Towards Plug and Play: Cyber-Physical Components and Automatic Verification

Валерий Вяткин

17 августа 2020 г.

Valeriy.Vyatkin@gmail.com

LULEÅ UNIVERSITY OF TECHNOLOGY

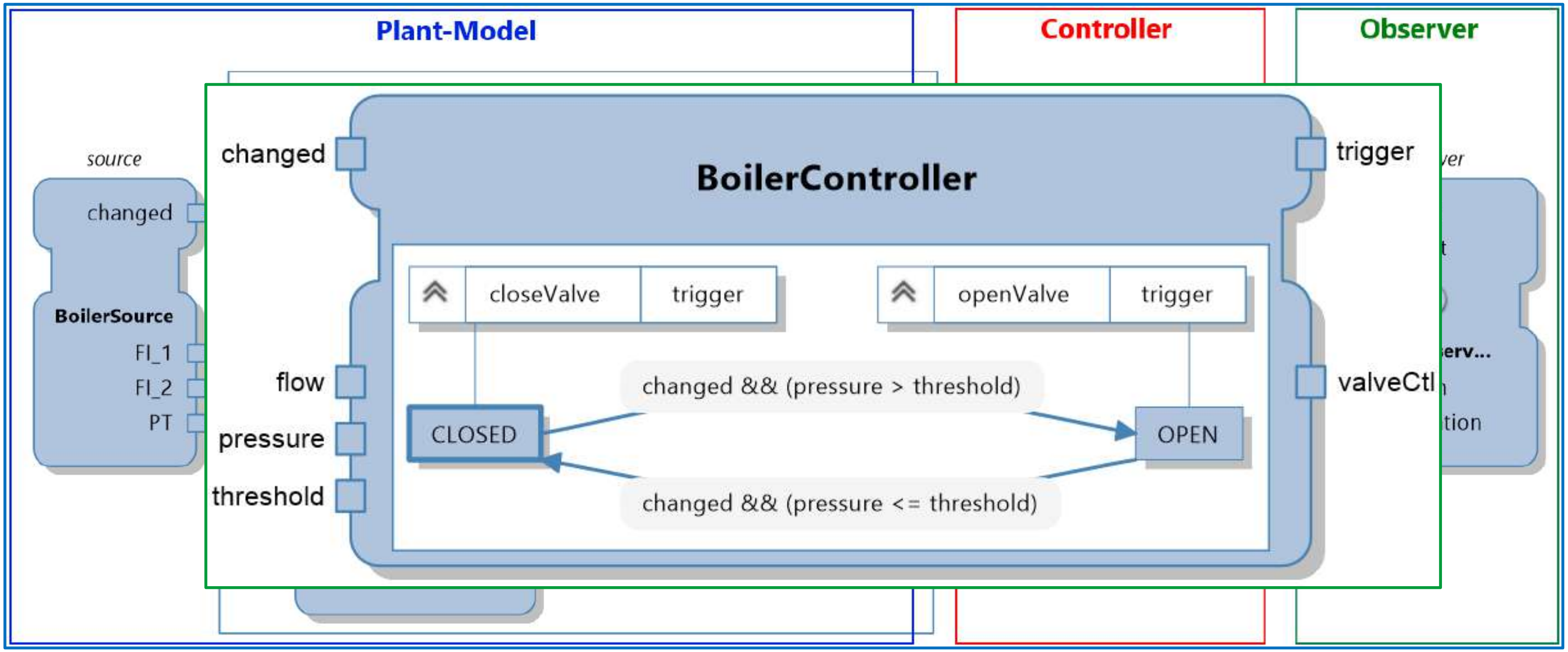
Aalto University
School of Electrical Engineering

ITMO UNIVERSITY

Plan of the lecture

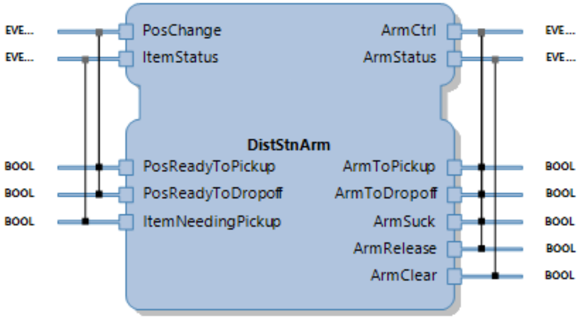
- Component architecture for CPS
  - Examples
  - Automatic system generation
- The challenge of testing
- Formal verification
- Closed-loop verification
- Integrated tool chains

Distributed Component Architecture of IEC 61499



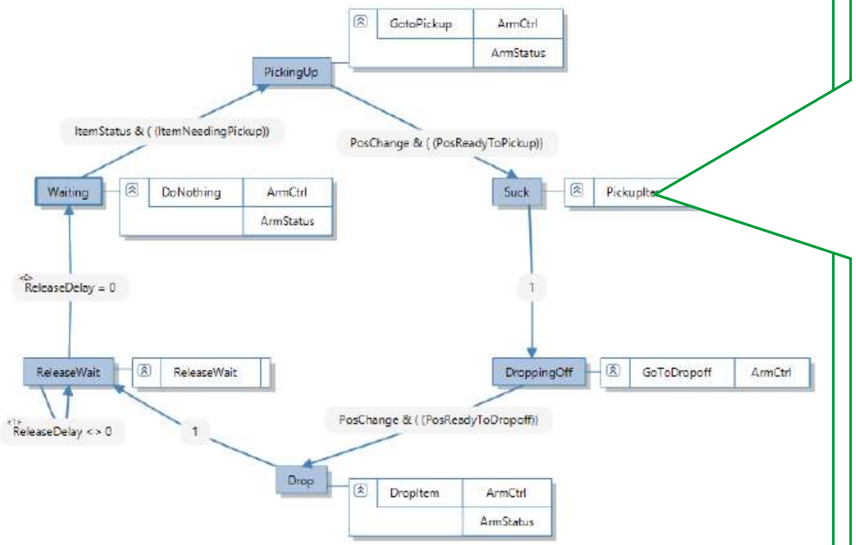
Function block

Function Block Interface



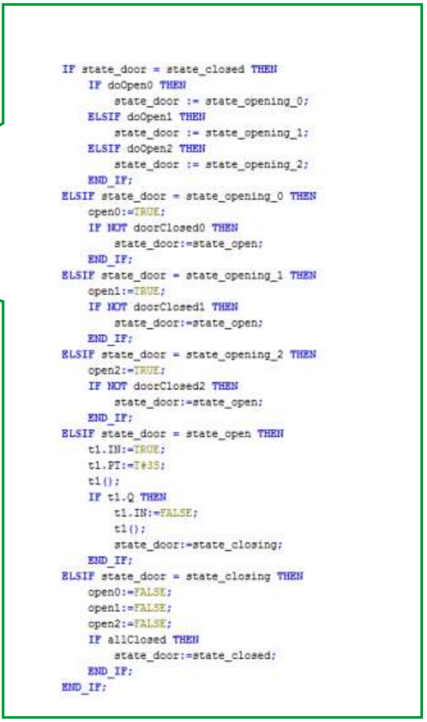
Function Block Interface explicitly declares input/output events and variables of a function block.

State machine implementation



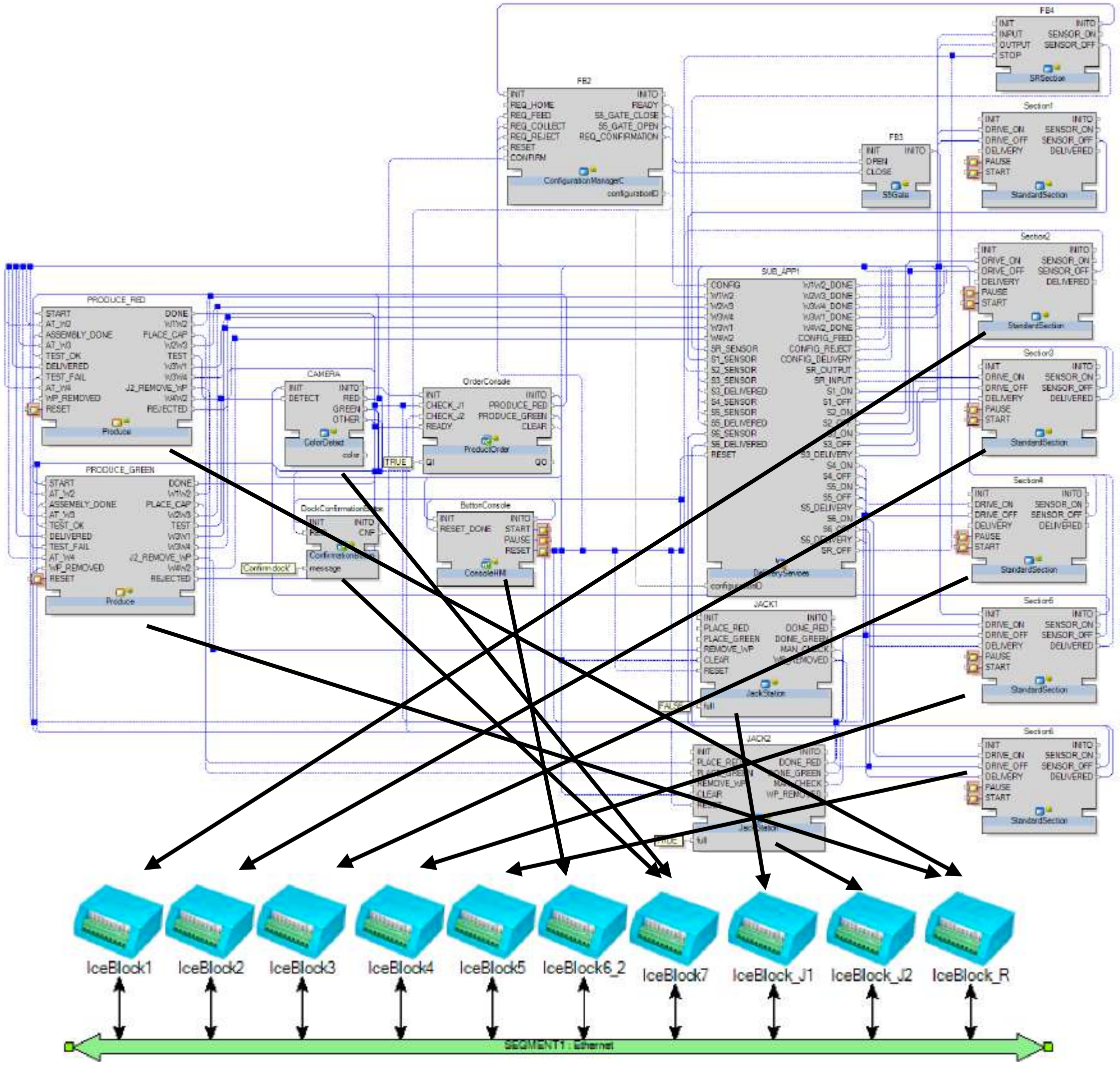
Behavior of a Basic Function Block is implemented by an execution control chart. Textual algorithms can be invoked upon entering a state.

Legacy code

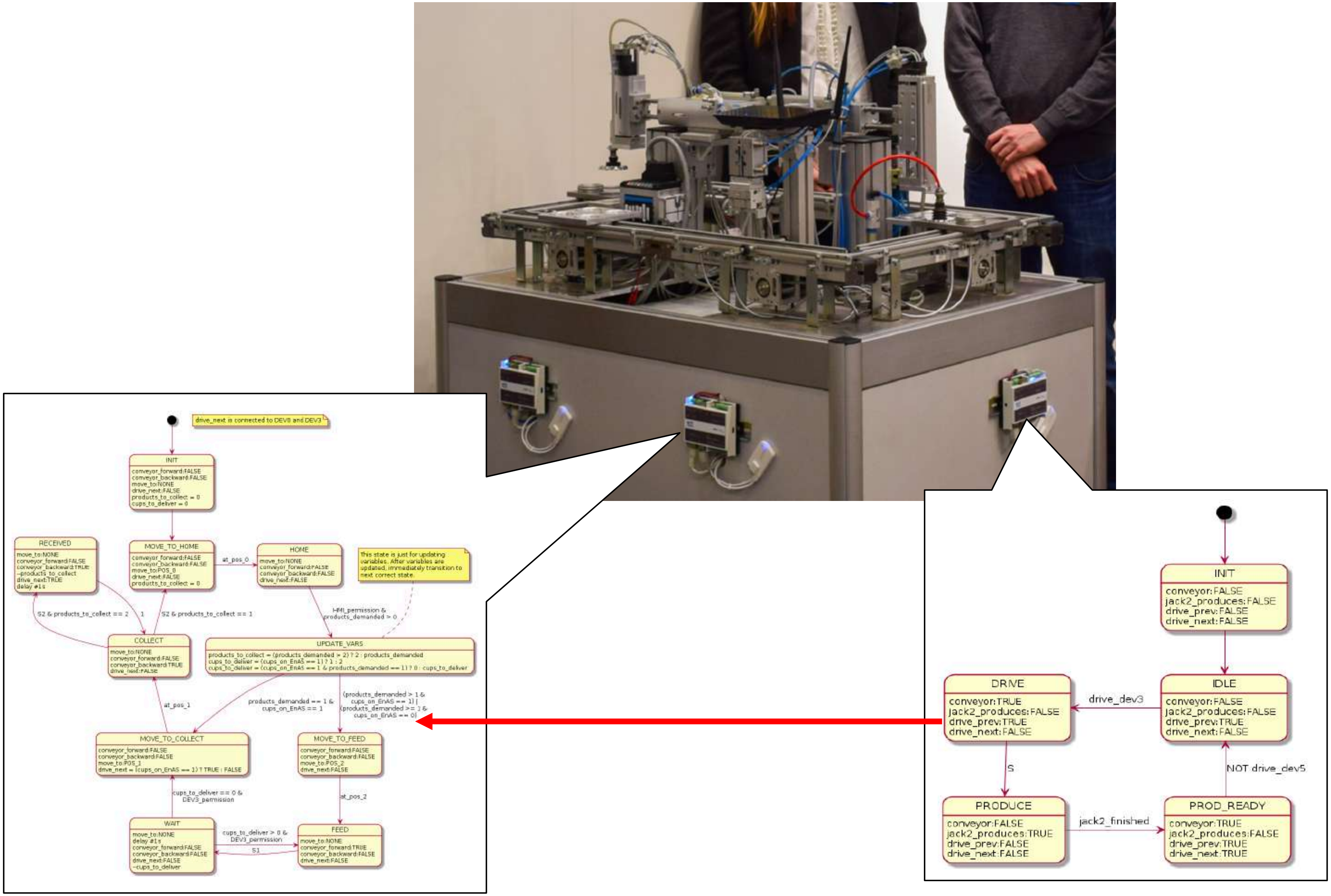


```
IF state_door = state_closed THEN
    IF doOpen0 THEN
        state_door := state_opening_0;
    ELSIF doOpen1 THEN
        state_door := state_opening_1;
    ELSIF doOpen2 THEN
        state_door := state_opening_2;
    END_IF;
ELSIF state_door = state_opening_0 THEN
    open0:=TRUE;
    IF NOT doorClosed0 THEN
        state_door:=state_open;
    END_IF;
ELSIF state_door = state_opening_1 THEN
    open1:=TRUE;
    IF NOT doorClosed1 THEN
        state_door:=state_open;
    END_IF;
ELSIF state_door = state_opening_2 THEN
    open2:=TRUE;
    IF NOT doorClosed2 THEN
        state_door:=state_open;
    END_IF;
ELSIF state_door = state_open THEN
    t1.IN:=TRUE;
    t1.PT:=T#3S;
    t1();
    IF t1.Q THEN
        t1.IN:=FALSE;
        t1();
        state_door:=state_closing;
    END_IF;
ELSIF state_door = state_closing THEN
    open0:=FALSE;
    open1:=FALSE;
    open2:=FALSE;
    IF allClosed THEN
        state_door:=state_closed;
    END_IF;
END_IF;
```

