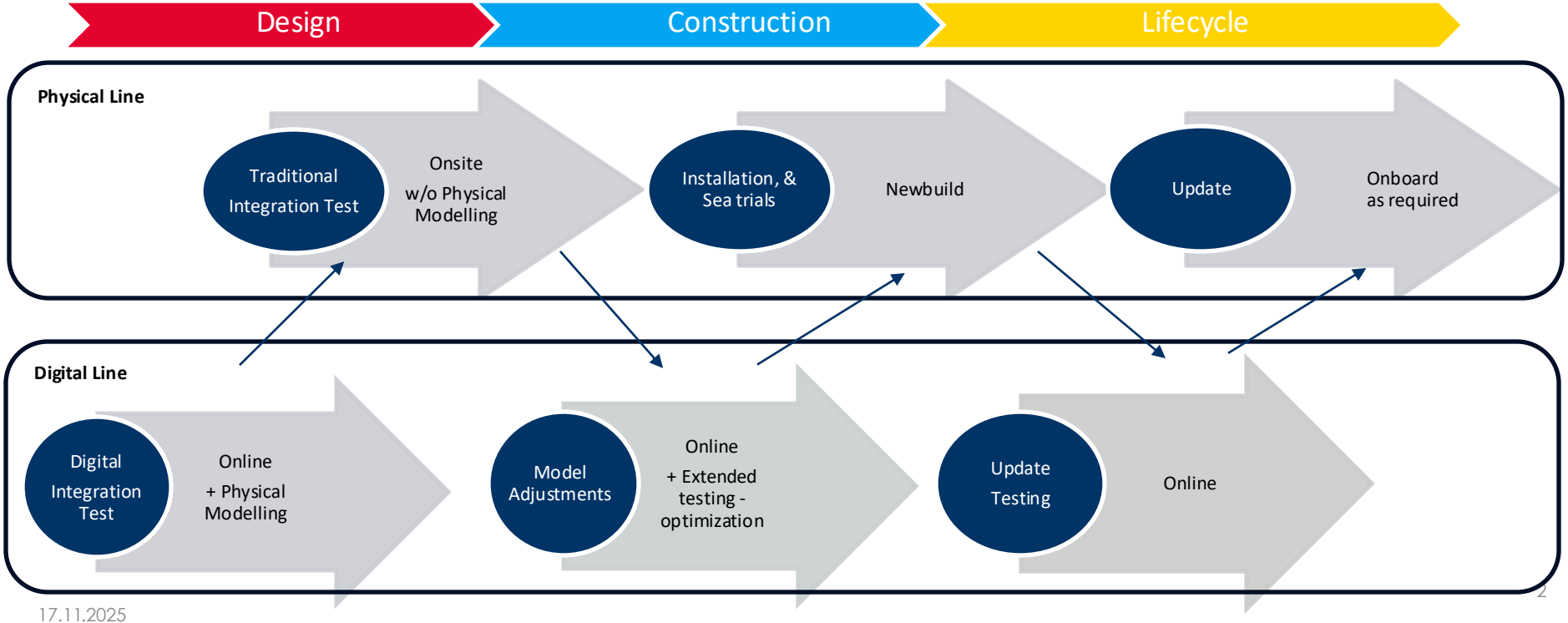
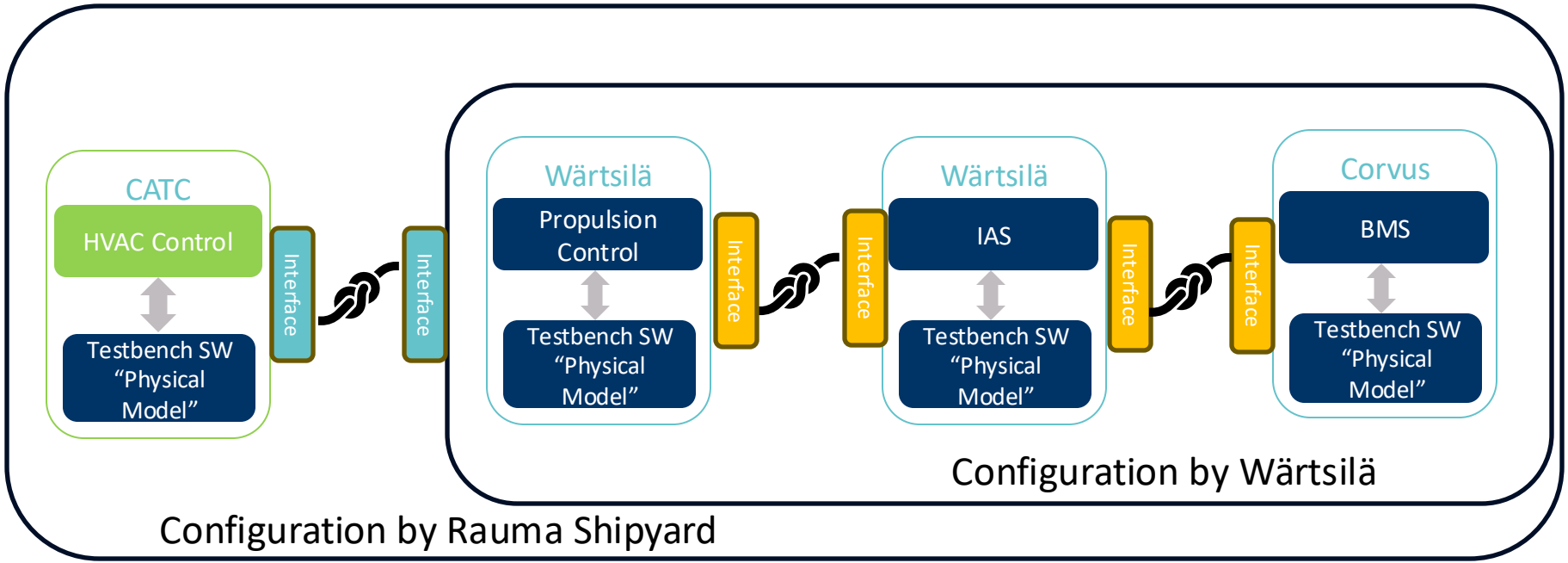
Transforming PLC Control Logic into FMUs Using OpenPLC: An Open Source Approach

