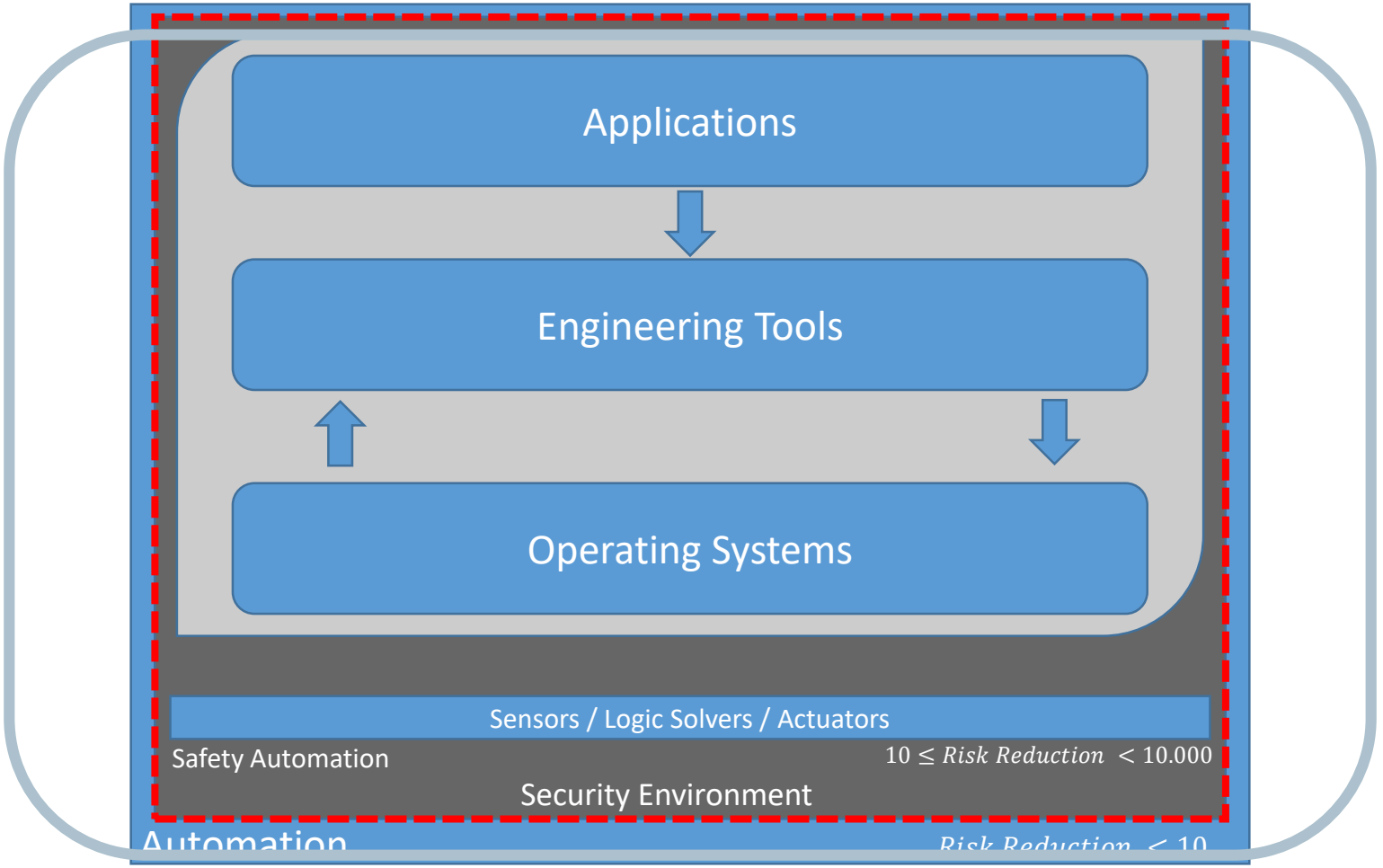
Challenges to be mastered:

- Need to use a COTS Platform
- Need to define kind and usage of operating system
- Need to harden Software and data against manipulations
- Need to implement an Audit trail
- Need to implement a User Management
- Need to develop an IP protection concept

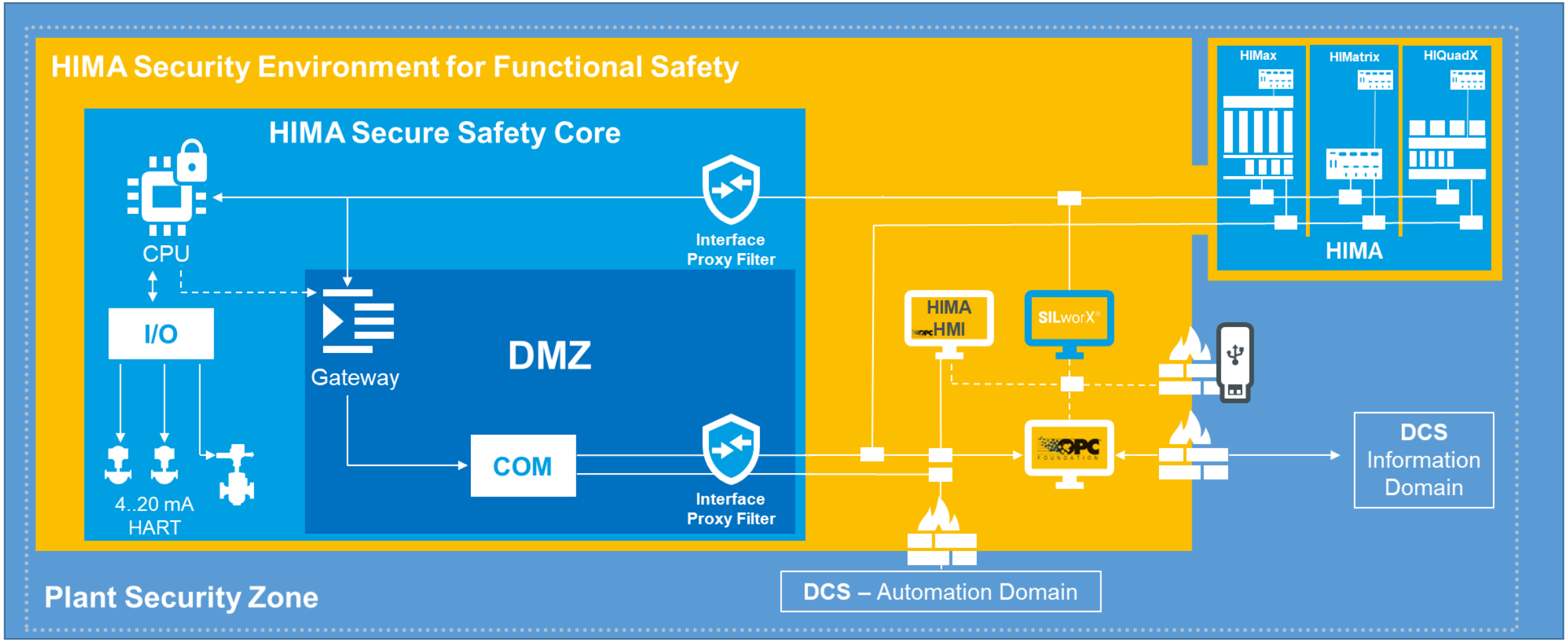
Tool made available is:

- Developed following SIL recommendations
- Having SIL 3 capable SW functions
  capable to deliver SIL 3 compliant SW
- Covering design as well as testing
- Integrated part of a SIL compliant engineering workflow

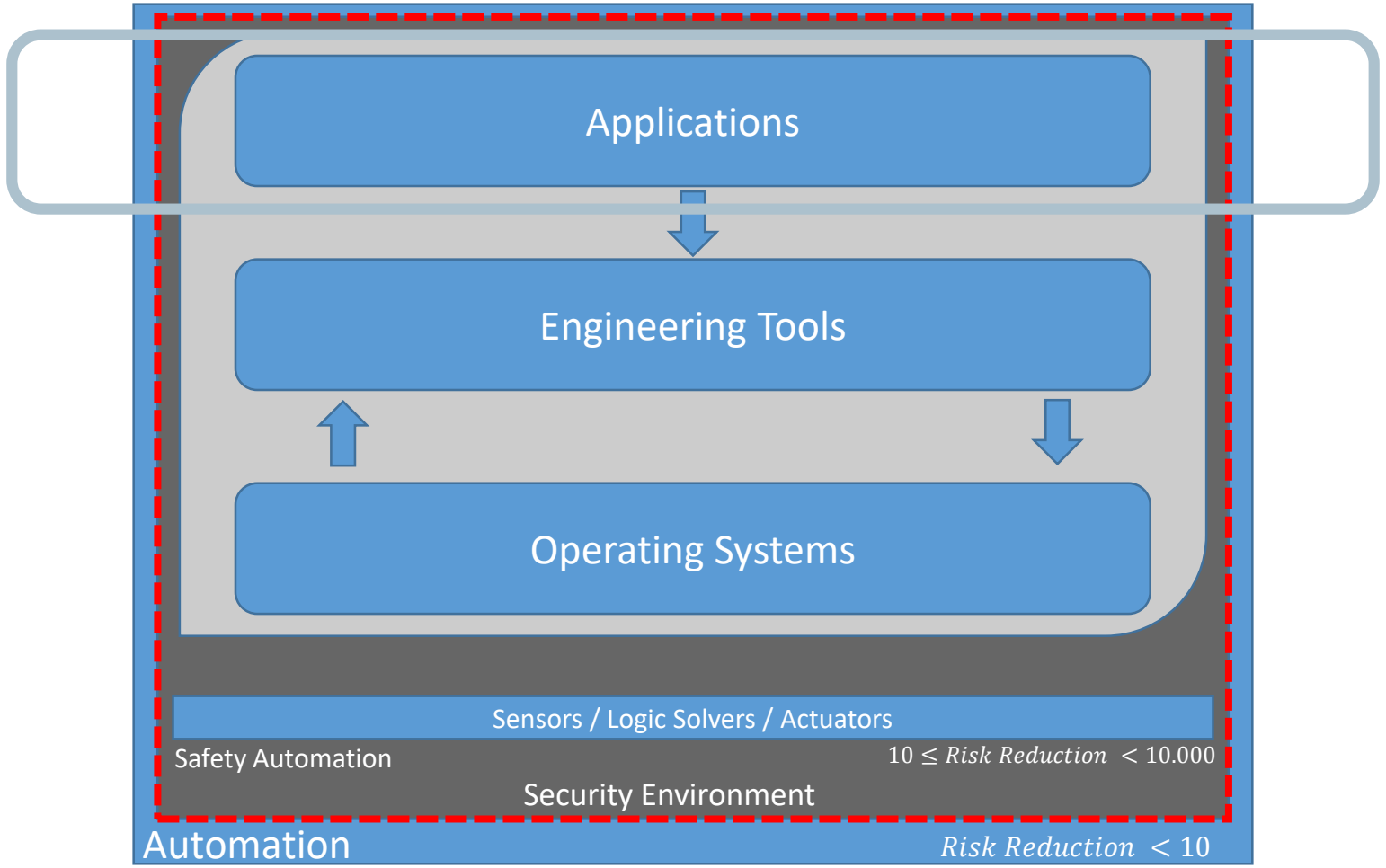
High-End Software: Applications



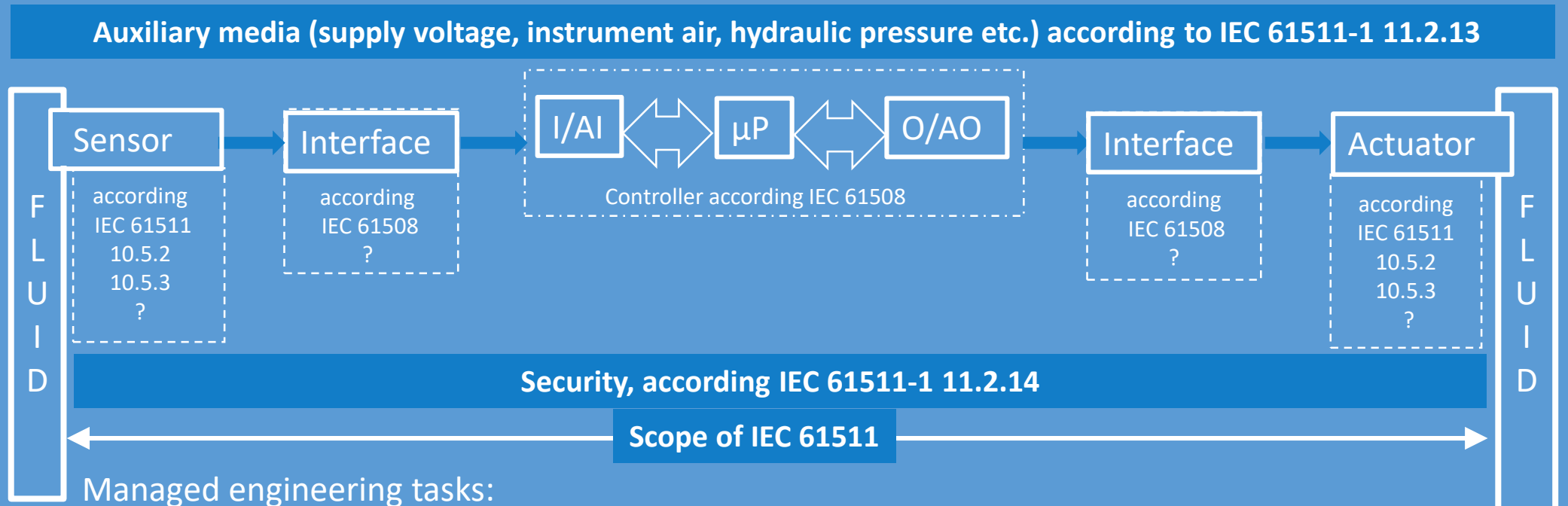
High-End Software: Security



High-End Software: Applications



Safety Engineering as per IEC 61511 ed. 2

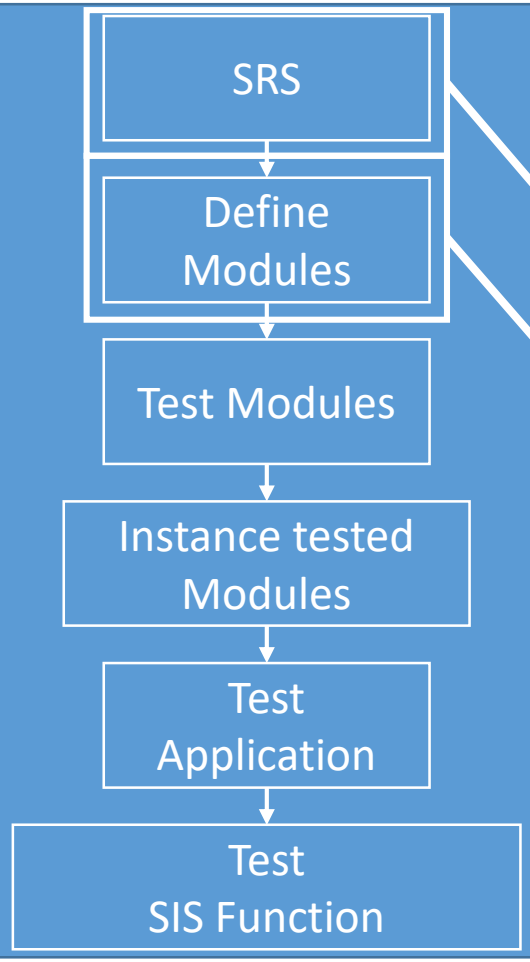


Managed engineering tasks:

- Define function
- Select & qualify equipment
- Select, implement & qualify functions

IEC 61511-1 10.5.2: Selection of devices
IEC 61511-1 10.5.3: Selection of devices based on prior use

Application Development Process



		Effects					
	TAG	PMV35001A	PMV35001B	PMV35002	PMV35003	PMV35004	
Causes TAG	SIL	3		2	2	2	
PICZA35002_A	3	I		I	I	I	
PICZA35002_B							
PICZA35002_C							
TICZA35002_A	3	II		I	I	I	
TICZA35002_B							
TICZA35002_C							
GI35001AC	3	I		C1	C1	O1	
GI35001BC	3	I		C1	C1	O1	
GI35001CC	3	I		C1	C1	O1	
LICZA35101_A	3	II		II	II	II	
LICZA35101_B							
LICZA35101_C							