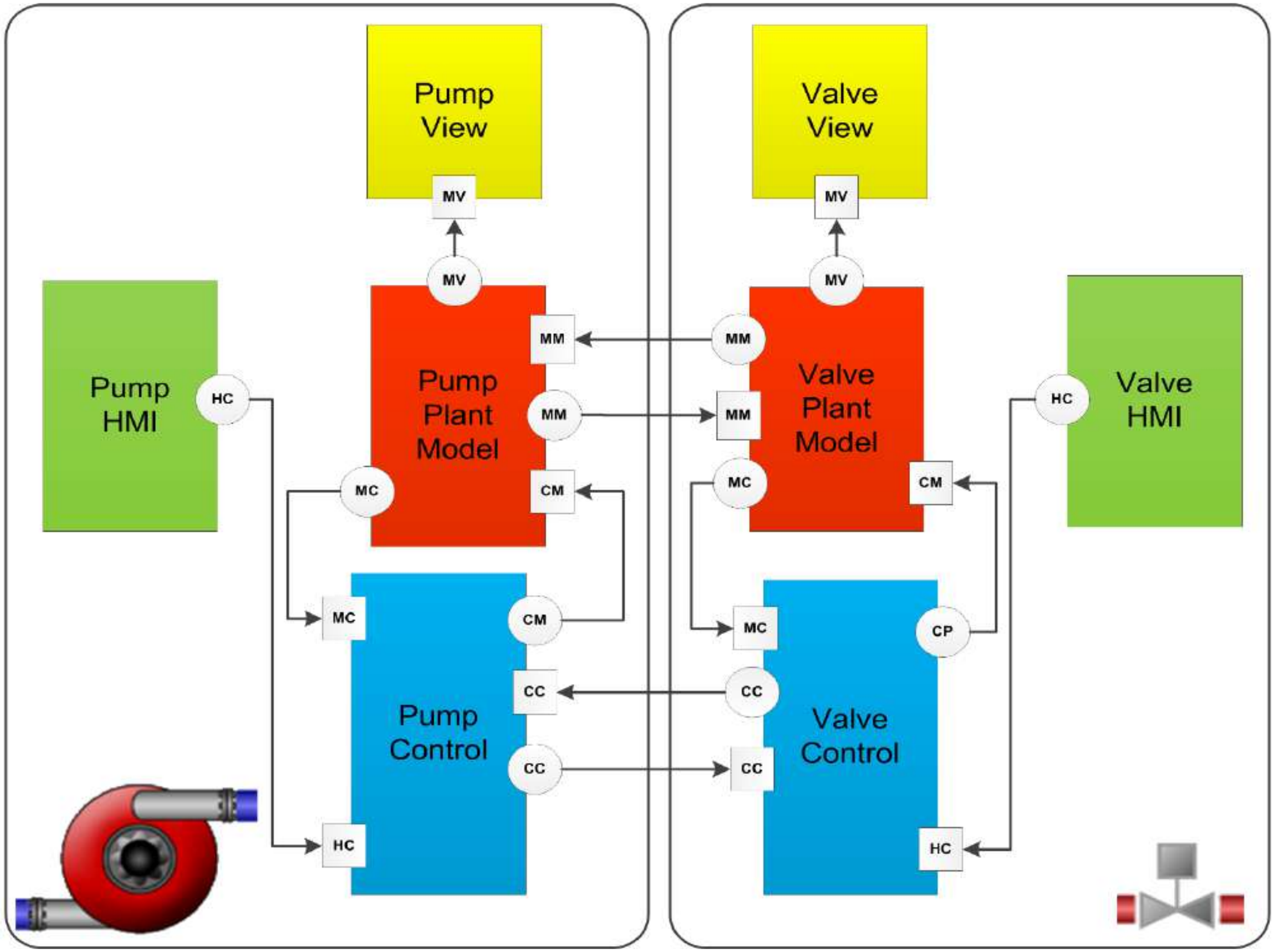
IEC61499: seamless distribution



Communicating state-machines



Composition of CPCA

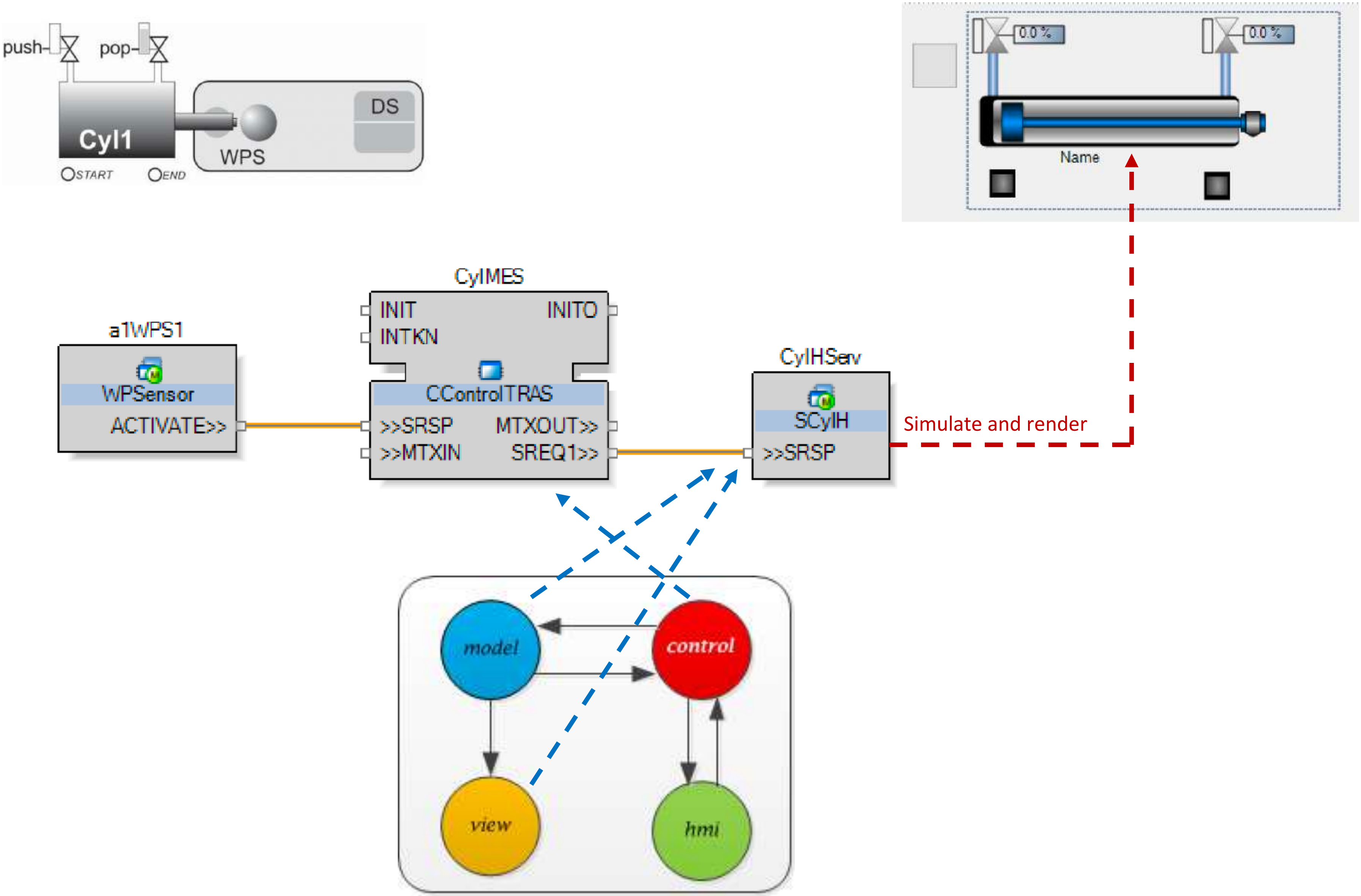


VDMA demonstrator

Integrates components of 25 vendors
High-level control: Communicating state machines connected via message passing
Transport: OPC-UA

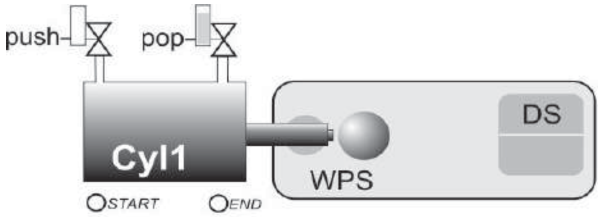
- https://www.youtube.com/watch?v=kT_3IHimNyc
- Same, but live: https://www.youtube.com/watch?v=QQwcIcrONMc