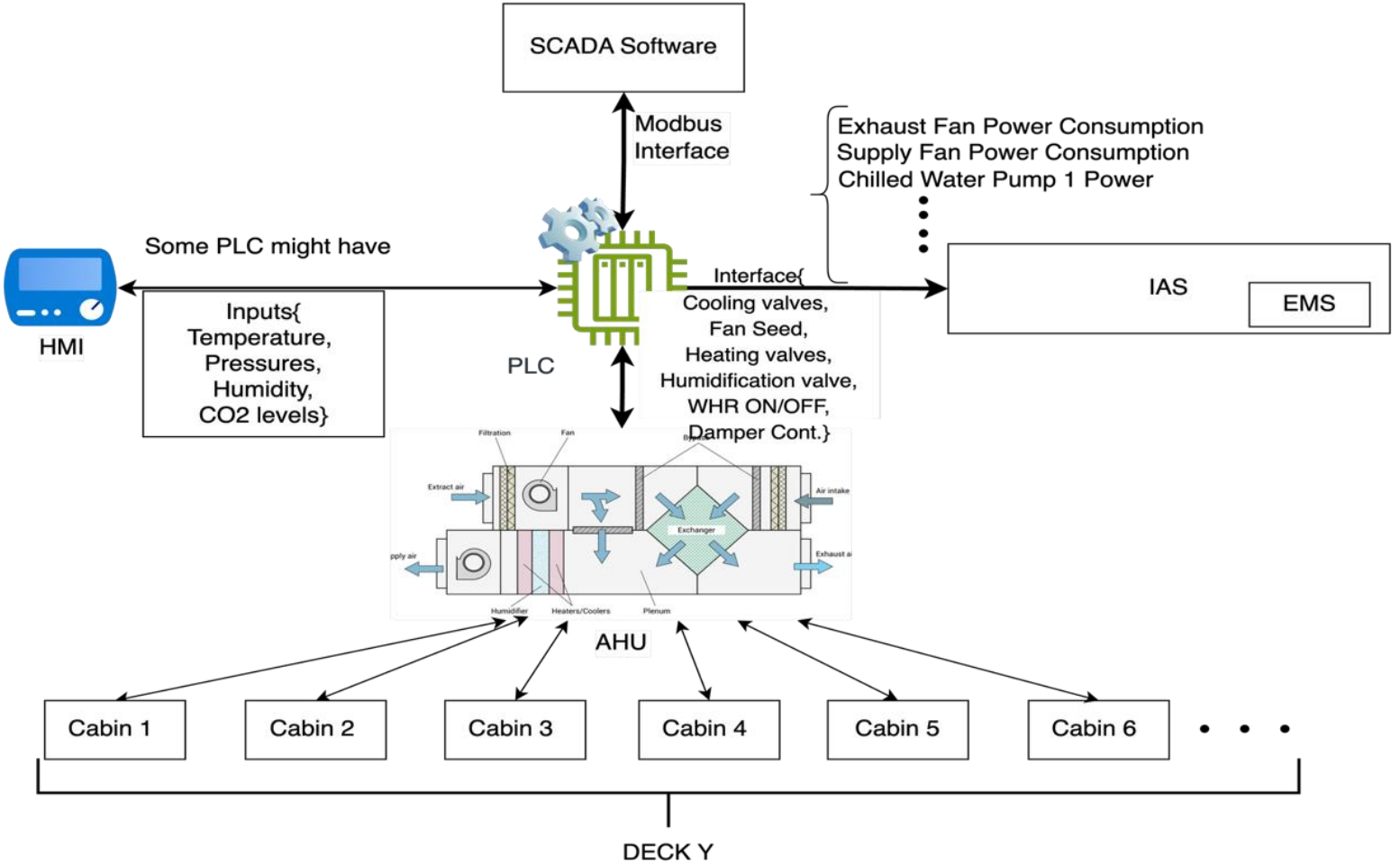
Hergys Rexha
Åbo Akademi University,
Turku, Finland