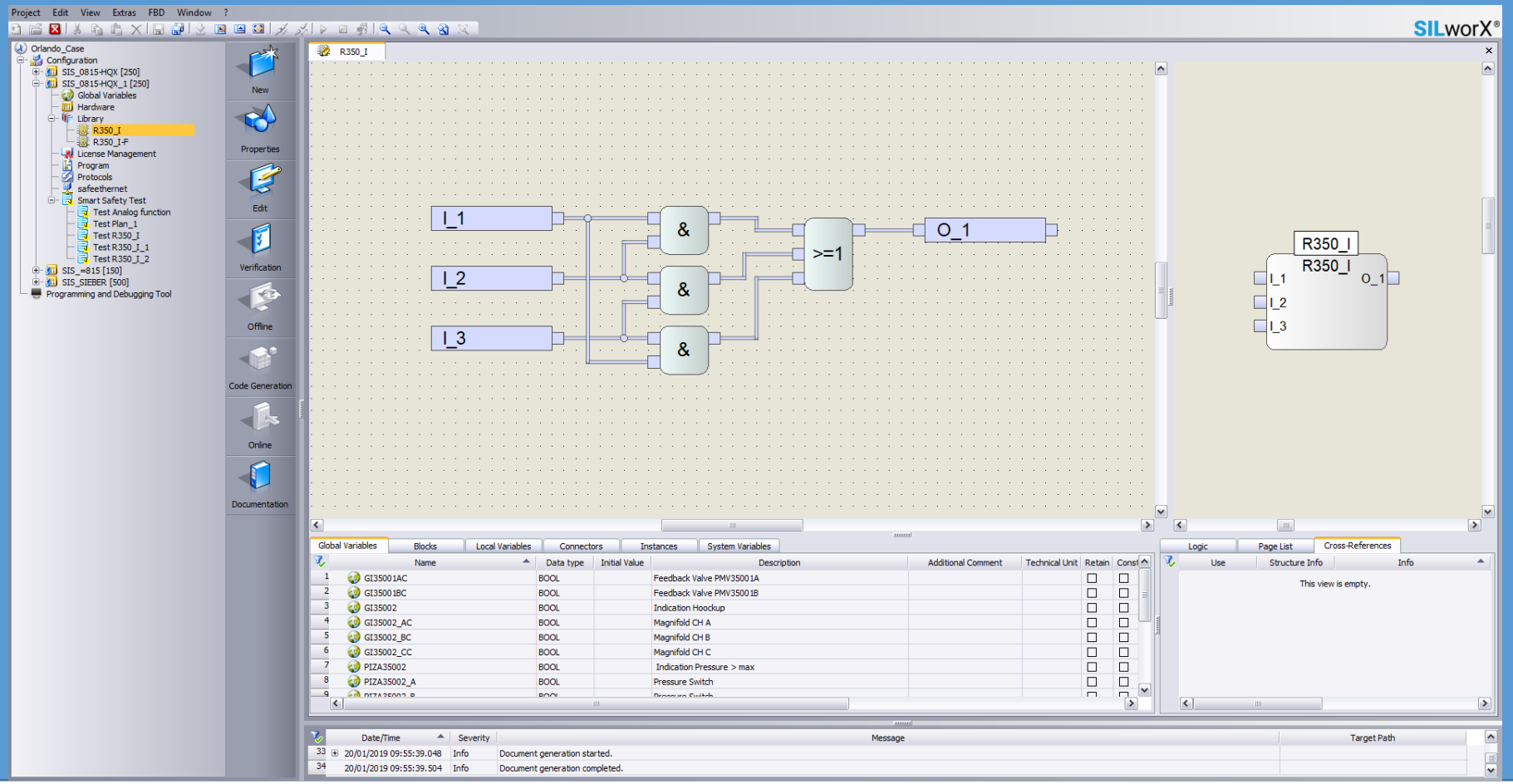
Legend

I	2oo3 Input switch off when trip point **Low**
II	2oo3 Analog Input switch
O1	Monitor Valve travel; open if too slow
C1	Monitor Valve travel; close if too slow

For further details please refer to IEC 61511-1 10.3.1

Implementation of Software Modules

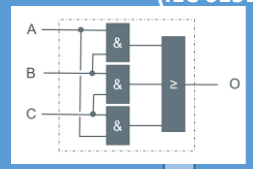
- Typical function either in
  - FBD
  - ST
  - C++
- Different languages can be combined



How to Validate Software Modules

SRS

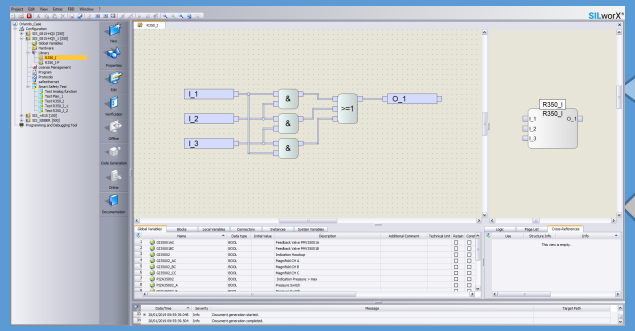
(IEC 61511-1 10.3.2)



	A	B	C	O
1	0	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	1
7	1	1	0	1
8	1	1	1	1

Application Module

(IEC 61511-1 12.1 to 12.3)

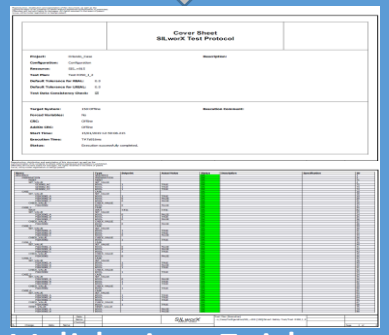


Validated modules can be protected against (unwanted) changes

In SIL 3 Quality!

SILworX®

Engineering Tool

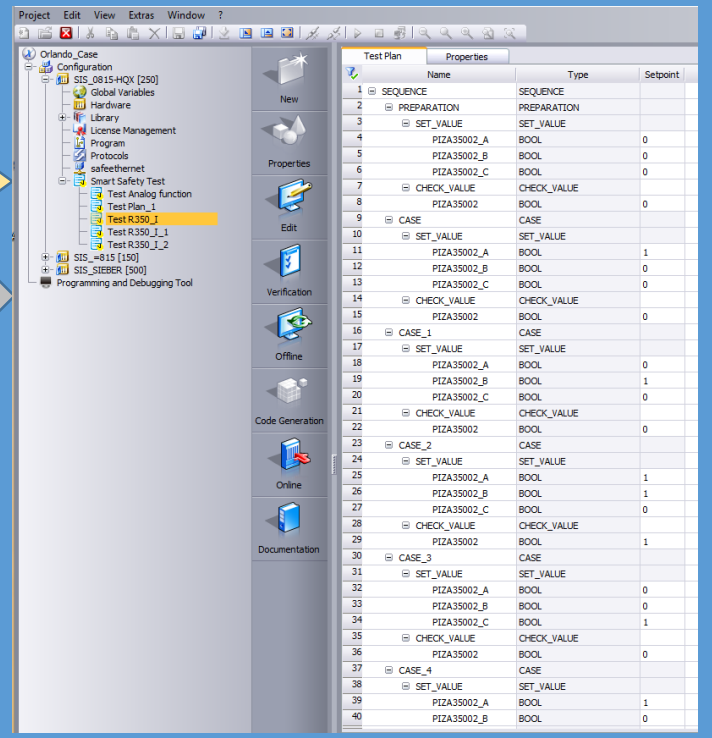


Validation Evidence

IEC 61511-1 15.2.4

Validation Planning

(IEC 61511-1 15.2.2)