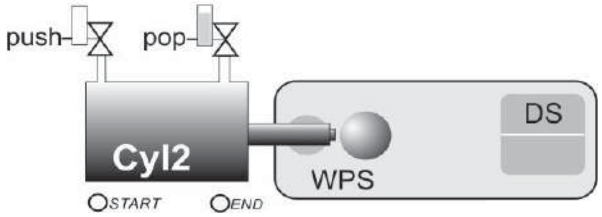
CPSC implemented with FBs

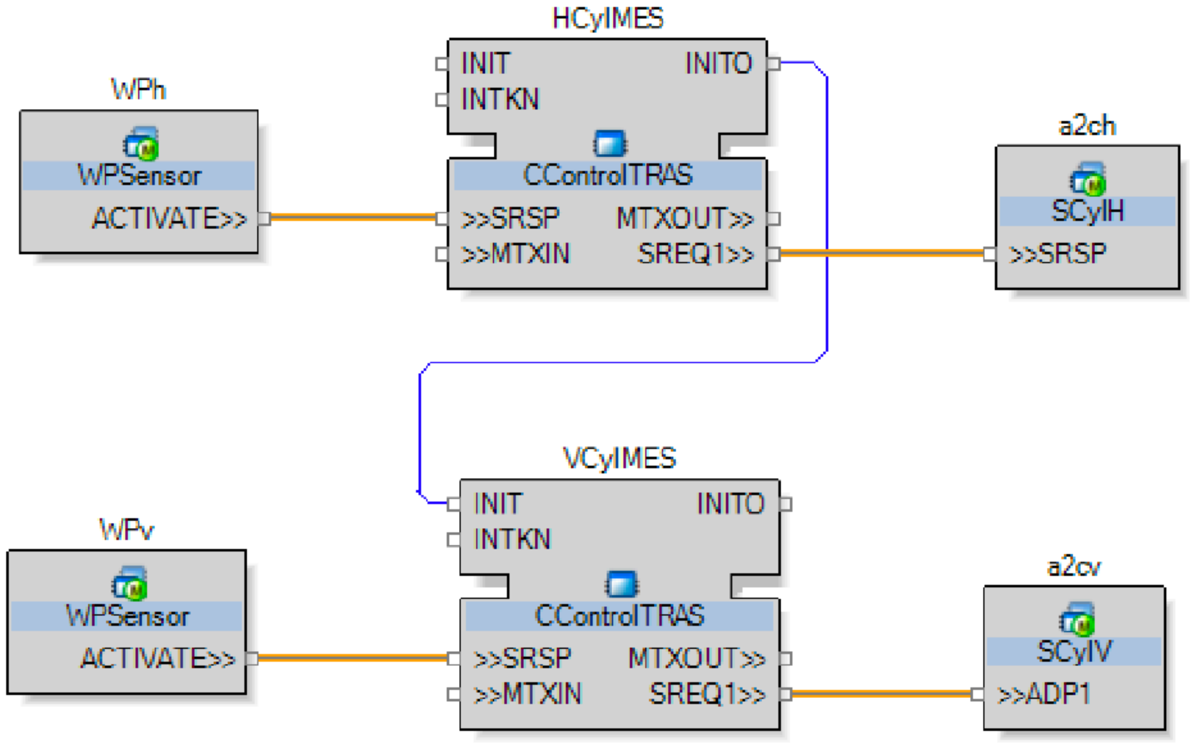


Simulate and render

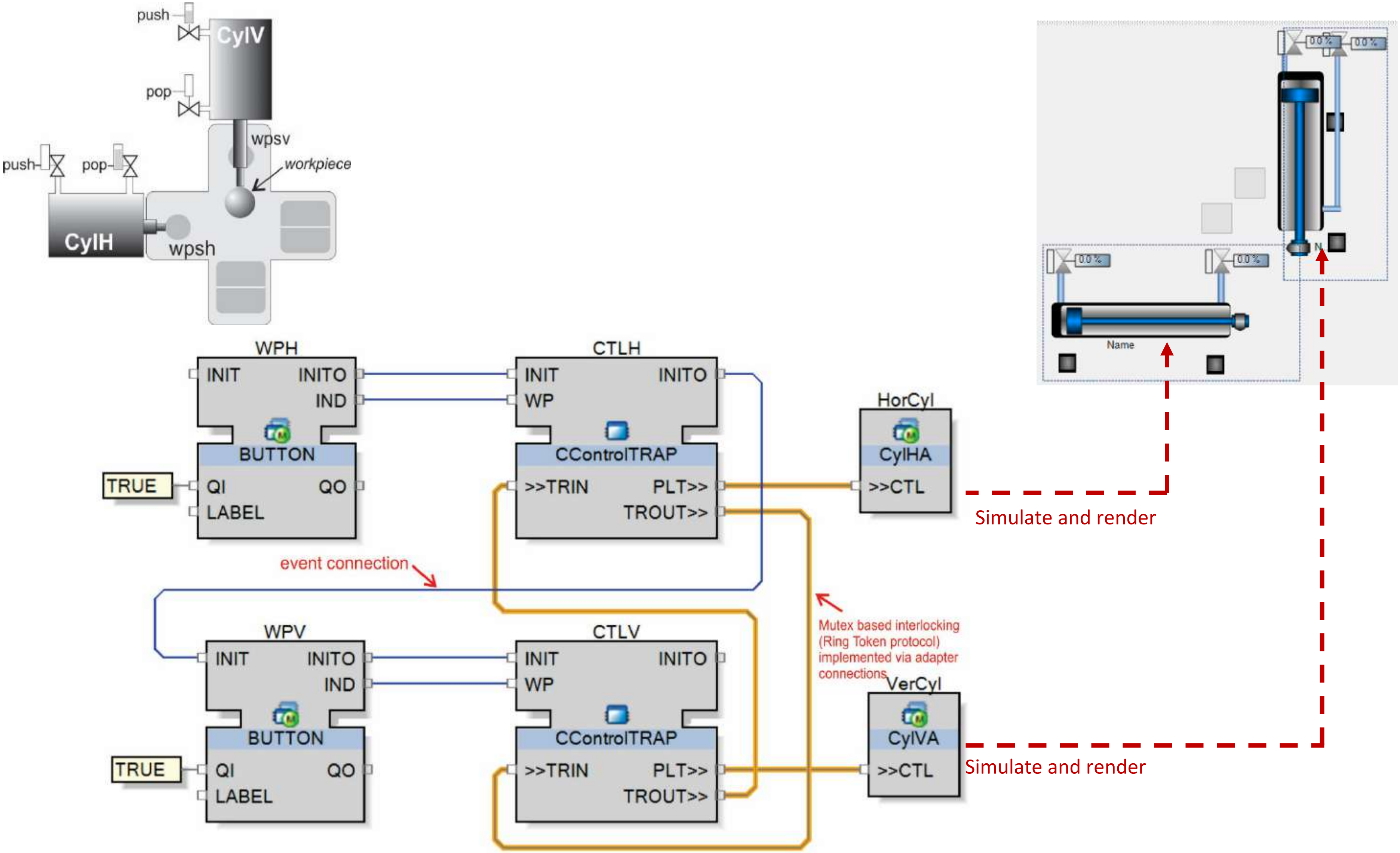
Two independent processes







Intersecting cylinders: Mutual Exclusion



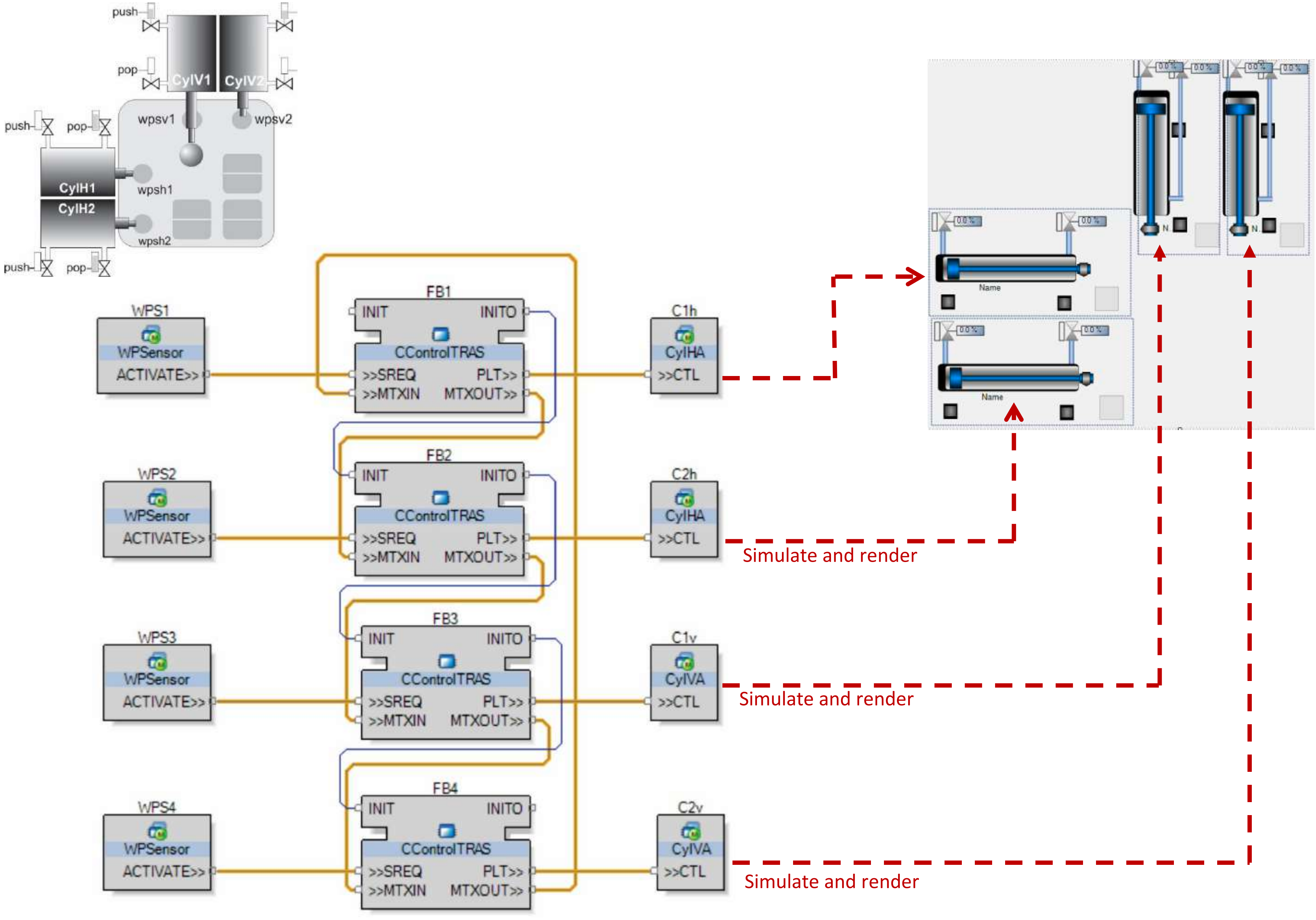
Simulate and render

event connection

Mutex based interlocking (Ring Token protocol) implemented via adapter connections

Simulate and render

Four cylinders

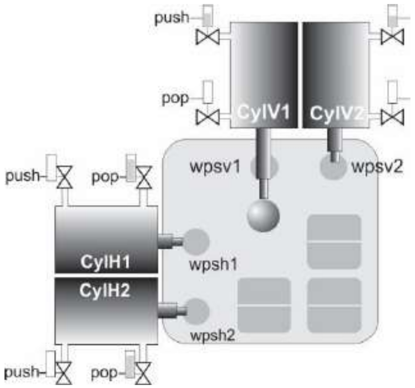


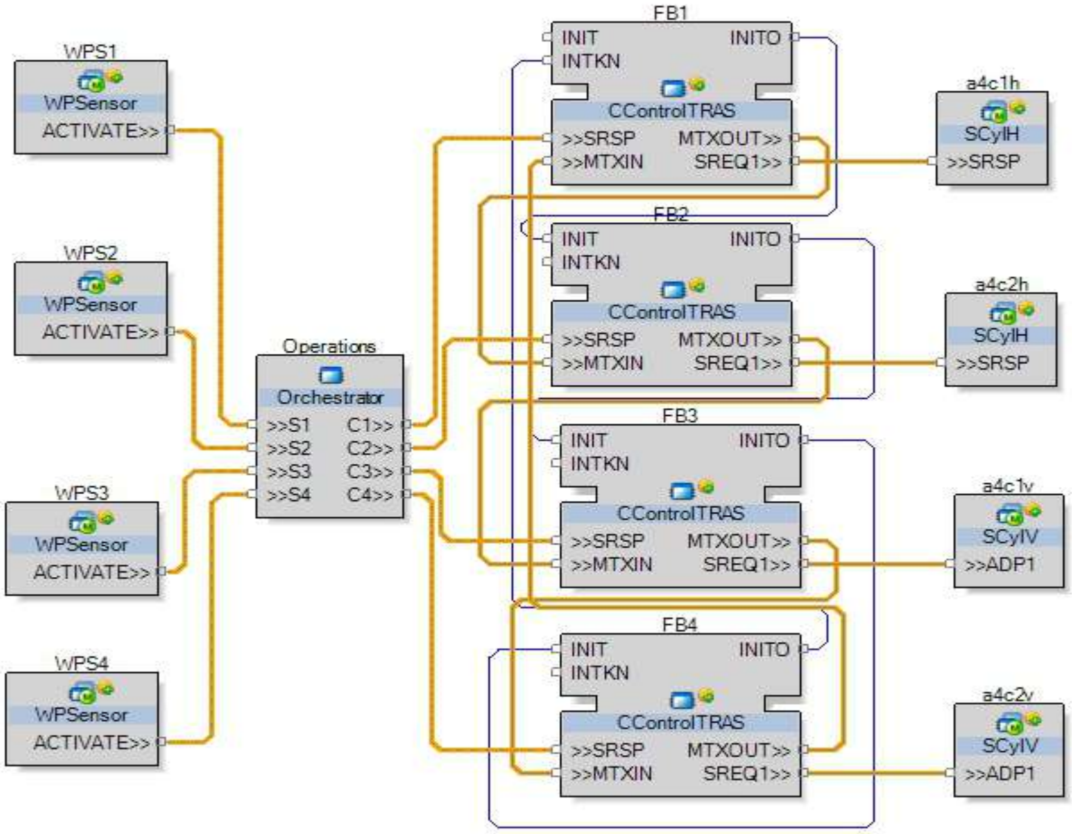
Simulate and render

Simulate and render

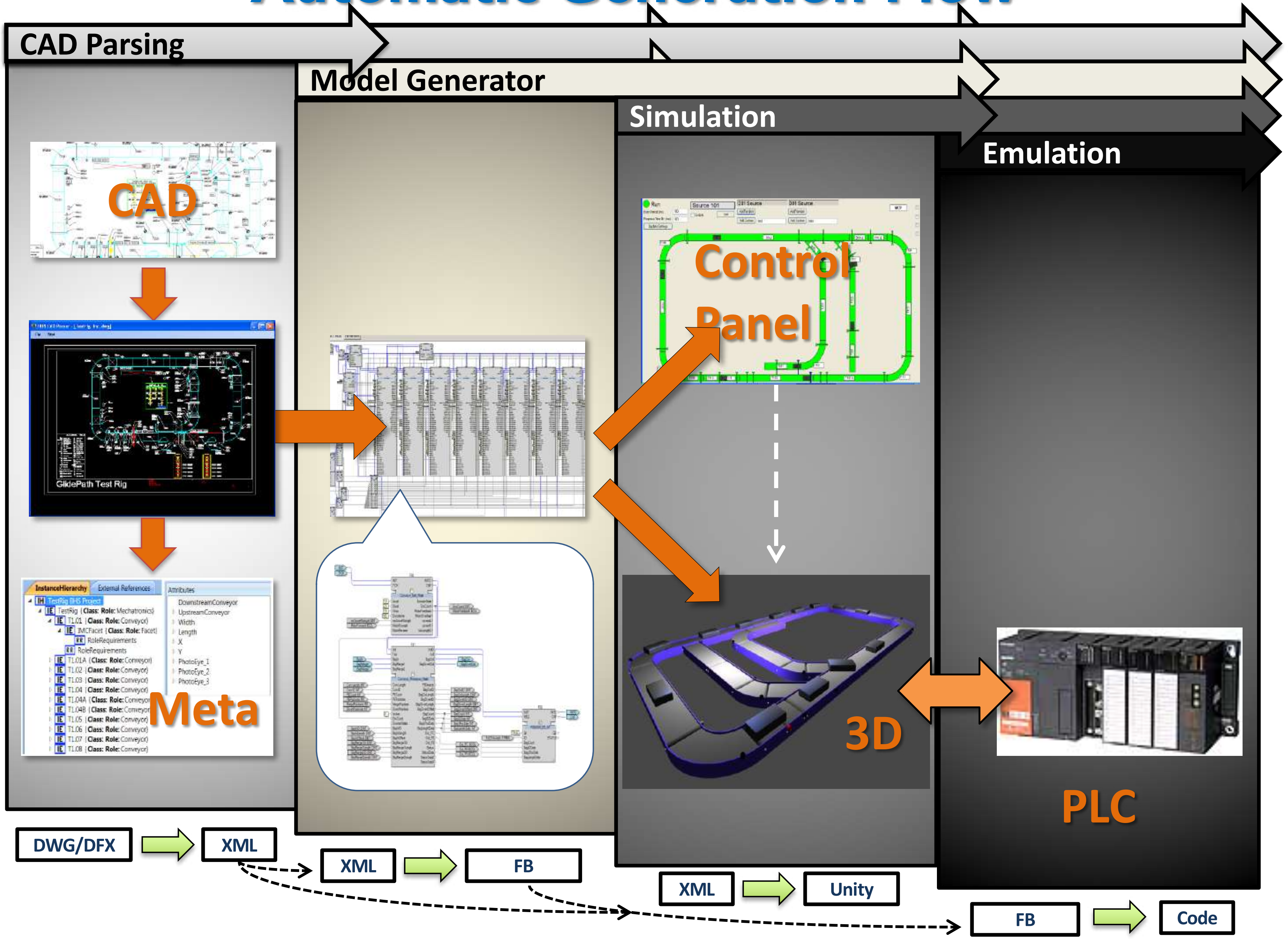
Simulate and render

Central Orchestration





Automatic Generation Flow



CAD Parsing

Model Generator

Simulation

Emulation

CAD
GlidePath Test Rig
InstanceHierarchy
External References
Attributes
TestRig BHS Project
TestRig {Class: Role: Mechatronics}
TL01 {Class: Role: Conveyor}
IMCFacet {Class: Role: Facet}
RoleRequirements
RoleRequirements
TL01A {Class: Role: Conveyor}
TL02 {Class: Role: Conveyor}
TL03 {Class: Role: Conveyor}
TL04 {Class: Role: Conveyor}
TL04A {Class: Role: Conveyor}
TL04B {Class: Role: Conveyor}
TL05 {Class: Role: Conveyor}
TL06 {Class: Role: Conveyor}
TL07 {Class: Role: Conveyor}
TL08 {Class: Role: Conveyor}
DownstreamConveyor
UpstreamConveyor
Width
Length
X
Y
PhotoEye_1
PhotoEye_2
PhotoEye_3
Meta
Run
Source 101
Control Panel
3D
PLC