Traditional steps in shipbuilding and updates



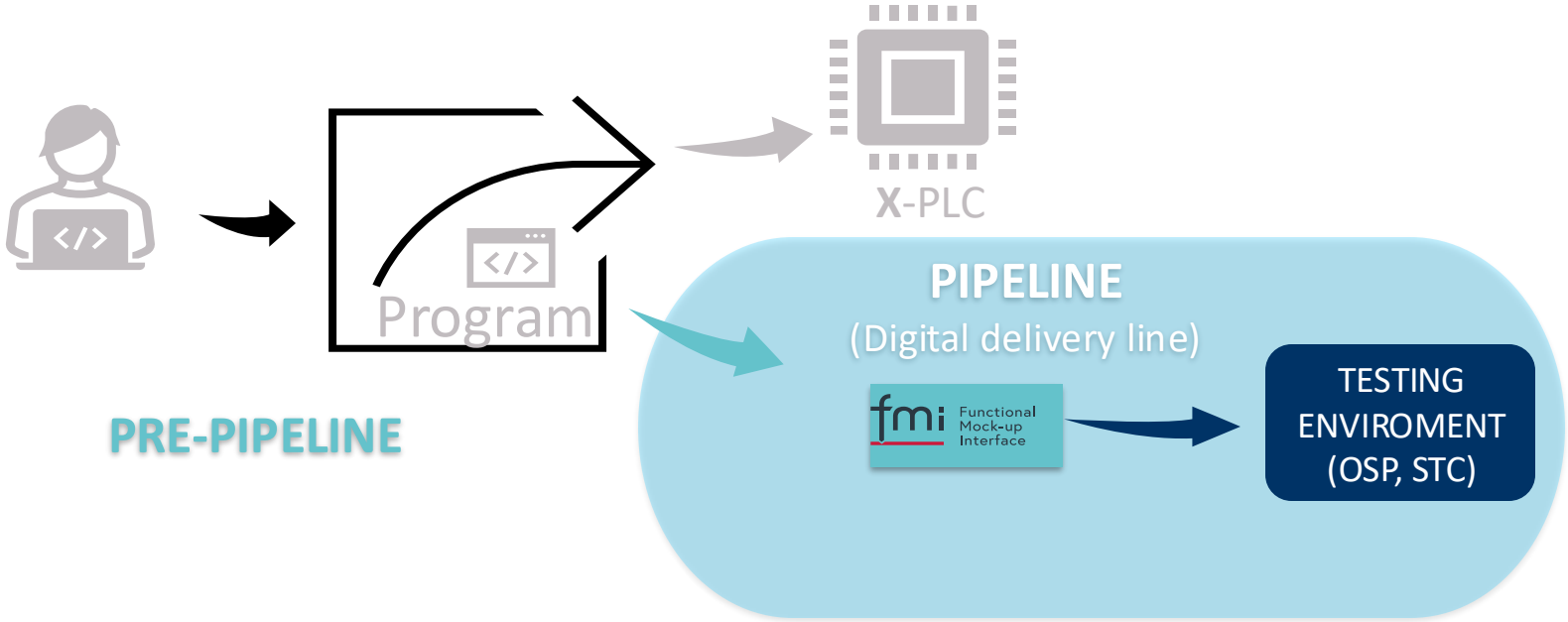
Real use-case





Åbo Akademi University

Digital delivery in automation



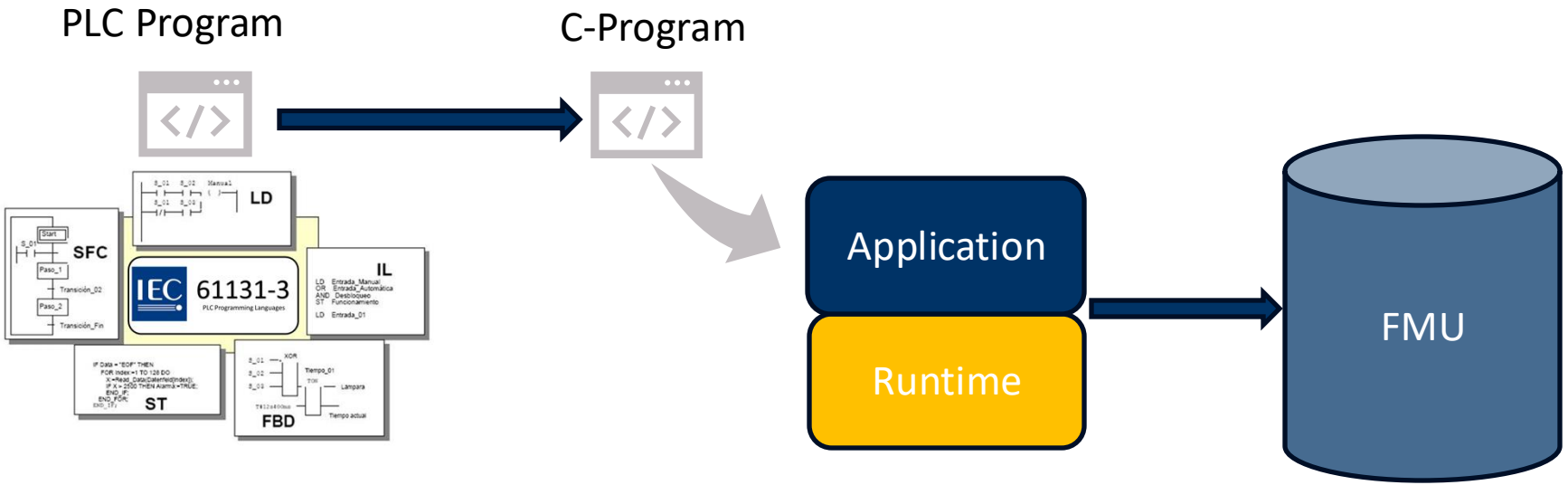
PLC vendor FMI support

PLC Vendor	Software Tool(s)	FMU Export Support	Details
Siemens	Simcenter Amesim	Yes (with license)	FMU export supported via optional FMI export license in Amesim.
Rockwell Automation	Studio 5000, FactoryTalk	No	No native FMU export functionality found.
Schneider Electric	EcoStruxure Machine Expert	No	No FMU export support documented.
Mitsubishi Electric	GX Works2/3, iQ Works	No	No FMU export capability mentioned.
Beckhoff	TwinCAT 3	Initial Release	FMU export supported via TwinCAT 3 Simulation Runtime (TE1421).
Wago	Codesys V3	No	FMU export enabled via extended compilation pipeline in recent research.