Tasks to Handle Validation Plans

- Define baseline at SILworX
- Export to flat file (CSV)
- Modify flat file as required
- Re-import into SILworX

Name	Type	Setpoint	Description	Specification
SEQUENCE	SEQUENCE			
SEQUENCE.PREPARATION	PREPARATION			
SEQUENCE.PREPARATION.RESET	RESET			
SEQUENCE.PREPARATION.SET_VALUE	SET_VALUE			
SEQUENCE.PREPARATION.SET_VALUE.GI35002_AC	BOOL	1	2 00 3 Input A	1= "ON"
SEQUENCE.PREPARATION.SET_VALUE.GI35002_BC	BOOL	1	2 00 3 Input B	1= "ON"
SEQUENCE.PREPARATION.SET_VALUE.GI35002_CC	BOOL	1	2 00 3 Input C	1= "ON"
SEQUENCE.CASE	CASE			
SEQUENCE.CASE.SET_VALUE	SET_VALUE			
SEQUENCE.CASE.SET_VALUE.PIZA35002_A	BOOL	1	2 00 3 Input A	1= "ON"
SEQUENCE.CASE.SET_VALUE.PIZA35002_B	BOOL	0	2 00 3 Input B	1= "ON"
SEQUENCE.CASE.SET_VALUE.PIZA35002_C	BOOL	0	2 00 3 Input C	1= "ON"
SEQUENCE.CASE.CHECK_VALUE	CHECK_VALUE			
SEQUENCE.CASE.CHECK_VALUE.PIZA35002	BOOL	0	Check Output	1= "ON"
SEQUENCE.CASE_1	CASE			
SEQUENCE.CASE_1.WAIT	WAIT	T#5s	Wait to stabilize	
SEQUENCE.CASE_1.SET_VALUE	SET_VALUE			
SEQUENCE.CASE_1.SET_VALUE.PIZA35002_A	BOOL	0	2 00 3 Input A	1= "ON"
SEQUENCE.CASE_1.SET_VALUE.PIZA35002_B	BOOL	1	2 00 3 Input B	1= "ON"
SEQUENCE.CASE_1.SET_VALUE.PIZA35002_C	BOOL	0	2 00 3 Input C	1= "ON"
SEQUENCE.CASE_1.CHECK_VALUE	CHECK_VALUE			
SEQUENCE.CASE_1.CHECK_VALUE.PIZA35002	BOOL	0	Check Output	1= "ON"
SEQUENCE.CASE_2	CASE			
SEQUENCE.CASE_2.SET_VALUE	SET_VALUE			
SEQUENCE.CASE_2.SET_VALUE.PIZA35002_A	BOOL	1	2 00 3 Input A	1= "ON"
SEQUENCE.CASE_2.SET_VALUE.PIZA35002_B	BOOL	1	2 00 3 Input B	1= "ON"
SEQUENCE.CASE_2.SET_VALUE.PIZA35002_C	BOOL	0	2 00 3 Input C	1= "ON"
SEQUENCE.CASE_2.CHECK_VALUE	CHECK_VALUE			
SEQUENCE.CASE_2.CHECK_VALUE.PIZA35002	BOOL	1	Check Output	1= "ON"
SEQUENCE.CASE_3	CASE			
SEQUENCE.CASE_3.SET_VALUE	SET_VALUE			
SEQUENCE.CASE_3.SET_VALUE.PIZA35002_A	BOOL	0	2 00 3 Input A	1= "ON"
SEQUENCE.CASE_3.SET_VALUE.PIZA35002_B	BOOL	0	2 00 3 Input B	1= "ON"
SEQUENCE.CASE_3.SET_VALUE.PIZA35002_C	BOOL	1	2 00 3 Input C	1= "ON"
SEQUENCE.CASE_3.CHECK_VALUE	CHECK_VALUE			
SEQUENCE.CASE_3.CHECK_VALUE.PIZA35002	BOOL	0	Check Output	1= "ON"

Set pre-conditions

Execute test case

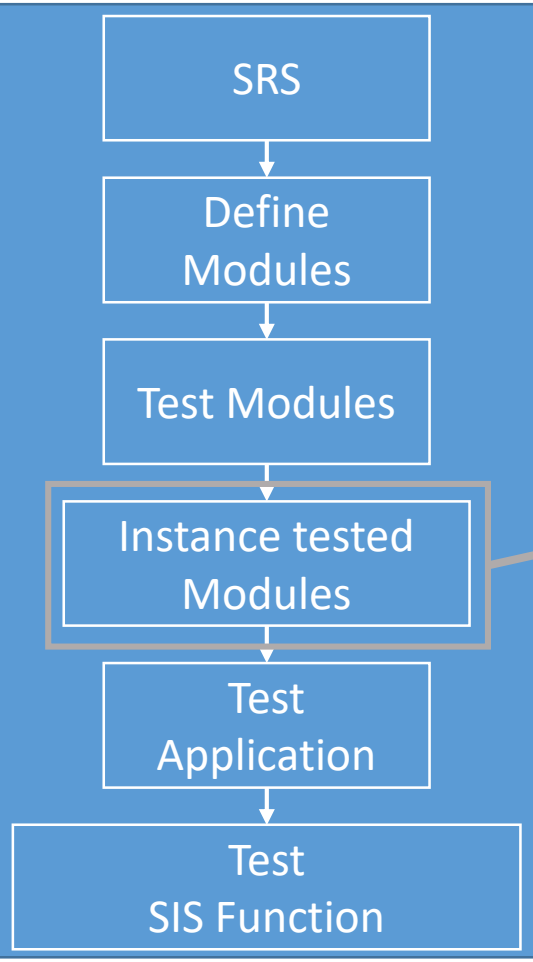
Wait state, optional

Execute test case

Execute test case

Execute test case

Instancing Process



				Effects					
		TAG		PMV35001A	PMV35001B	PMV35002	PMV35003	PMV35004	
Causes	TAG	SIL	Typical	3		2	2	2	
				1		2	2	2	
	PICZA35002_A	3	A	I		I	I	I	
	PICZA35002_B								
	PICZA35002_C								
	TICZA35002_A	3	B	I		I	I	I	
	TICZA35002_B								
	TICZA35002_C								
	GI35001AC	3	C	O1		C1	C1	O1	
	GI35001BC	3	C	O1		C1	C1	O1	
	LICZA35101_A	3	D	II		II	II	II	
	LICZA35101_B								
	LICZA35101_C								
	GI35002AC	3	C	C2					
	GI35002BC			C2					
	GI35002CC			C2					