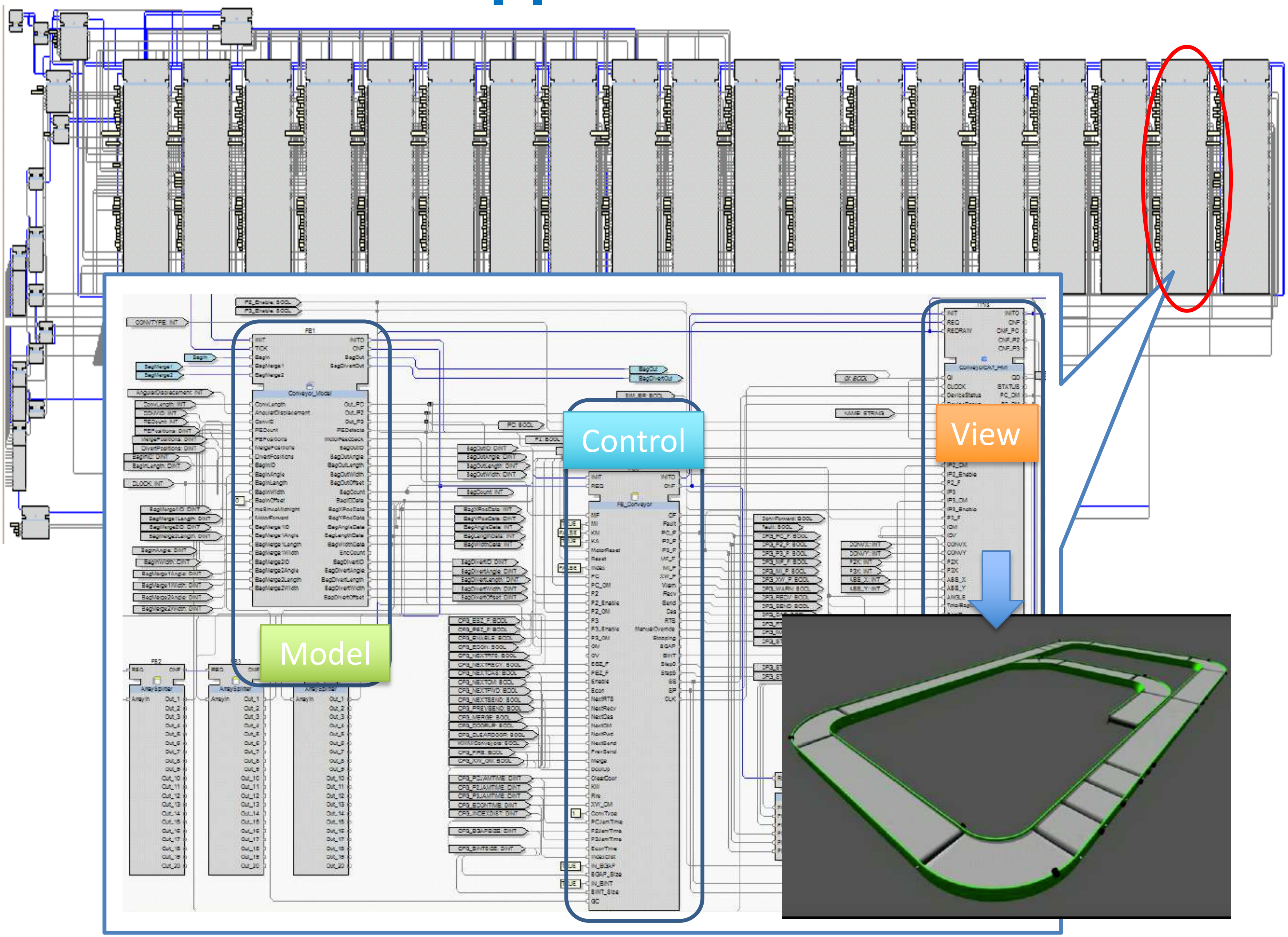
DWG/DFX → XML

XML → FB

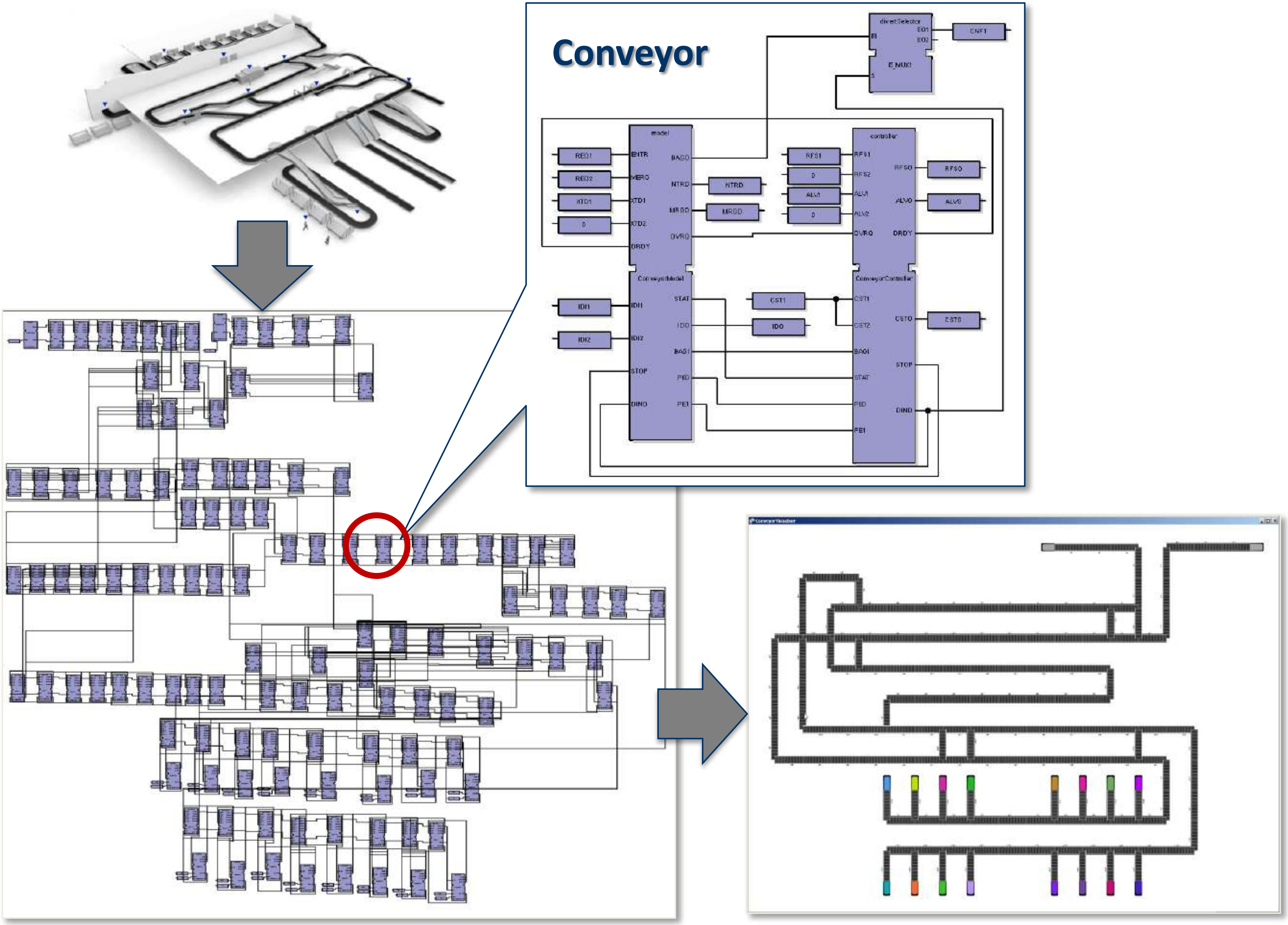
XML → Unity

FB → Code

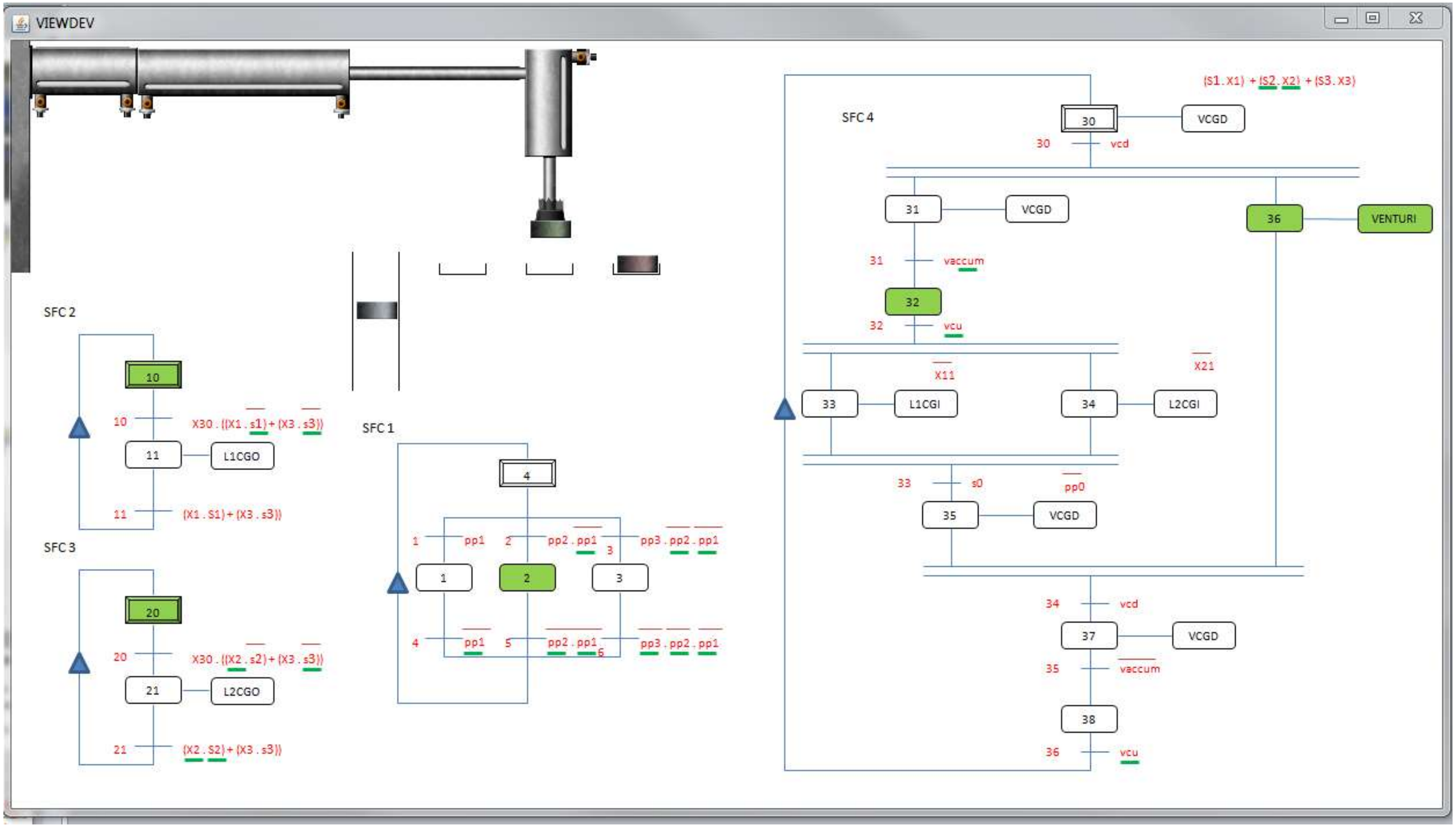
Generated FB Application



Scaling


Conveyor
divertSelector
EI
EO1
EO2
E_MUX1
S
CNF1
model
REQ1
REQ2
XTD1
0
ENTR
MERG
XTD1
XTD2
DRDY
BAGO
NTRD
MRGD
OVRQ
NTRD
MRGD
controller
RFS1
0
ALV1
0
RFS1
RFS2
ALV1
ALV2
OVRQ
RFSO
ALVO
DRDY
RFSO
ALVO
ConveyorModel
IDI1
IDI2
IDI1
IDI2
STOP
DIND
STAT
IDO
BAGI
PE0
PE1
CST1
IDO
ConveyorController
CST1
CST2
BAGI
STAT
PE0
PE1
CSTO
STOP
DIND
CSTO
ConveyorViewer

How do we test automation software?

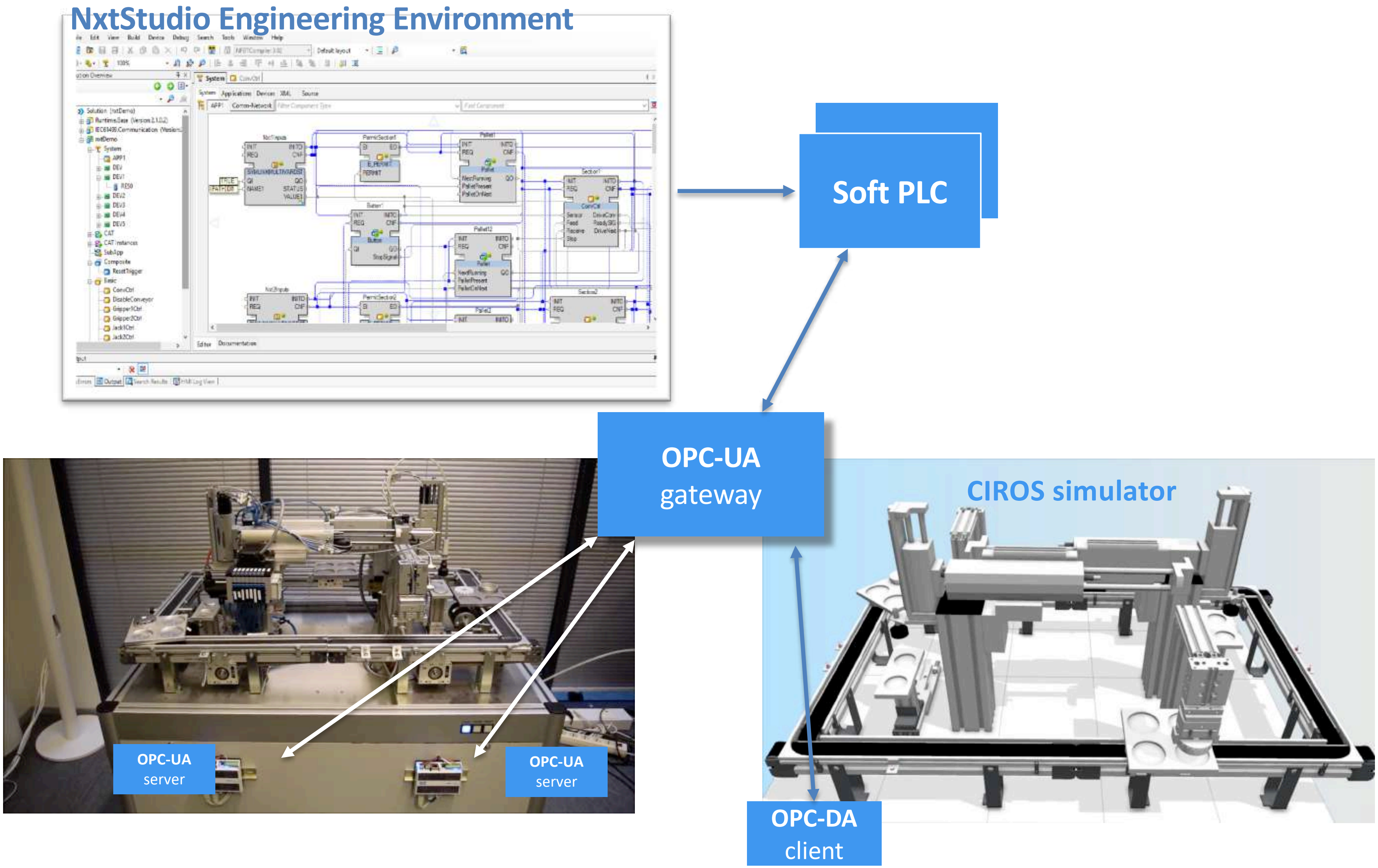


Validation of this code using simulation or in real plant is almost the same

How do we test automation software?

Funtional Testing

Use of Digital Twin for Testing



NxtStudio Engineering Environment

Soft PLC

OPC-UA gateway

CIROS simulator

OPC-UA server

OPC-UA server

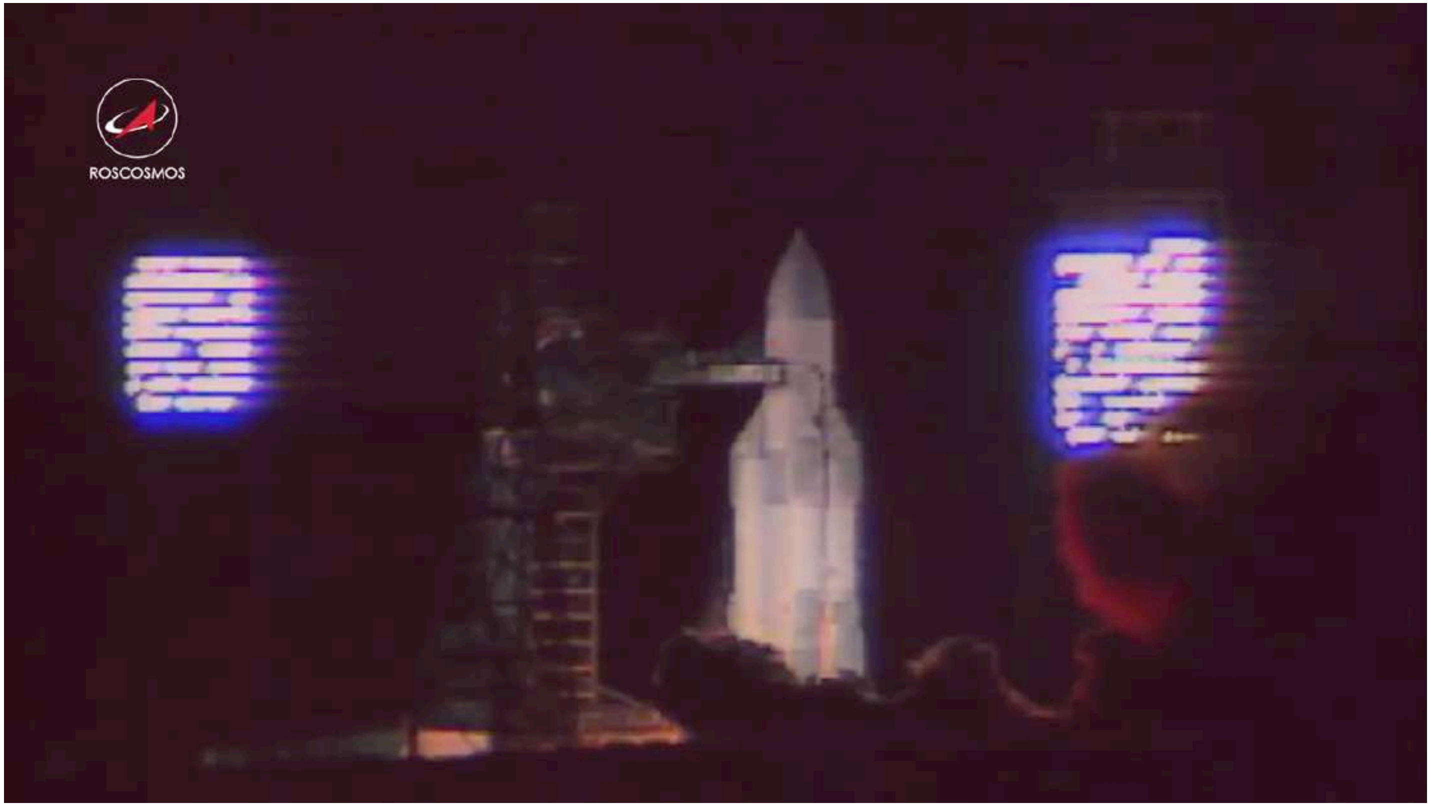
OPC-DA client

Motivation – Why Formal Methods?