[1]

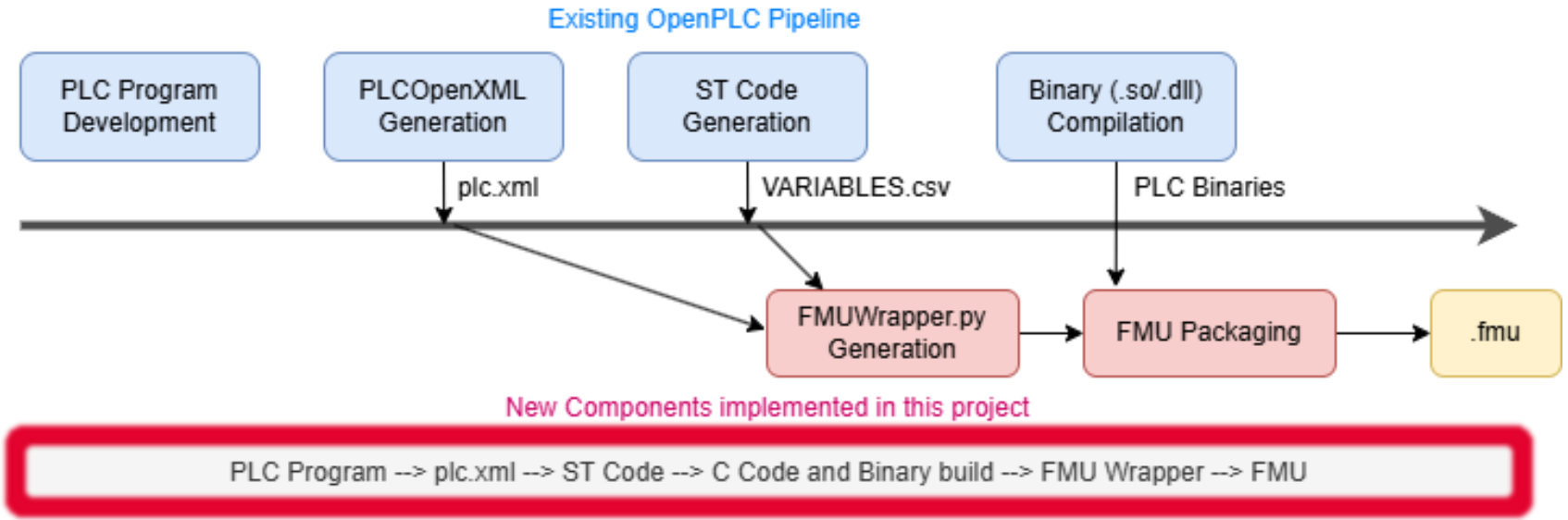
Automatic translation process pipeline

Åbo Akademi University

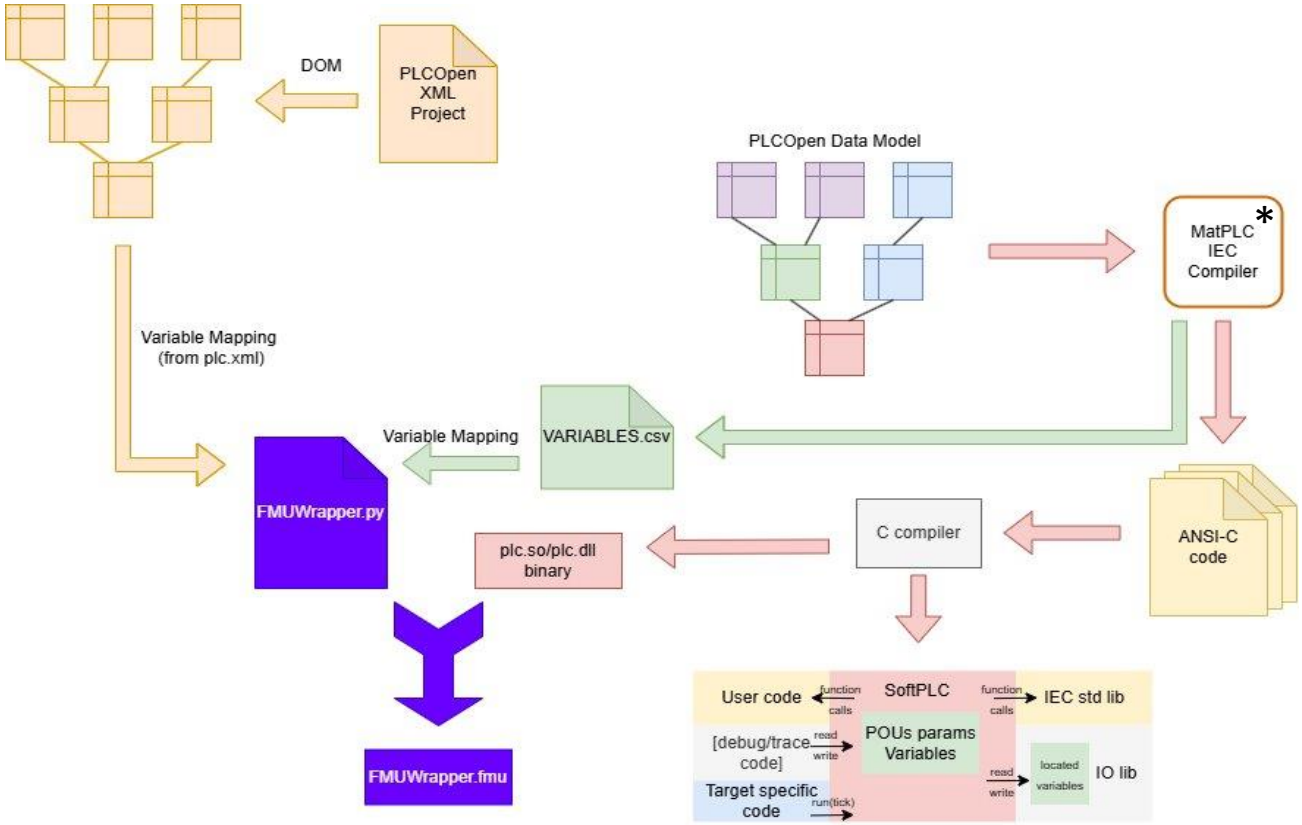


Pipeline Overview in OpenPLC

Åbo Akademi University

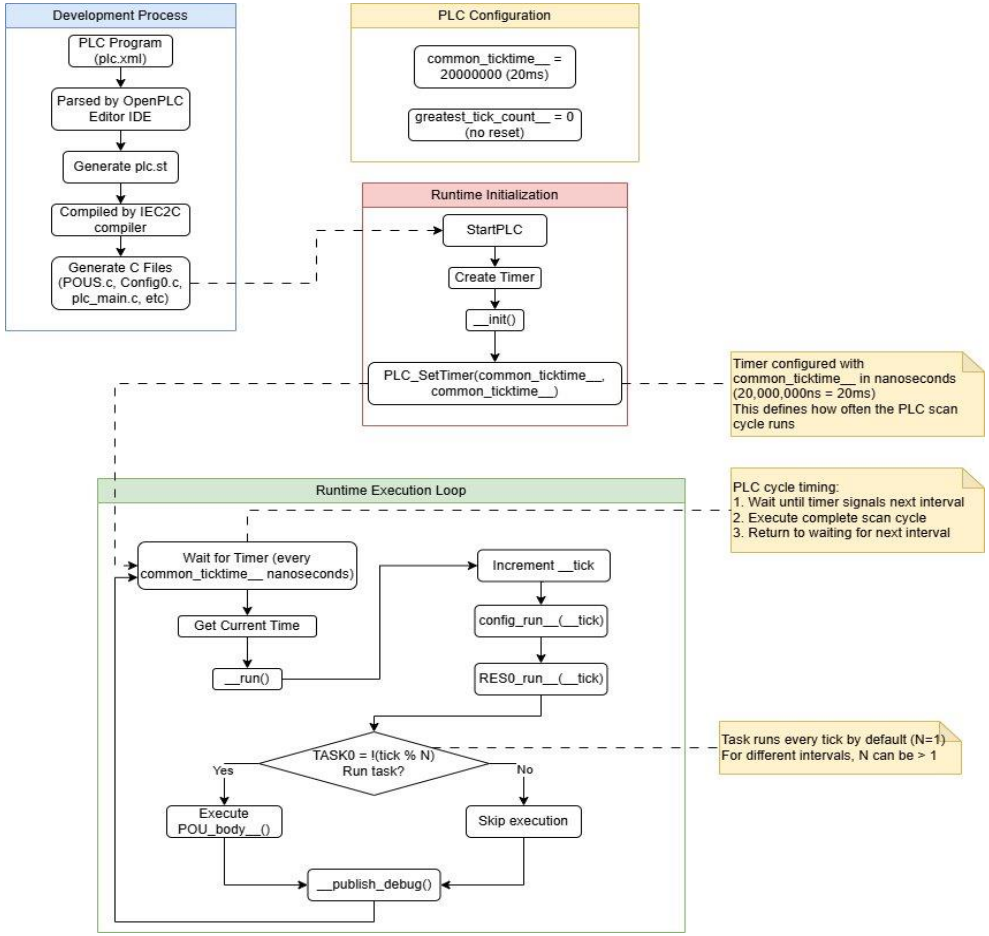


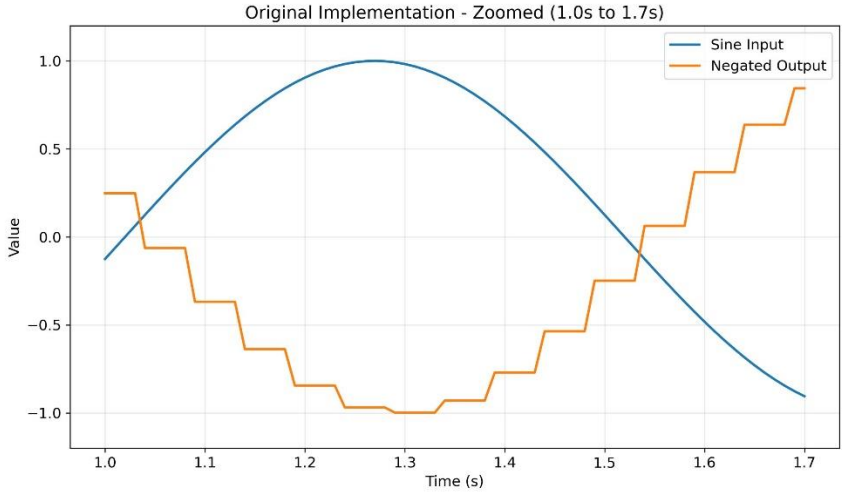
Technical Workflow



*https://www.doria.fi/handle/10024/192965

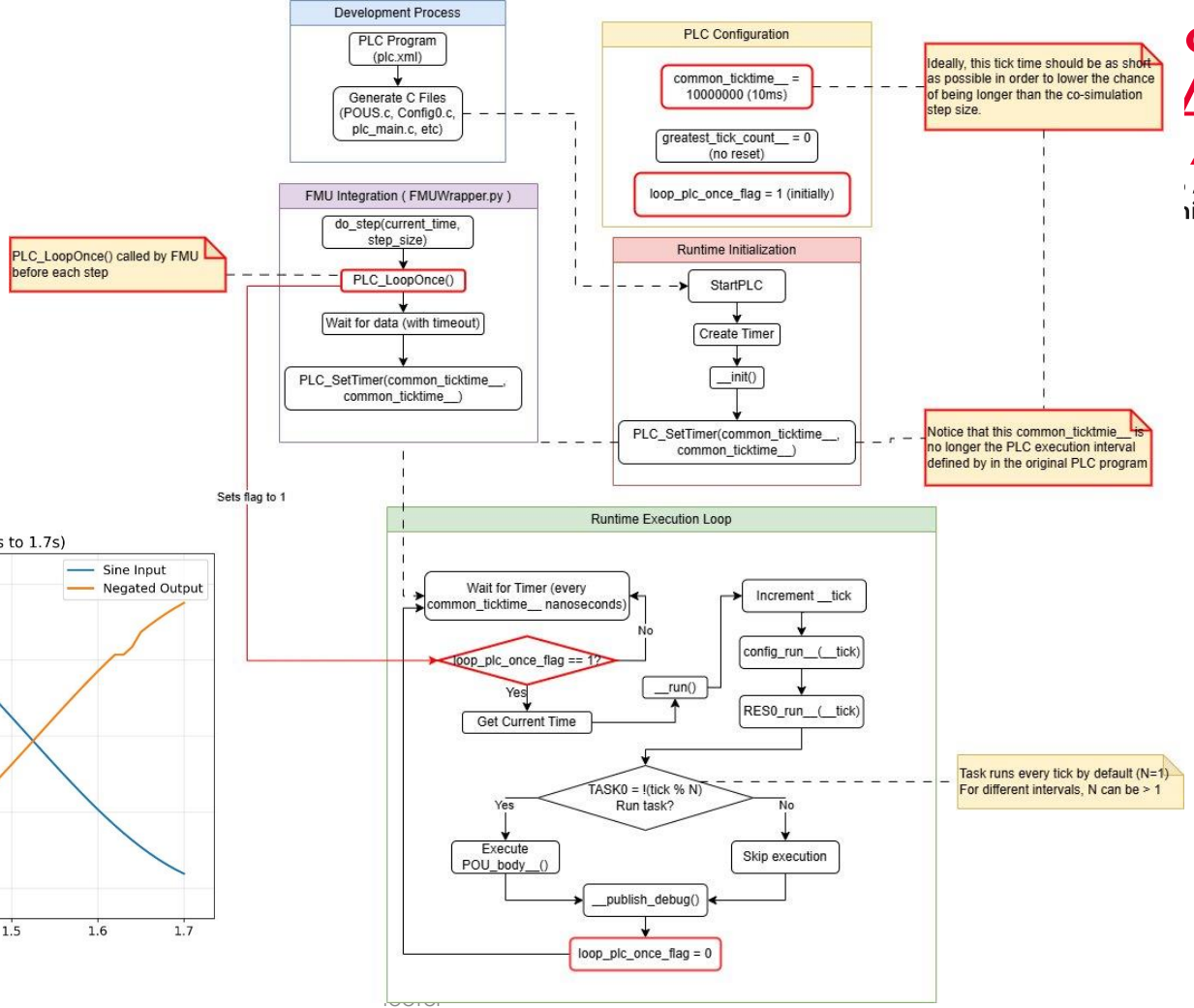
Co-simulation period vs. PLC runtime

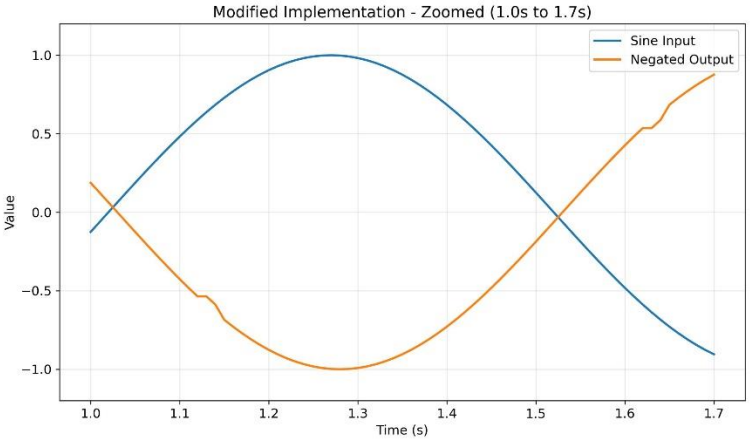




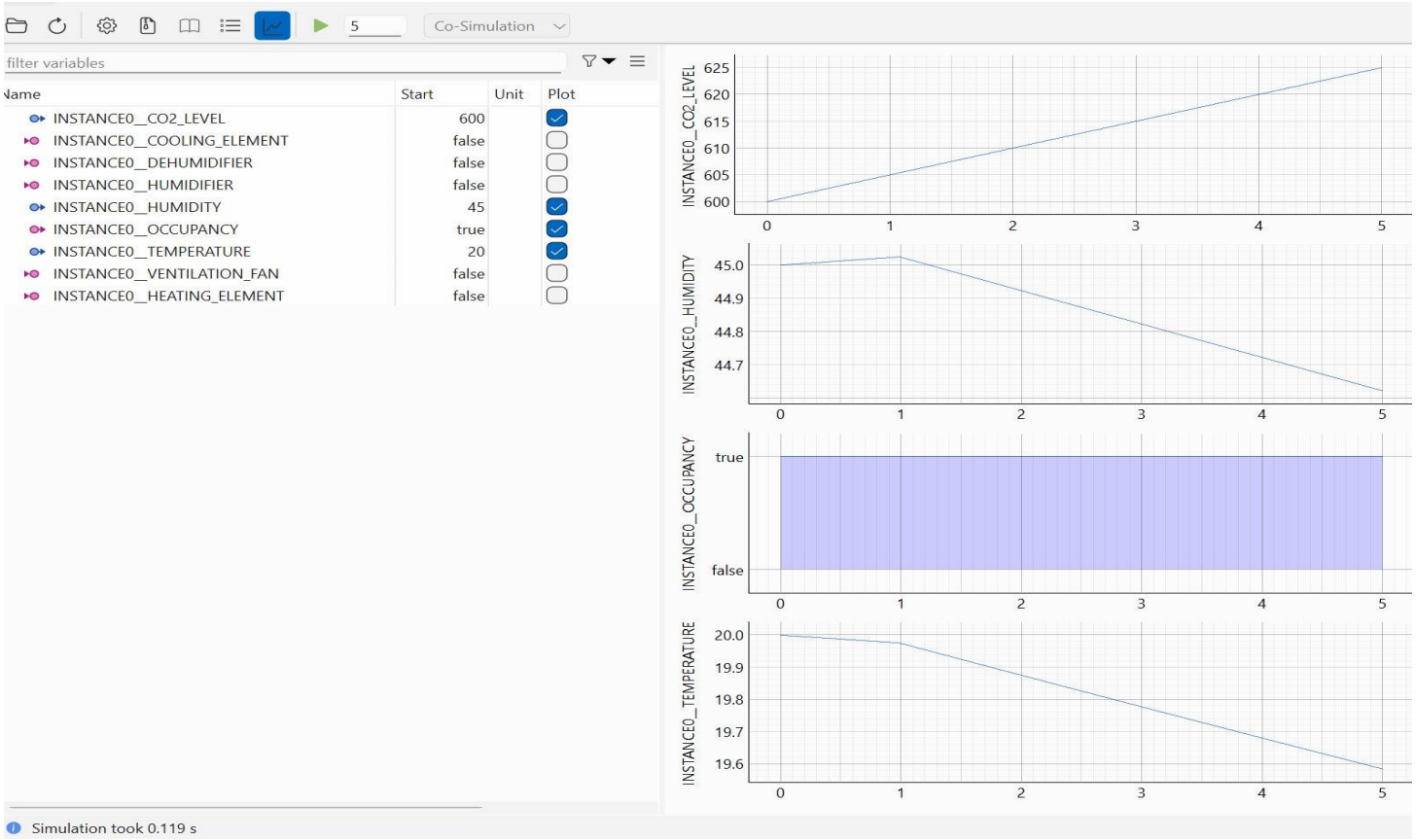
Åbo Akademi University

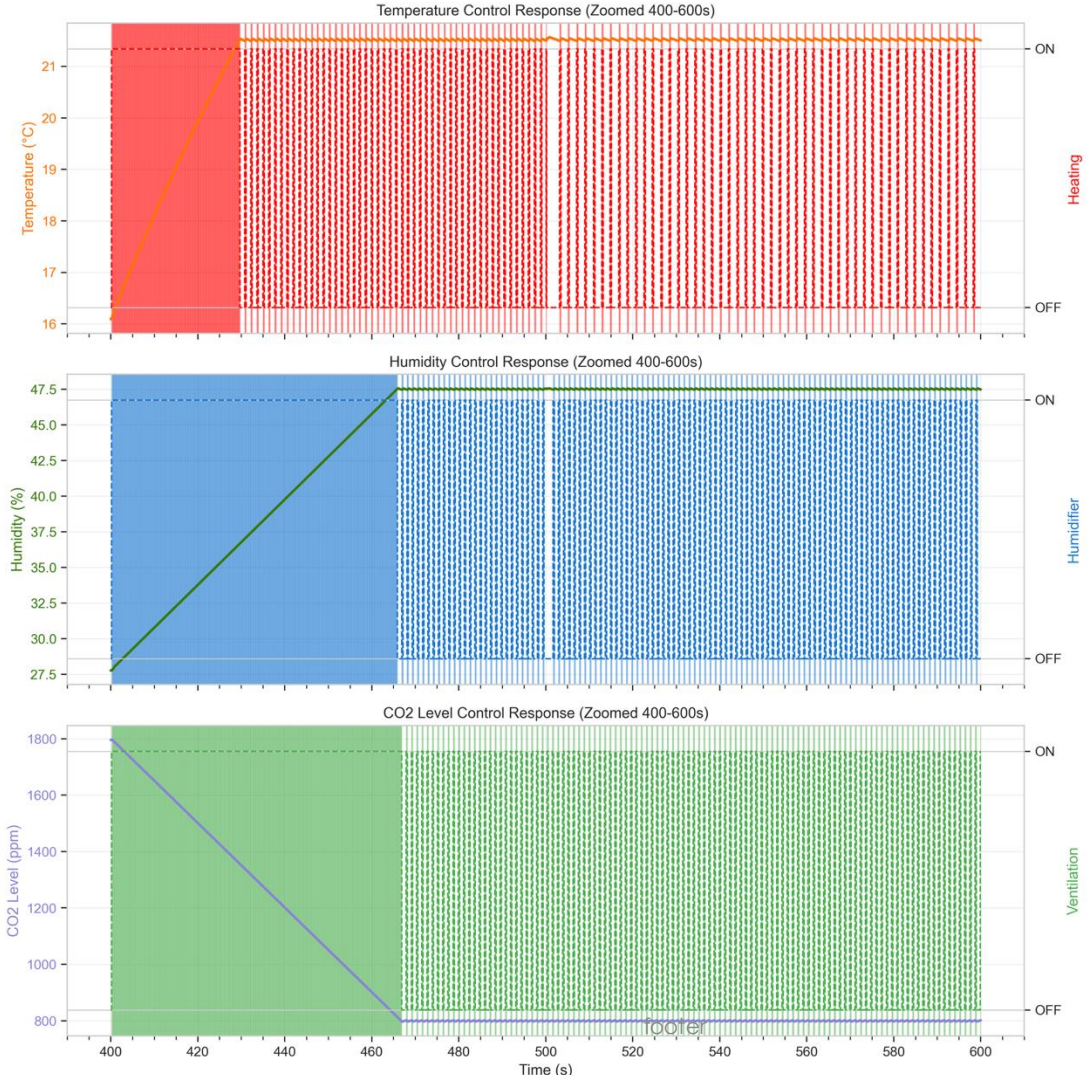