Legend

I	2oo3 Input switch off when trip point **High** or 2003 violended
II	2oo3 Input switch off when trip point **Low** or 2003 violended
O1	Monitor Valve travel; open if too slow
C1	Monitor Valve travel; close if too slow
C2	digital 2003, open when triggered

How to Instance Modules

Create a Table by exporting a flat file

- Name Instances
- Assign I/O Tags
- Re-Import the Table into SILworX

POU Name	Page Positio	Page Name Page 0/0	Description	Drawing Number	Instance Pos: 65/30	Value Field 42/41	Value Field 42/46	Value Field 42/51	Value Field 83/42
R350_I	36/24	40				PIZA35002_A	PIZA35002_A	PIZA35002_A	PIZA35002
R350_I	53/52	40				GI35002_AC	GI35002_BC	GI35002_CC	GI35002
350_II	36/24	60				LICZA35101A	LICZA35101B	LICZA35101C	LICZA35101
350_I	36/25	80				PICZA10001A	PICZA10001B	PICZA10001C	PICZA10001
R350_I	36/24	140				PIZA45002_A	PIZA45002_A	PIZA45002_A	PIZA45002
R350_I	53/52	140				GI45002_AC	GI45002_BC	GI45002_CC	GI45002
350_II	36/24	160				LICZA45101A	LICZA45101B	LICZA45101C	LICZA45101
350_I	36/25	180				PICZA20001A	PICZA20001B	PICZA20001C	PICZA20001
R350_I	36/24	240				PIZA55002_A	PIZA55002_A	PIZA55002_A	PIZA55002
R350_I	53/52	240				GI55002_AC	GI55002_BC	GI55002_CC	GI55002
350_II	36/24	260				LICZA55101A	LICZA55101B	LICZA55101C	LICZA55101
350_I	36/25	280				PICZA30001A	PICZA30001B	PICZA30001C	PICZA30001

SILworX will

- Create the instances
- Assign I/O
- Check data for inconsistencies
- Create the related logic

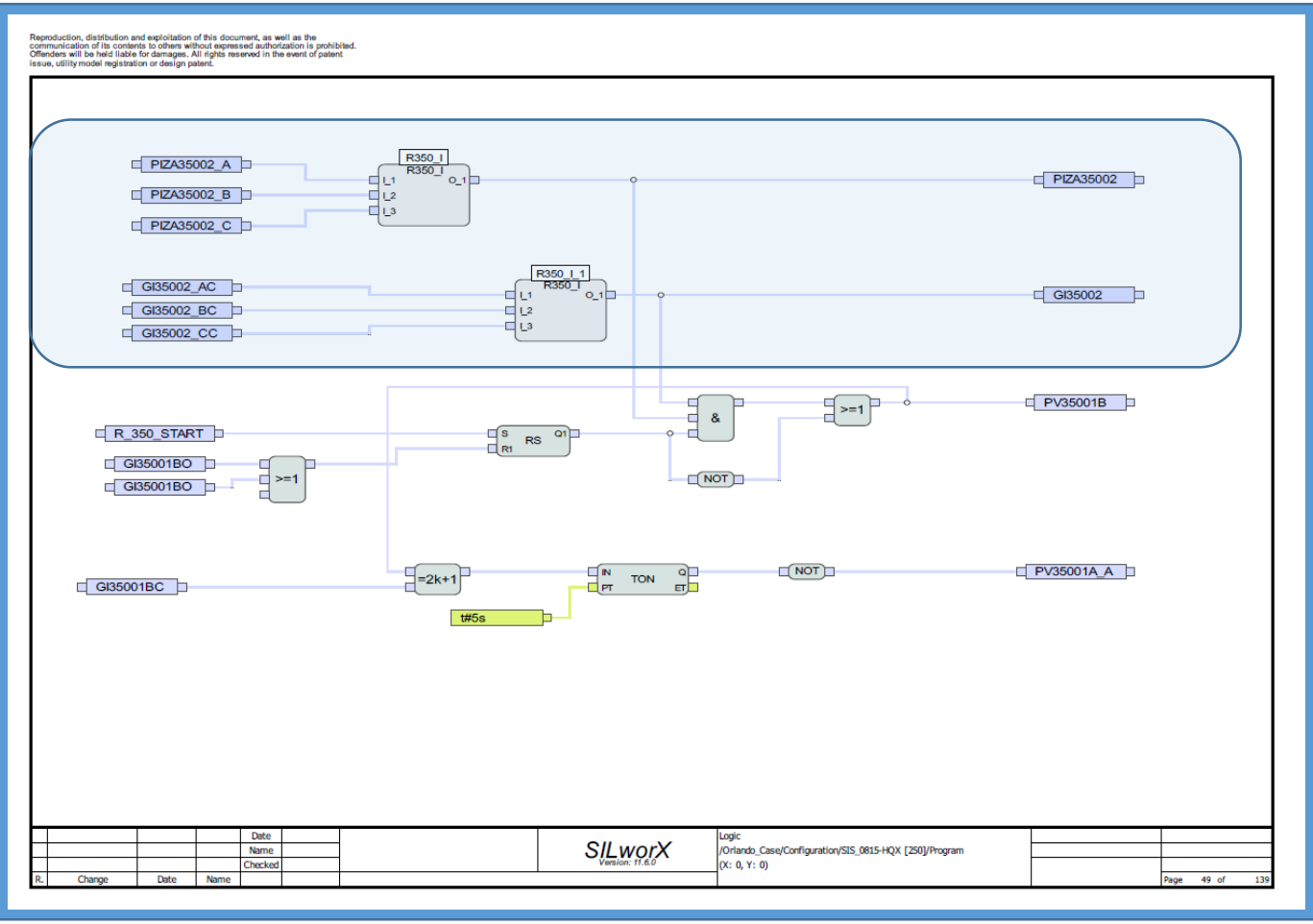
In SIL 3 Quality!