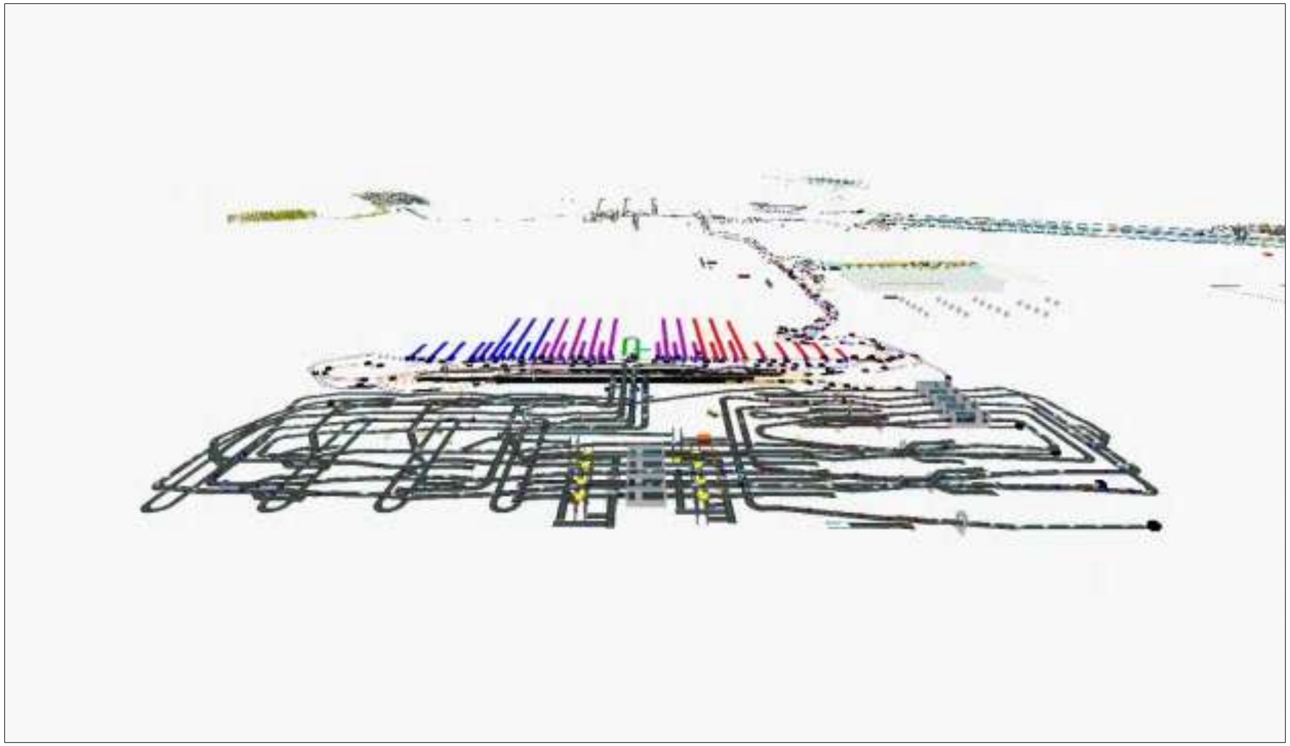
Lack of proper control software verification techniques has led to a number of spectacular technical failures. For example, Ariane 5 launch in 1995

- 64 bit floating point value converted to a 16 bit integer, resulting in overflow and ultimately system shut down
- Engineers thought such a situation will never occur
- **Results in loss of US$375 – US$800 million**

Energy: Russian space shuttle carrier

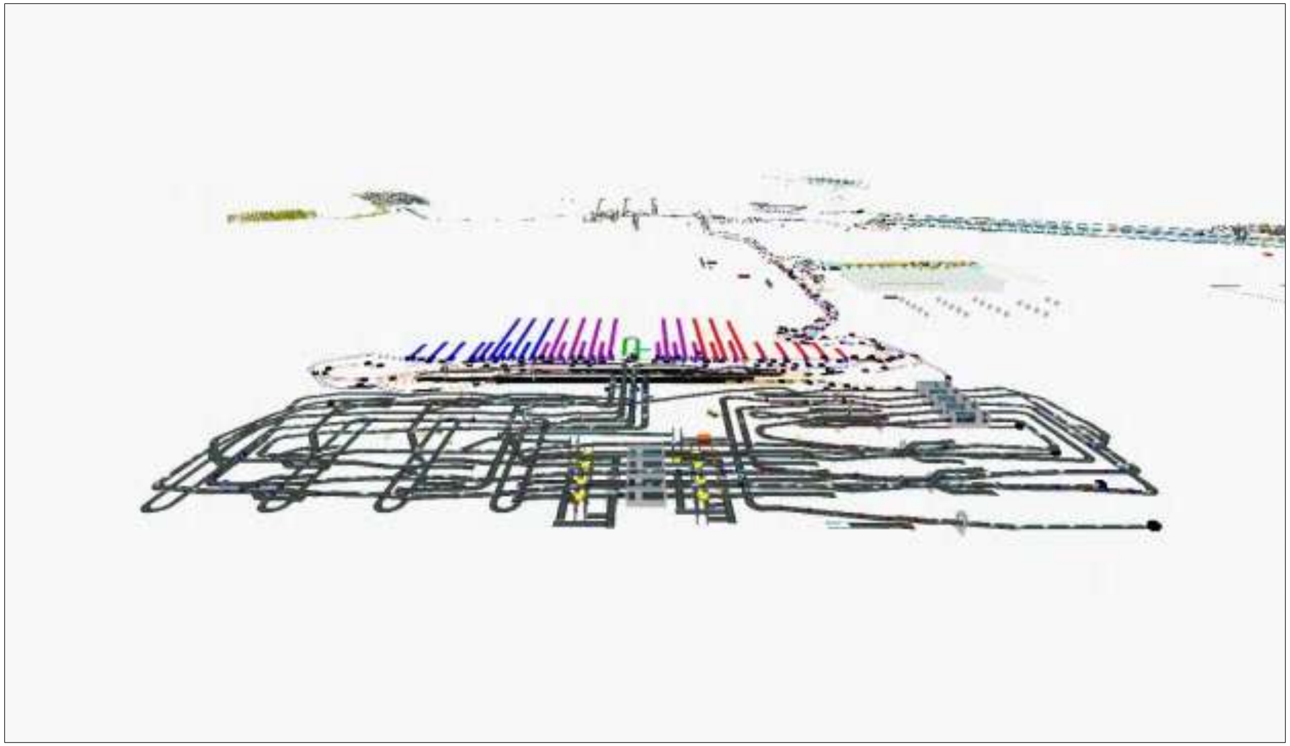


Airport baggage handling



- 30 million bags were temporarily lost by airlines in 2005, and 200,000 of those bags were never reunited with their owners, due to baggage mismanagement and not enough Verification & Validation(V&V)
- Airlines and airports have lost between US$1.6 million to US$2.0 million every year in last 6 years. Rate of increase is 12%

Airport baggage handling



Due to inadequate V&V, there have been delays in delivery of the baggage handling systems to airports, **resulting in losses estimated at US$1 million a day – Denver Airport BHS**

Airbus A400M

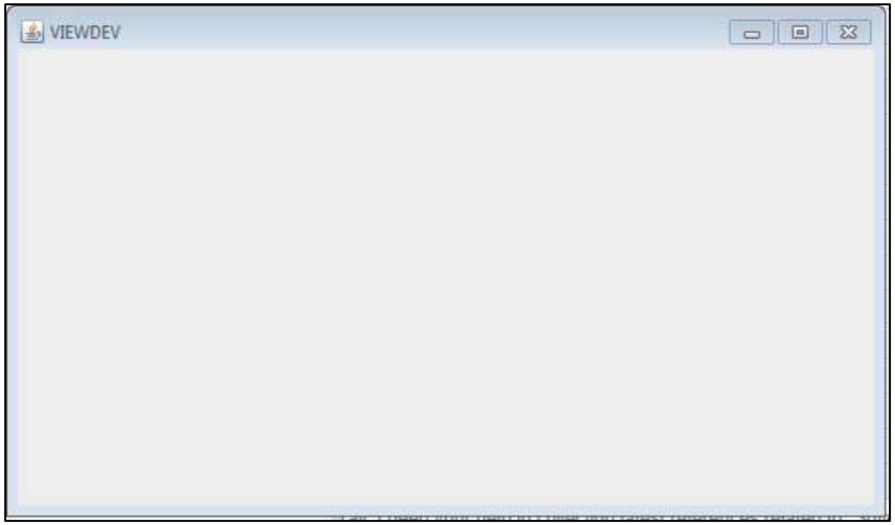
An Airbus A400M crashed in May, 2015 killing four crew members after three out of four engines failed after data was '*accidentally*' deleted on three of the four engines.

http://www.computerworld.com/article/2933491/security0/vital-engine-software-files-accidentally-wiped-linked-to-fatal-a400m-plane-crash.html

What about nuclear power plants ?

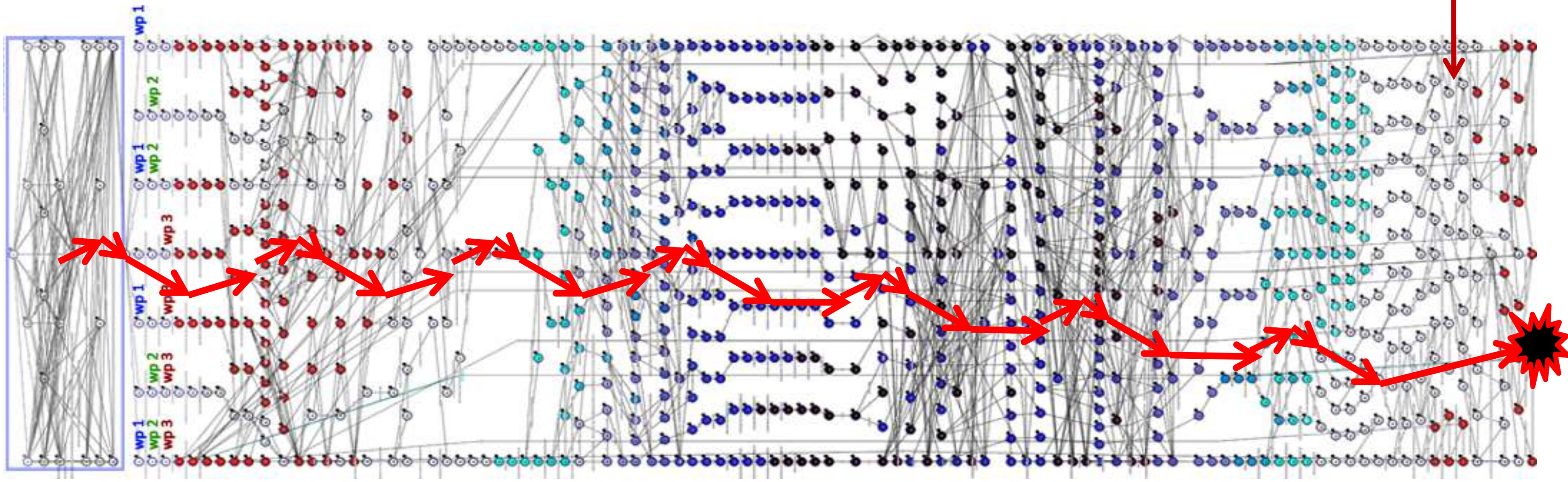
Olkiluoto 3

Limits of Simulation



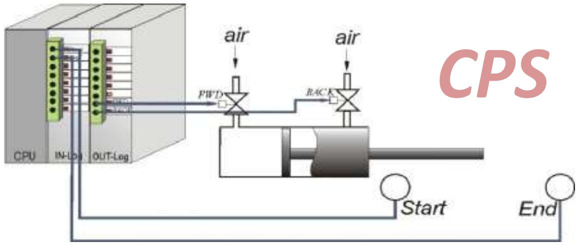
Every simulation run “plays” only one possible behaviour scenario out of the many possible ones.

It is time consuming or impossible to check all of them to ensure safe behaviour of the system

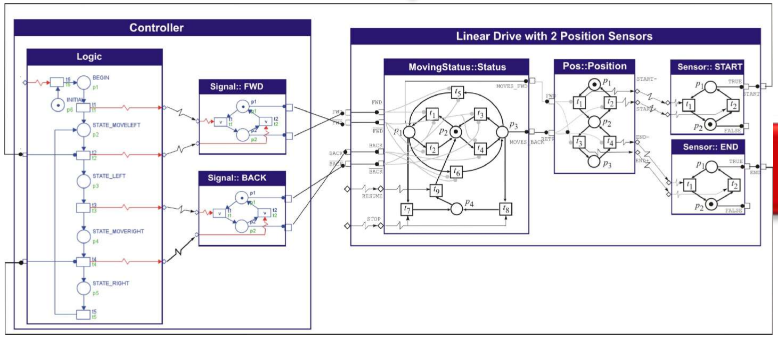


Formal verification tools can explore the complete state-space of the model

Formal Verification of CPS

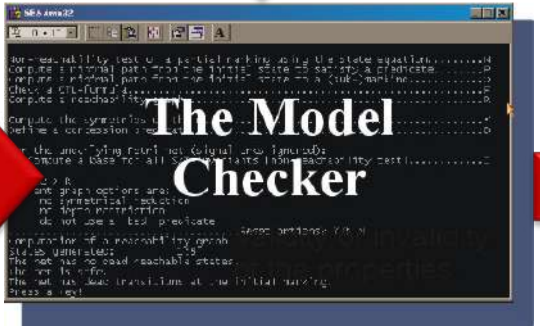


CPS



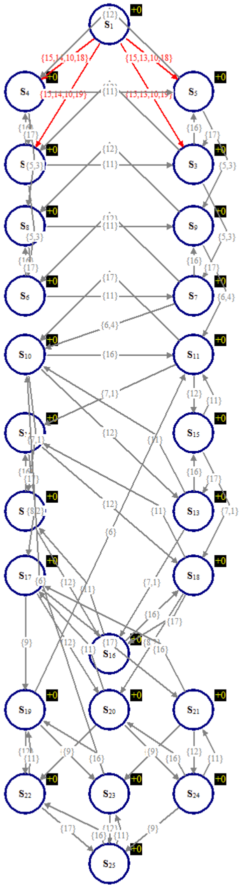
Closed-loop formal model of CPS

EF (BACK and FWD)