Solution


Development Process
PLC Program (plc.xml)
Generate C Files (POUS.c, Config0.c, plc_main.c, etc)
PLC Configuration
common_ticktime__ = 10000000 (10ms)
greatest_tick_count__ = 0 (no reset)
loop_plc_once_flag = 1 (initially)
Ideally, this tick time should be as short as possible in order to lower the chance of being longer than the co-simulation step size.
FMU Integration (FMUWrapper.py)
do_step(current_time, step_size)
PLC_LoopOnce()
Wait for data (with timeout)
PLC_SetTimer(common_ticktime__, common_ticktime__)
PLC_LoopOnce() called by FMU before each step
Runtime Initialization
StartPLC
Create Timer
__init()
PLC_SetTimer(common_ticktime__, common_ticktime__)
Notice that this common_ticktmie__ is no longer the PLC execution interval defined by in the original PLC program
Sets flag to 1
Runtime Execution Loop
Wait for Timer (every common_ticktime__ nanoseconds)
loop_plc_once_flag == 1?
No
Yes
Get Current Time
__run()
Increment __tick
config_run__(__tick)
RES0_run__(__tick)
TASK0 = !(tick % N) Run task?
Yes
No
Execute POU_body__()
Skip execution
__publish_debug()
loop_plc_once_flag = 0