Modules Instanced with SILworX

SILworX will

- Create the instances
- Assign I/O
- Check data for inconsistencies
- Create the related logic

In SIL 3 Quality



Application Software Validation Process

SRS

Define Modules

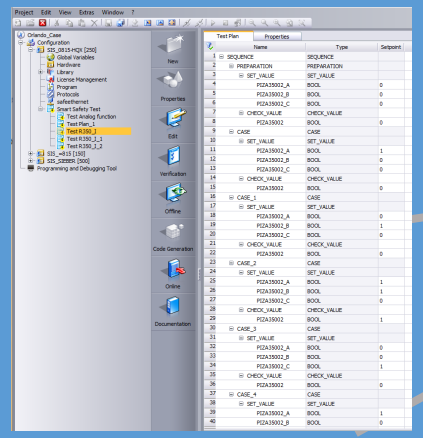
Test Modules

Instance tested Modules

Test Application

Test SIS Function

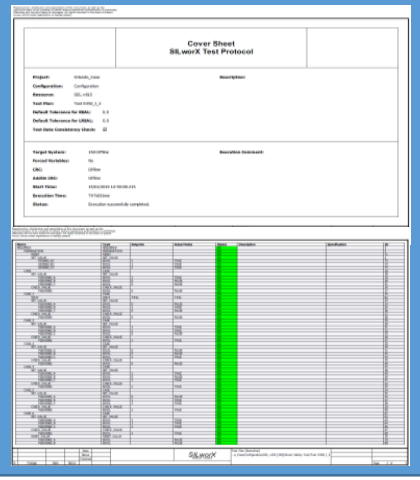
Validation Plan

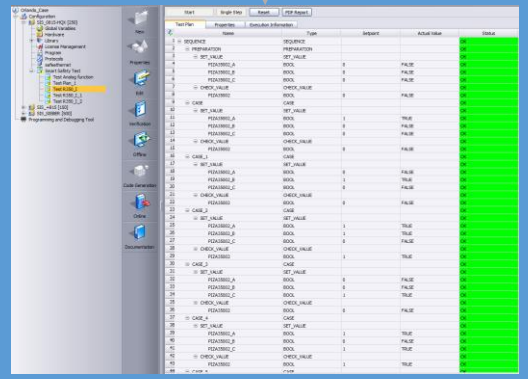


SILworX

Engineering Tool

Validation Evidence

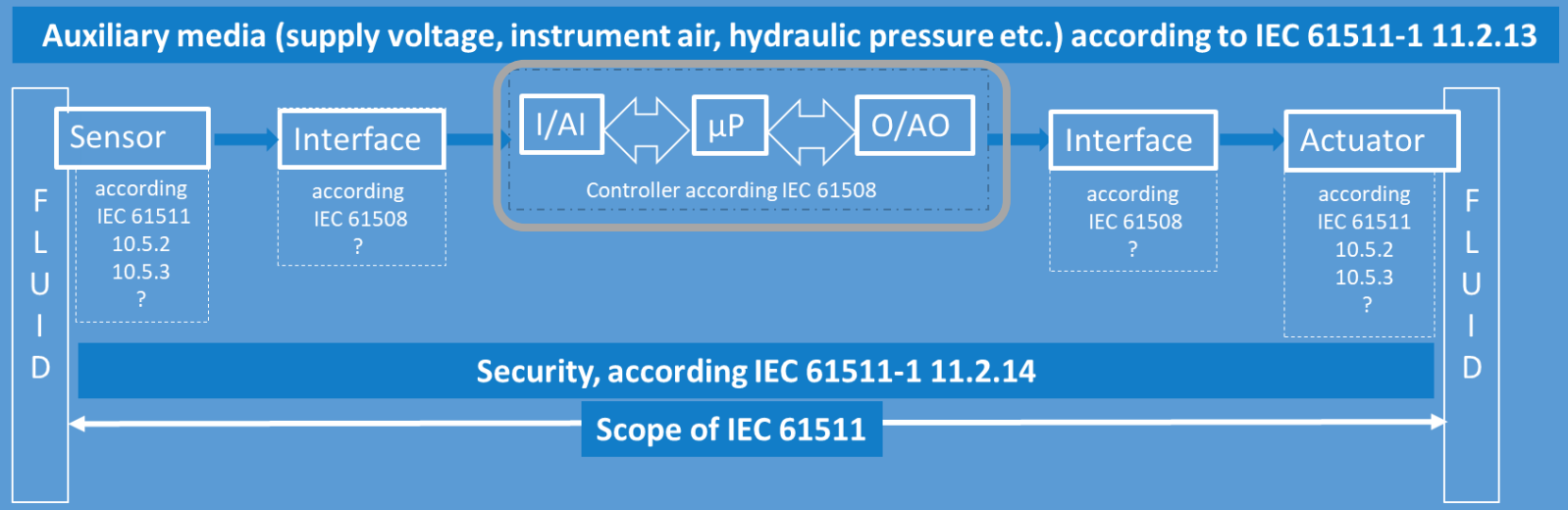




Validation Execution

Coverage of Application Software Validation

- Programmed functions
- Complex functionalities
- Interactions between different software modules



Validation Plan is

- designed in a fashion, diverse to programming
- allowing to test complete sequences
- executed in a SIL 3 compliant environment

What's about the “small” rest?