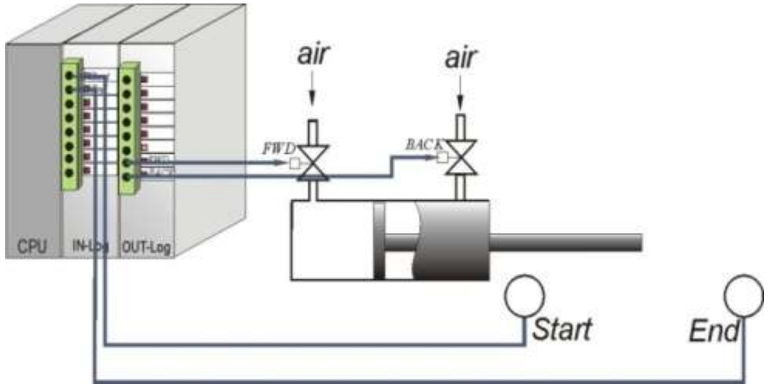


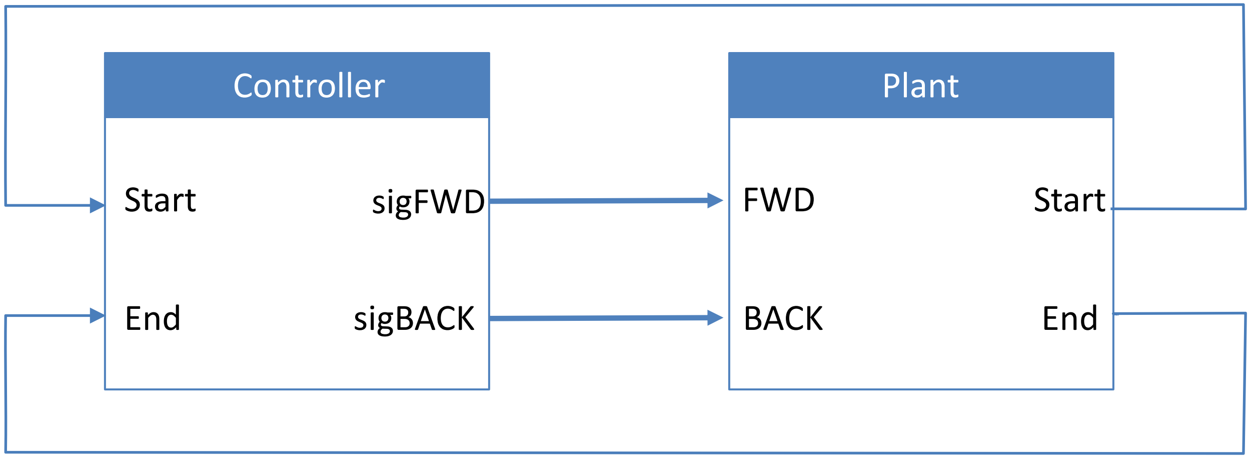
```
Non-reachability test of a partial marking using the state equation.........N
Compute a minimal path from the initial state to satisfy a predicate........P
Compute a minimal path from the initial state to a (sub-)marking............>
Check a CTL-formula.........................................................F
Compute a reachability graph................................................R

Compute the symmetries of the ..............................................*
Define a concession predicate...............................................D

For the underlying Petri net (signal arcs ignored):
  Compute a base for all S-invariants (non-reachability test)...............Z

  > R
  Current graph options are:
    no symmetrical reduction
    no depth restriction
    do not use a test predicate
                                   Change options? (Y/N)
Computation of a reachability graph
States generated:         25
The net has no dead reachable states.
The net is safe.
The net has dead transitions at the initial marking.
Press a key!
```