Task runs every tick by default (N=1) For different intervals, N can be > 1


Modified Implementation - Zoomed (1.0s to 1.7s)
Sine Input
Negated Output
Value
Time (s)
1.0
0.5
0.0
−0.5
−1.0
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7

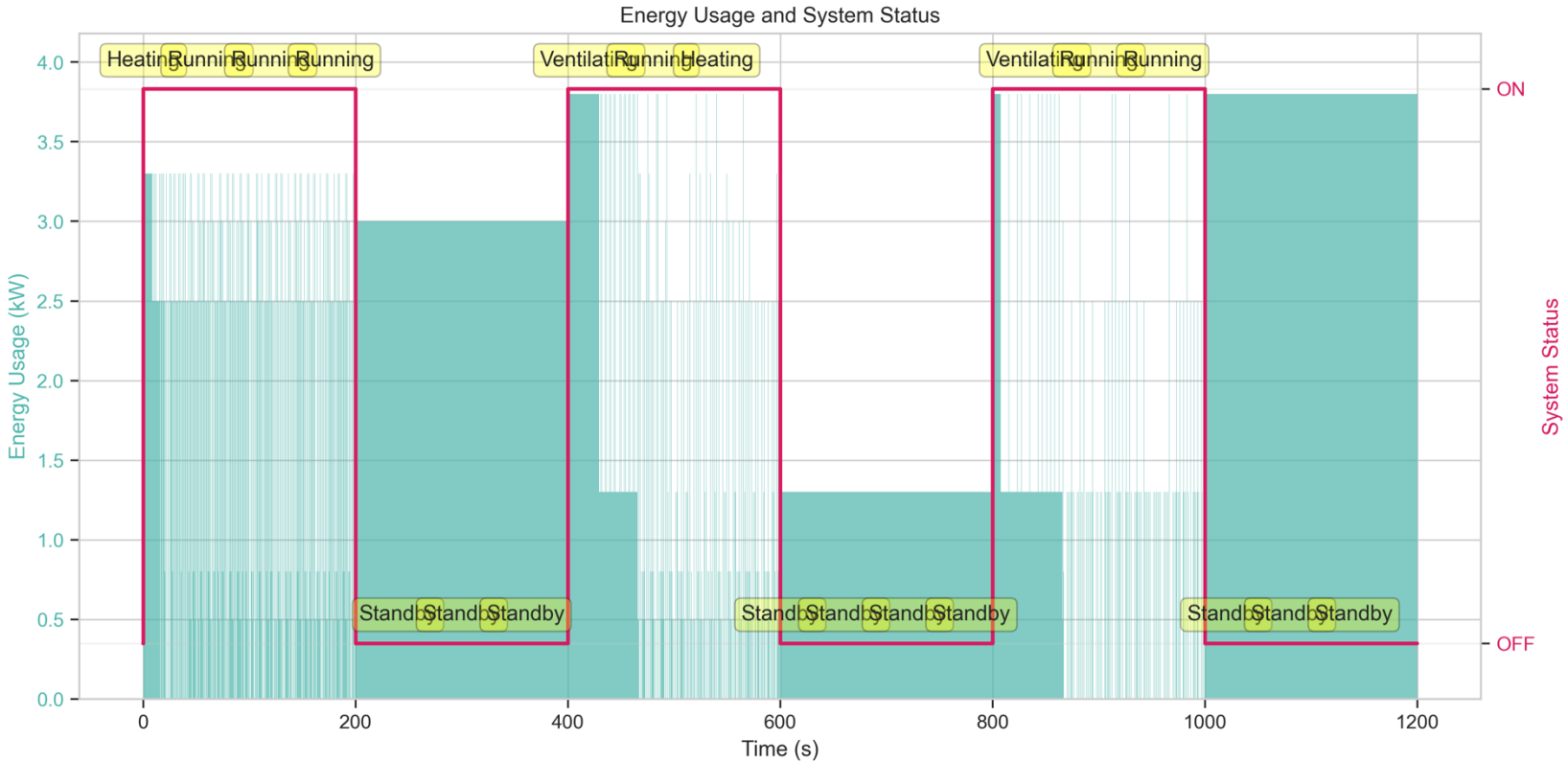
Simulations in OSP





Åbo Akademi University

Energy Results



Limitations of the approach

Åbo Akademi University

- Performance Overhead
- Synchronization issues
- Determinism limitations
- Integration with HIL

Summary

Åbo Akademi University

- Explored different mechanisms to export the control logic in automation systems in a simulation environment.
- Designed an automated FMU generation pipeline integrated into the OpenPLC Editor.
- Developed a dynamic loading system allowing PLC binaries to run inside a standalone FMU container for co-simulation or model exchange.
- Design and validation of a data exchange protocol for effective communication between the FMI interface and the PLC logic.
- Practical demonstration by means of a thorough HVAC control system case study, validating the efficacy of the approach in actual situations.

Thank you!

Questions?