SIS Application Validation

SRS

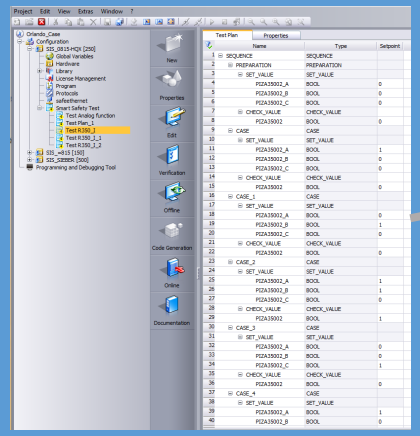
Define Modules

Test Modules

Instance tested Modules

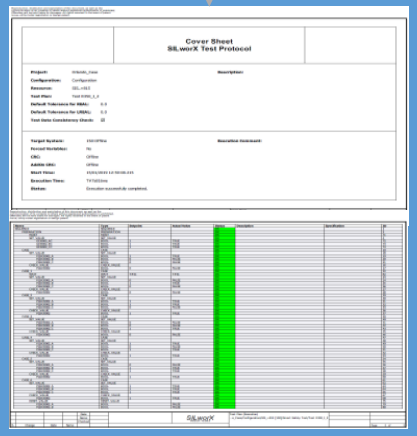
Test Application

Test SIS Function

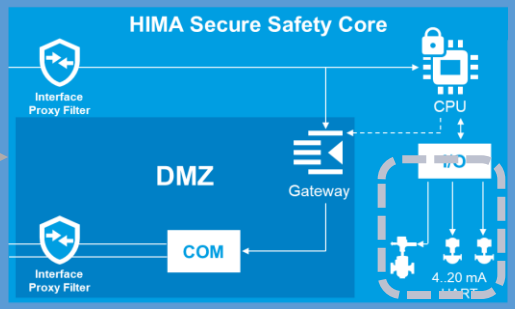


Validation Plan

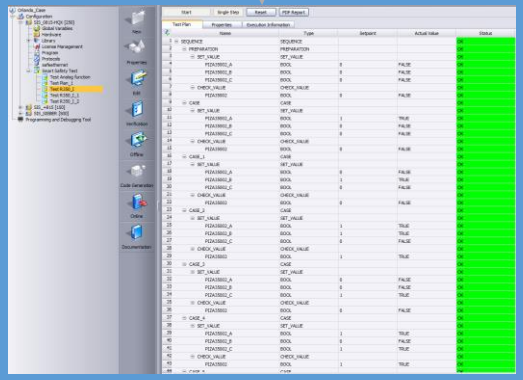
Engineering Tool



Validation Evidence



Validation Execution

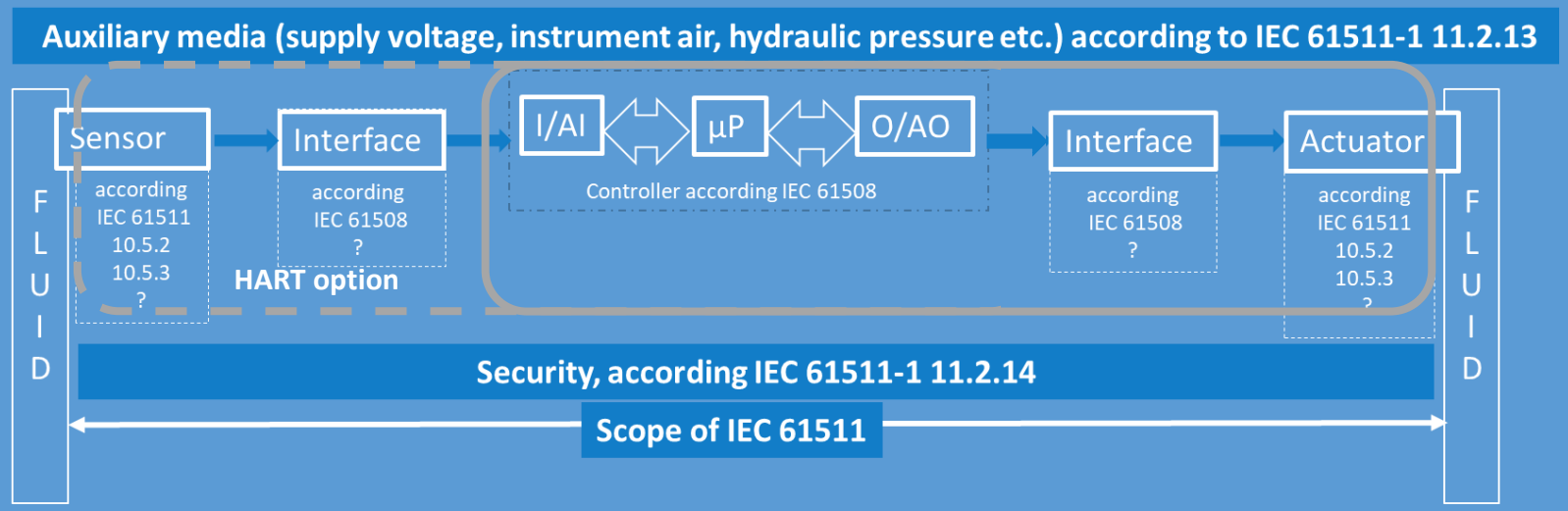


Coverage of Application Validation

- Programmed functions
- Interactions between different sub-systems
- Interaction with BPCS

SIS application validation is

- Executed in SIL 3
- Allowing to test entire SIS Equipment (depending on the instruments used)
- Automatically documenting (correct) test execution
- Supporting both, commissioning as well as recurrent testing



In SIL 3 Quality

Take-home Points

HIMA offers an automated validation tool, which …

- is SIL3-compliant
- meets the engineering requirements that arise from the IEC61511
- can execute and document a 100% application software validation improving the application quality
- can automatically test complete applications including sub-systems during commissioning

  and recurrent testing driving compliance while reducing cost
- can automatically document results of tests executed driving IEC 61511 compliance
- can drastically reduce cost of compliance with the required validation processes

  based on practical experiences those savings can reach 60 to 80%

Thank you!

Peter Sieber

VP Norms & Standards. VP Region China