State space of the CPS



Closed – loop Modelling





Controller

Start

sigFWD

End

sigBACK

Plant

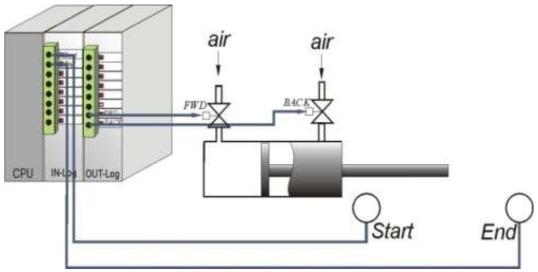
FWD

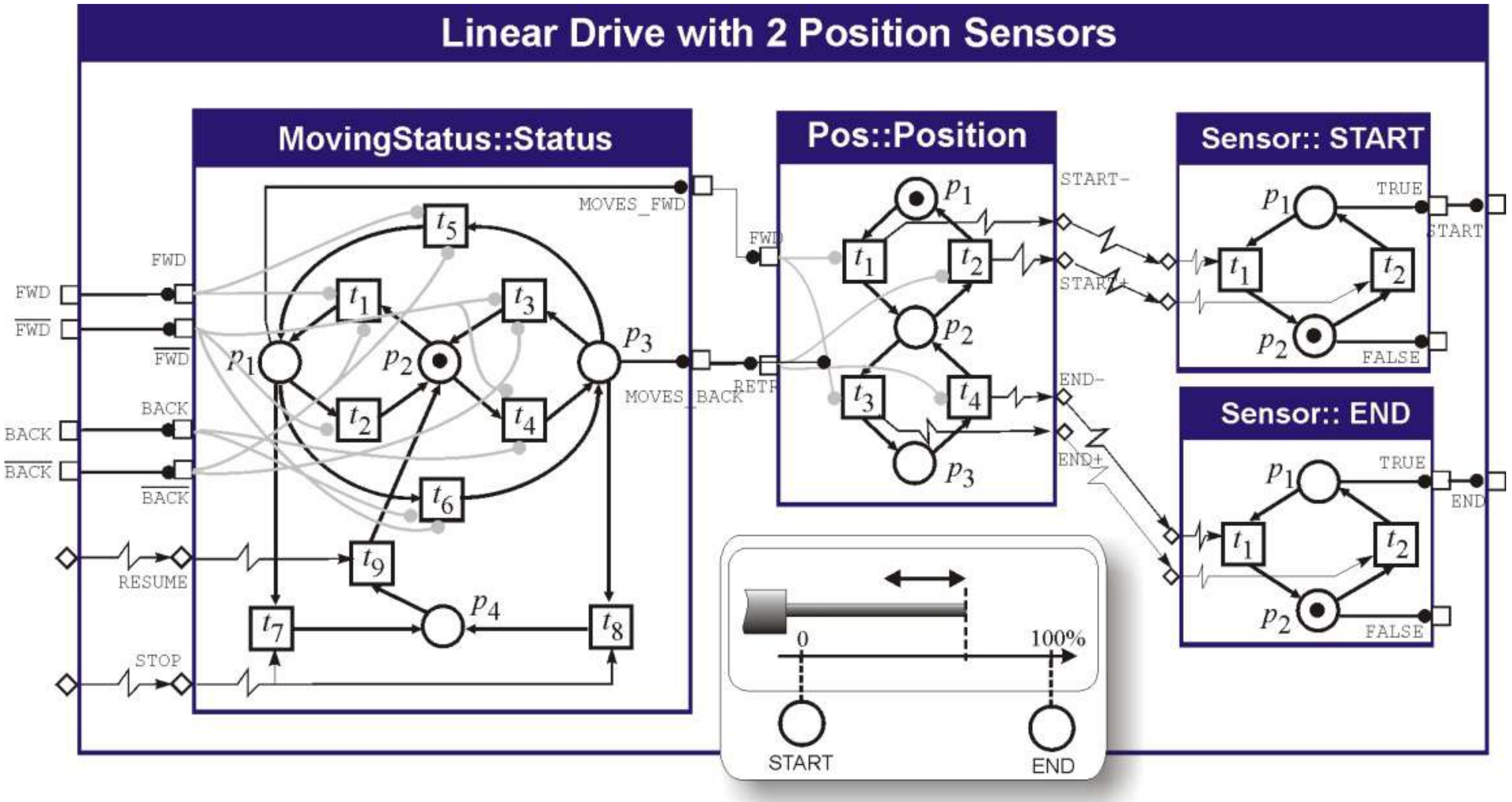
Start

BACK

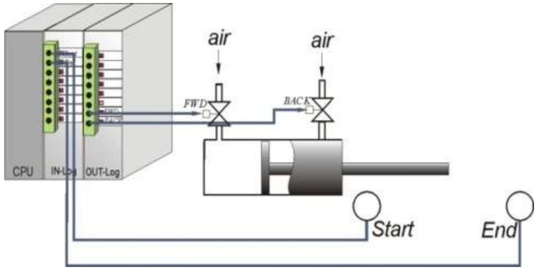
End

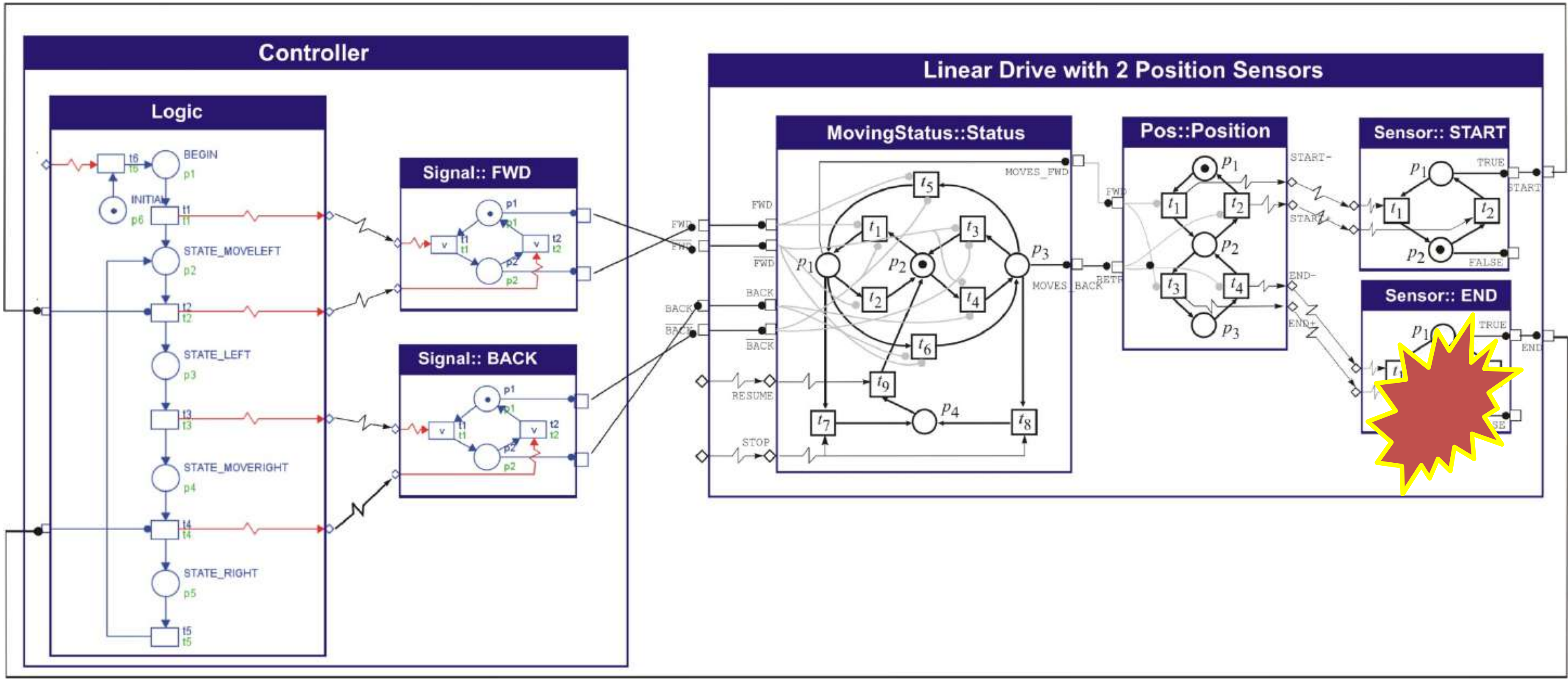
Model of Plant



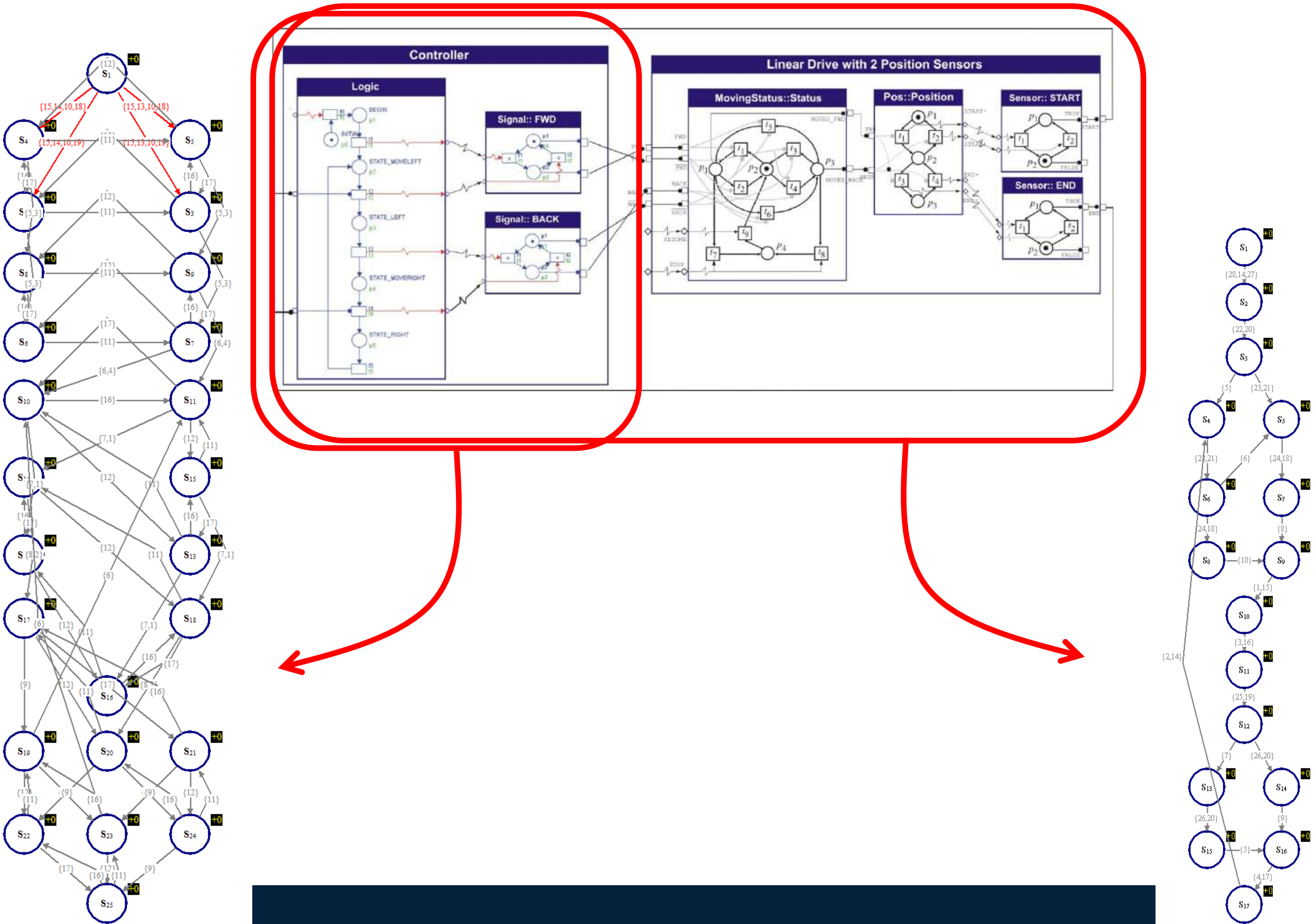


Closed-loop model in Net Condition-Event Systems

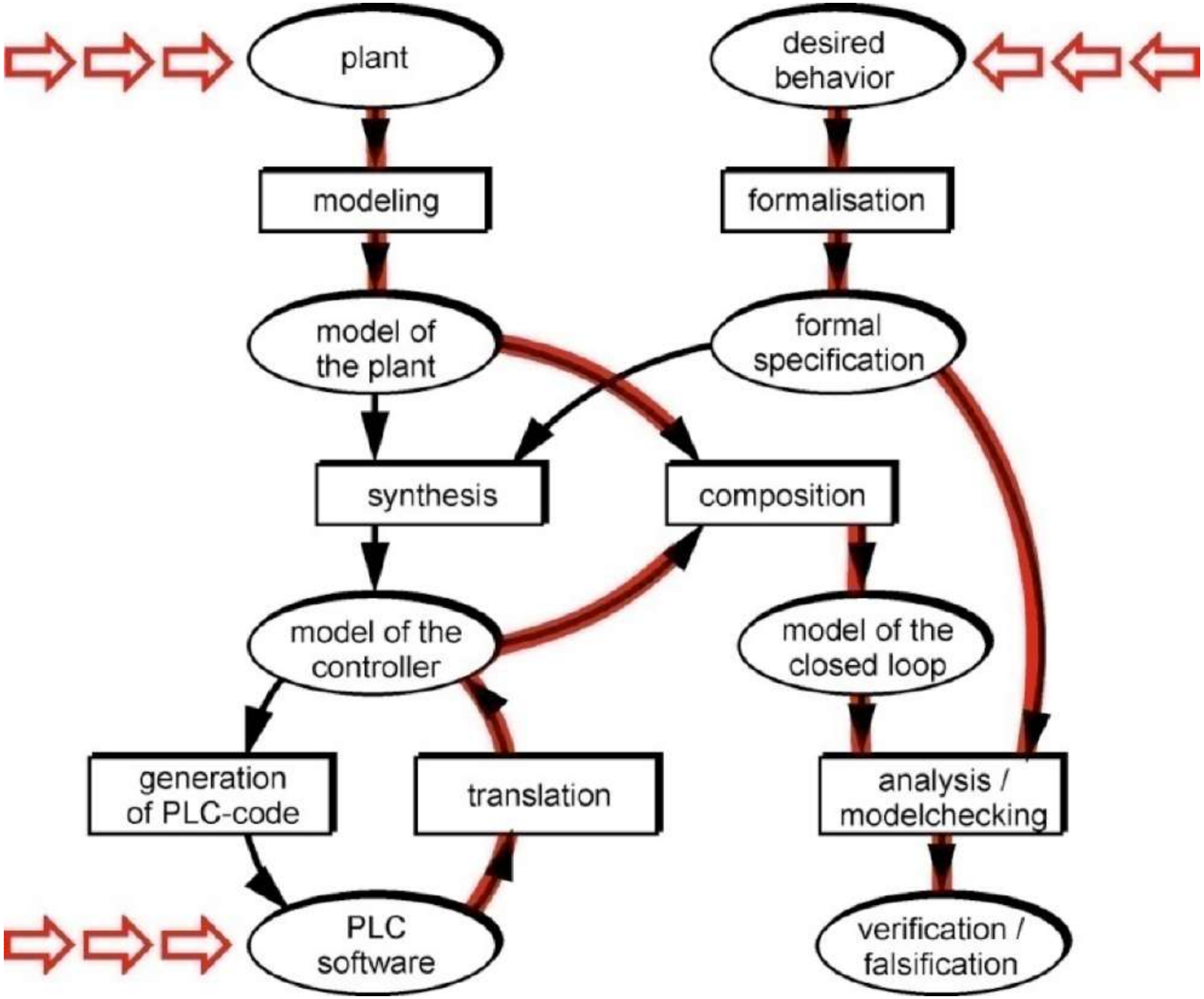




Complexity of Model vs. Complexity of Behaviour



Framework for Formal Methods in Automation (H.-M. Hanisch Diagram)



Challenges for Formal Verification

Who needs it?

- Only nuclear industry in Finland firmly requires it

Complexity

- Of model-checking
- Of model-creation
- Symbolic model-checking of large models

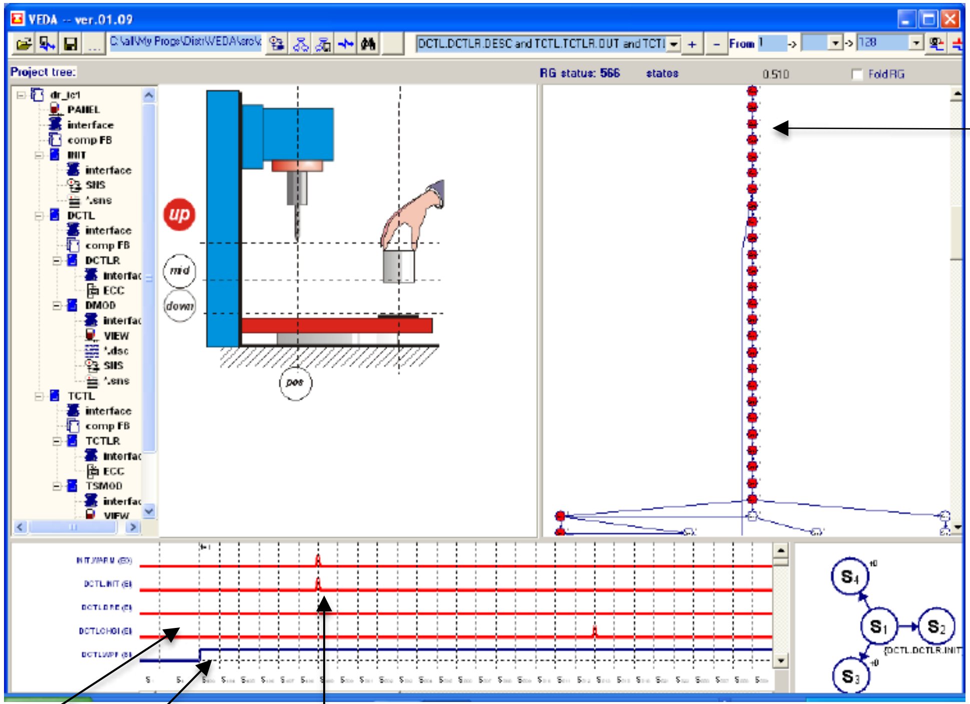
User-friendliness

- Model-generation
- Requirements
- Interpretation of counter-examples
- Integration to the routines

Trust to models

…

Prototype of IDE with Formal Verification



VEDA -- ver.01.09

C:\all\My Progs\Distr\VEDA\src\

DCTL.DCTLR.DESC and TCTL.TCTLR.OUT and TCTI

From 1 -> 128

RG status: 566 states

0.510

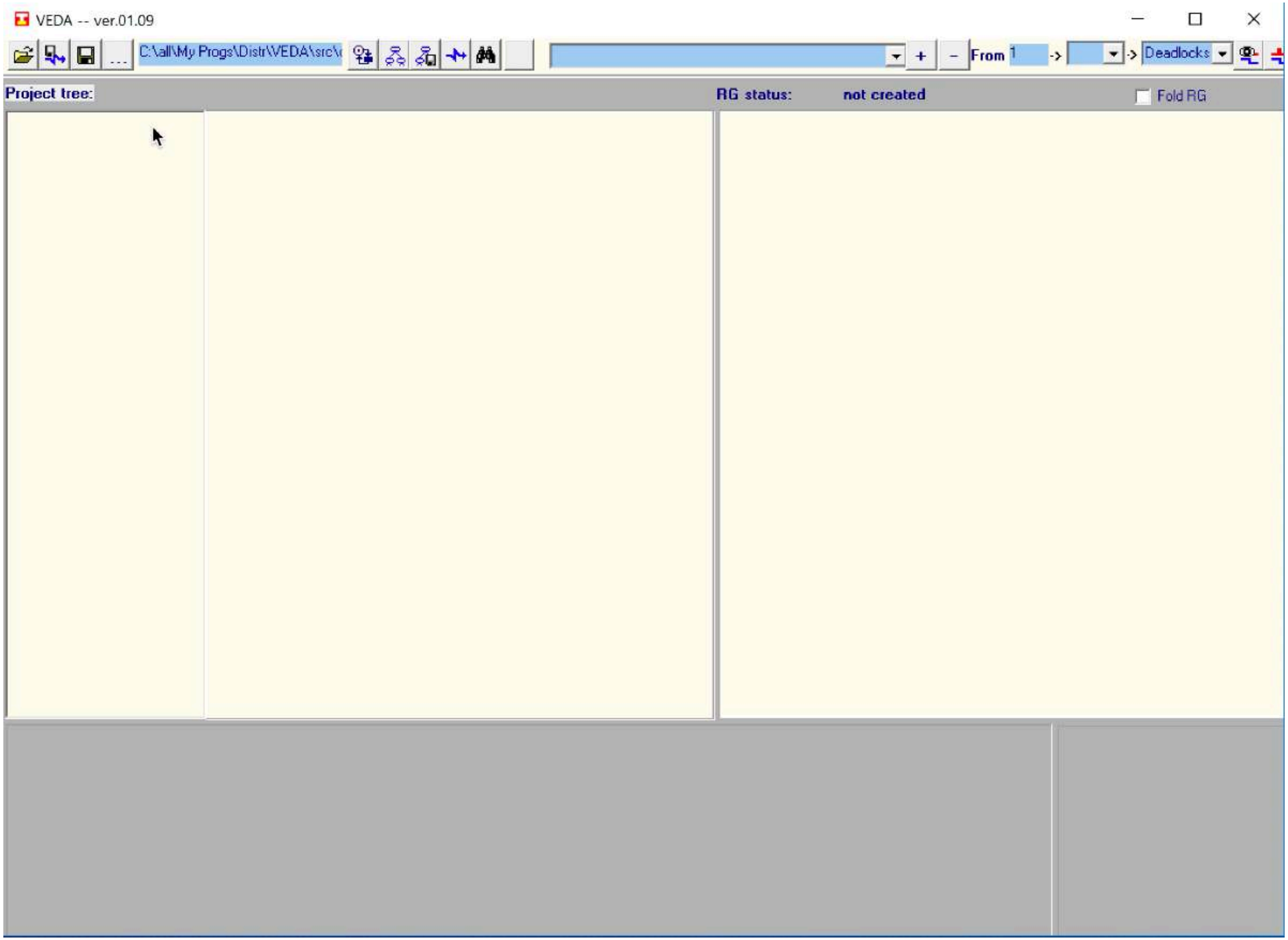
□ Fold RG

Reachability graph with a path to a state (counterexample) highlighted

Some state transitions have a non-zero time duration (plant), while others have no duration (controller).

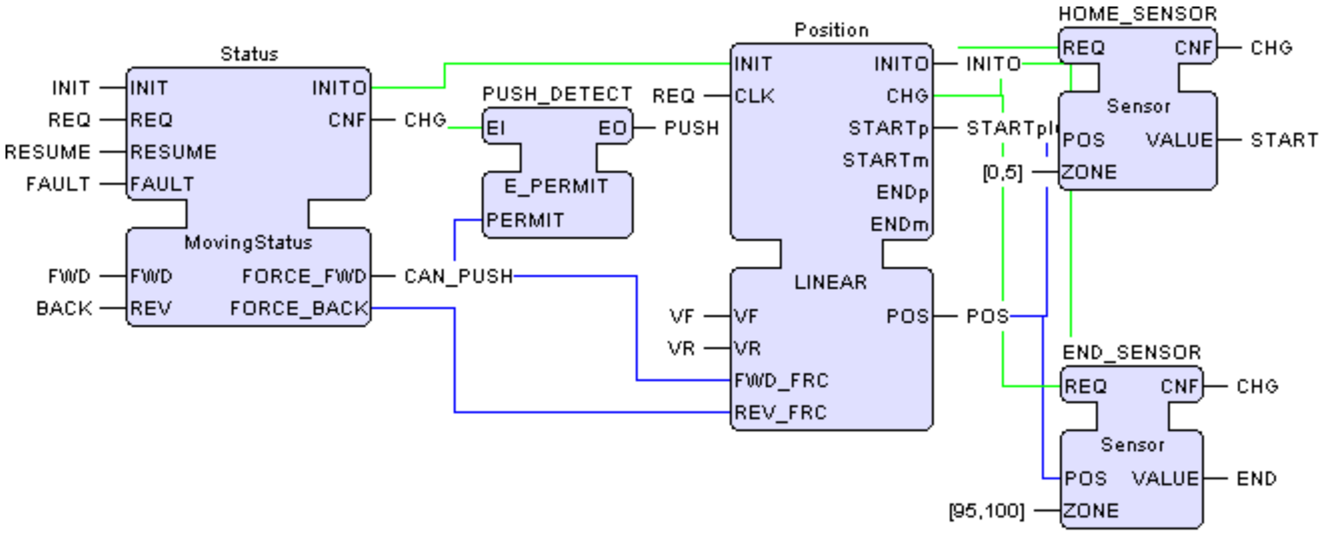
Each operation in controller corresponds to one state transition

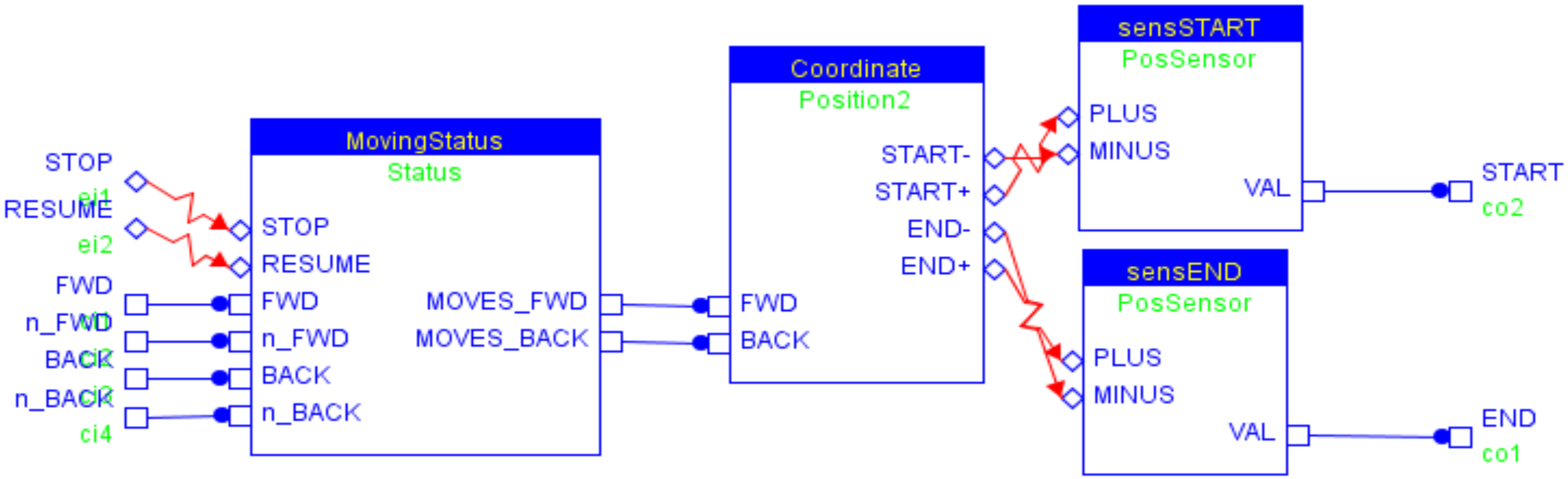
VEDA in work

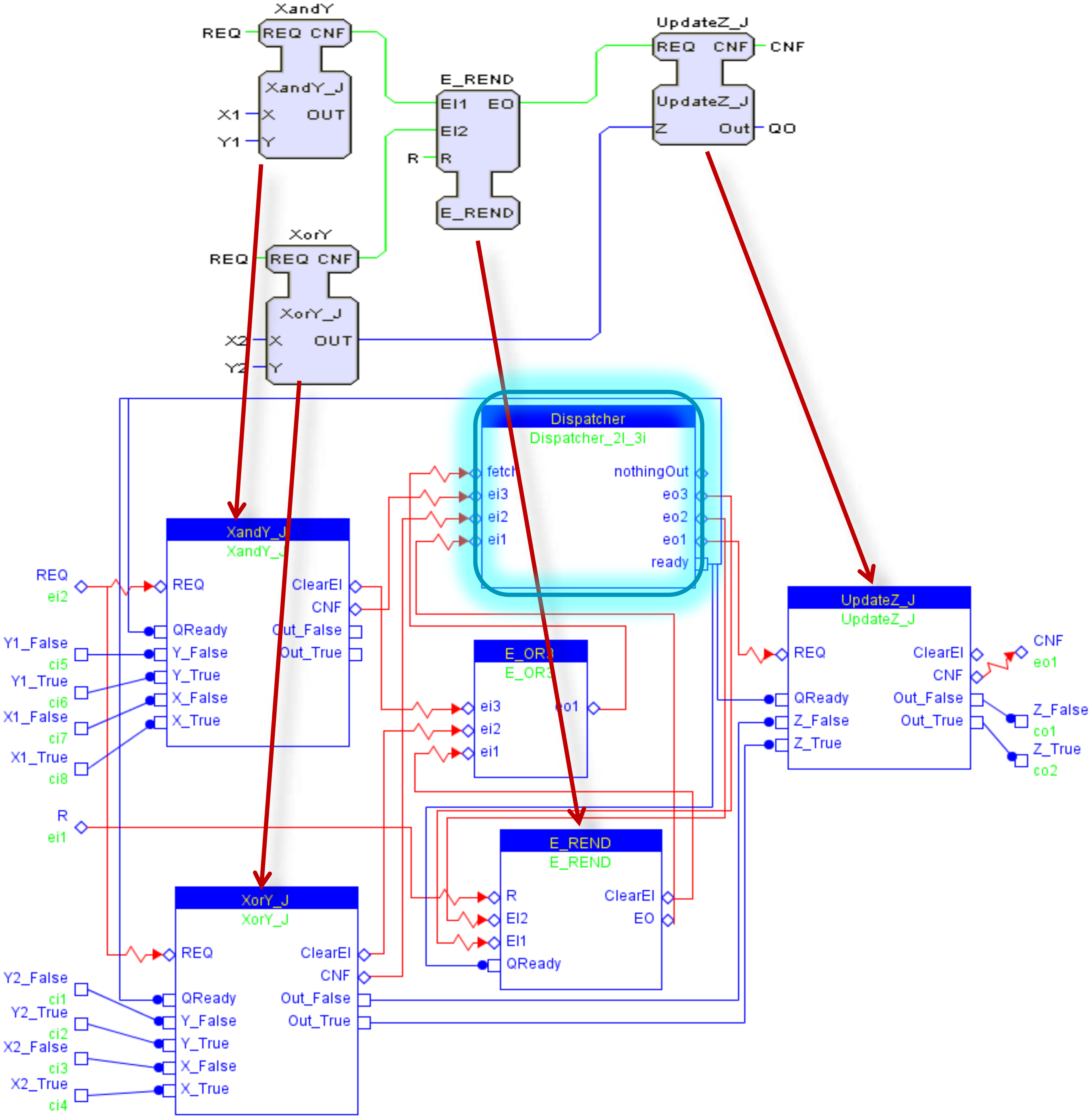

VEDA -- ver.01.09
C:\all\My Progs\Distr\VEDA\src\
+
–
From
1
->
->
Deadlocks
Project tree:
RG status:
not created
□ Fold RG

Executable model

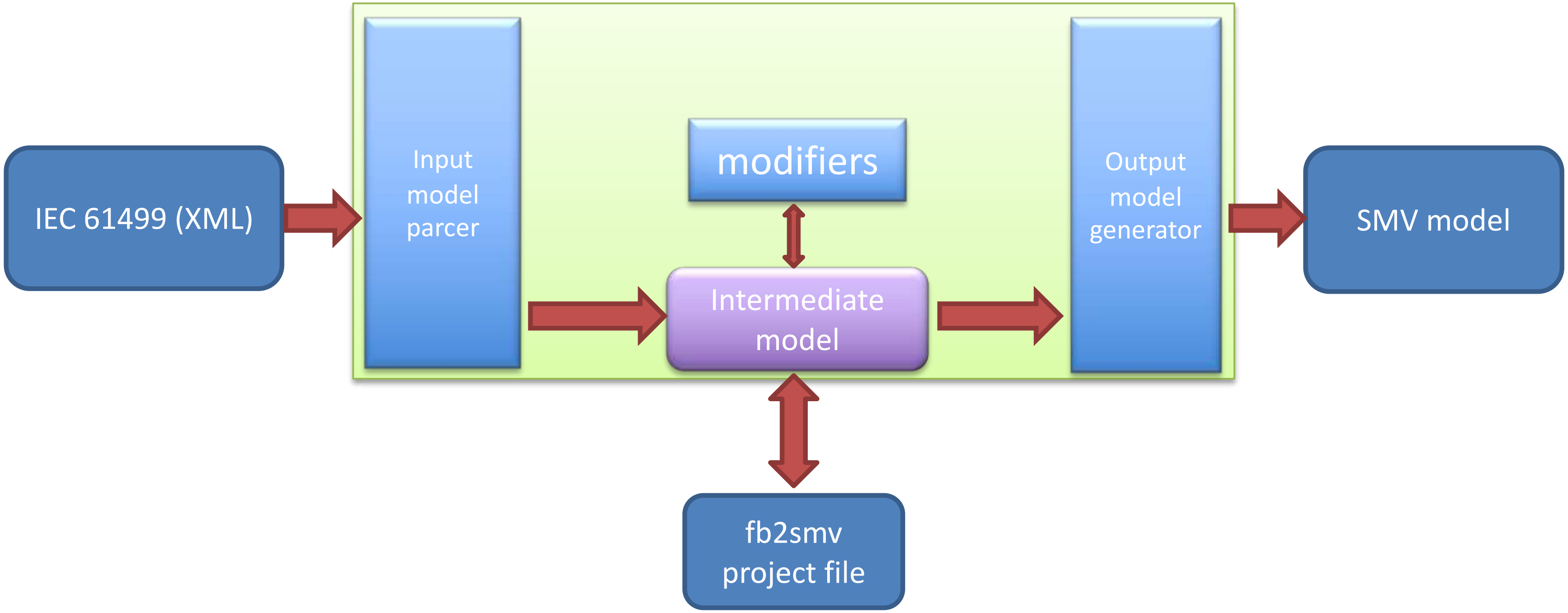
Formal model





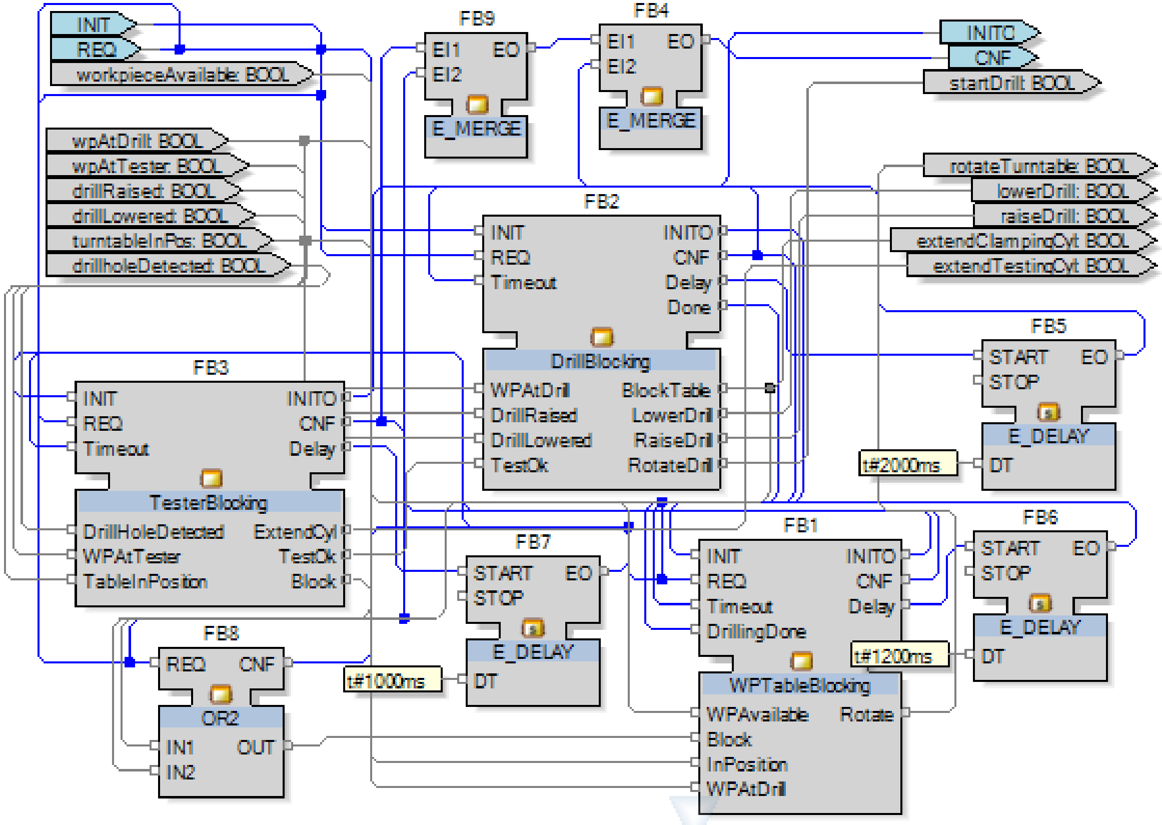

XandY
REQ
REQ CNF
XandY_J
X1
X
OUT
Y1
Y
E_REND
EI1 EO
EI2
R
R
E_REND
UpdateZ_J
REQ CNF
CNF
UpdateZ_J
Z
Out
QO
XorY
REQ
REQ CNF
XorY_J
X2
X
OUT
Y2
Y
Dispatcher
Dispatcher_2I_3i
fetch
ei3
ei2
ei1
nothingOut
eo3
eo2
eo1
ready
XandY_J
XandY_J
REQ
ei2
REQ
ClearEI
CNF
QReady
Out_False
Out_True
Y1_False
ci5
Y_False
Y1_True
ci6
Y_True
X1_False
ci7
X_False
X1_True
ci8
X_True
E_OR3
E_OR3
ei3
ei2
ei1
eo1
UpdateZ_J
UpdateZ_J
REQ
ClearEI
CNF
eo1
QReady
Out_False
Z_False
Out_True
Z_False
co1
Z_True
Z_True
co2
R
ei1
E_REND
E_REND
R
ClearEI
EI2
EO
EI1
QReady
XorY_J
XorY_J
REQ
ClearEI
CNF
QReady
Out_False
Out_True
Y2_False
ci1
Y_False
Y2_True
ci2
Y_True
X2_False
ci3
X_False
X2_True
ci4
X_True

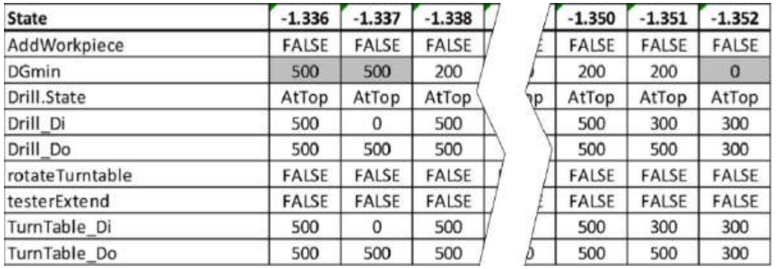
Tool: fb2smv



C# .NET 4.0 / MSVS 2015

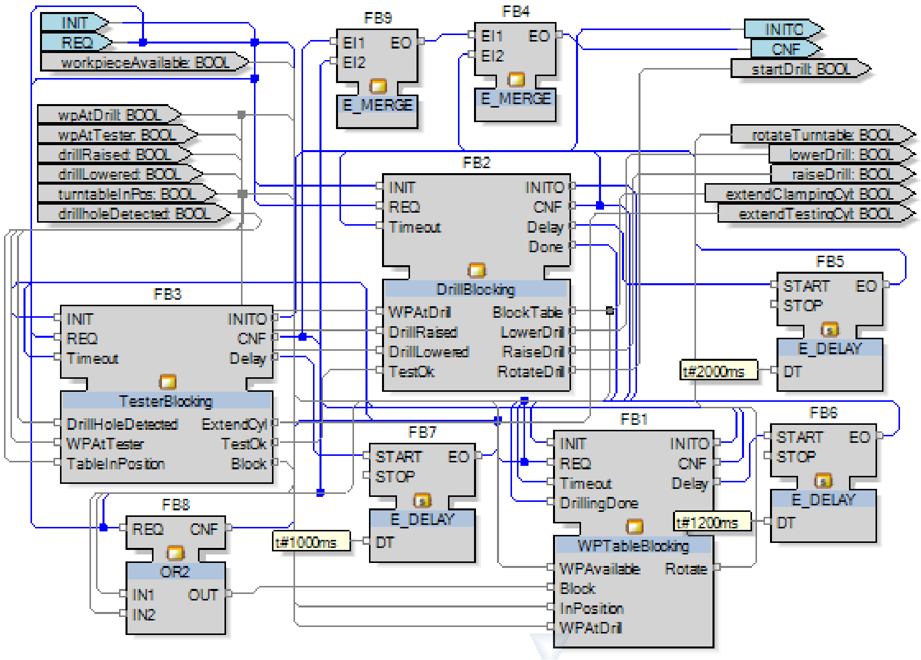
Case study: FESTO MPS 500 processing station



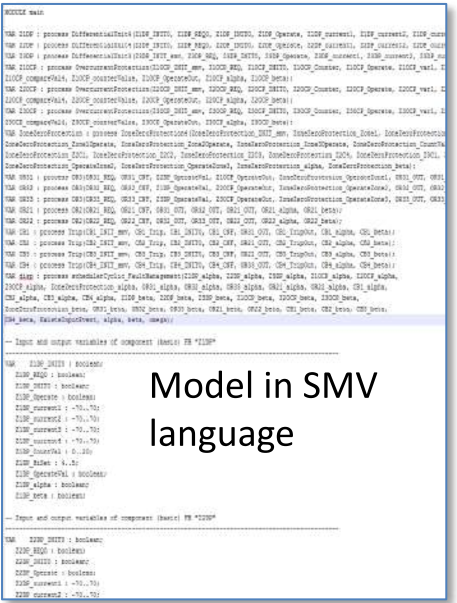


State	-1.336	-1.337	-1.338	-1.350	-1.351	-1.352
AddWorkpiece	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
DGmin	500	500	200	200	200	0
Drill.State	AtTop	AtTop	AtTop	AtTop	AtTop	AtTop
Drill_Di	500	0	500	500	300	300
Drill_Do	500	500	500	500	500	300
rotateTurntable	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
testerExtend	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
TurnTable_Di	500	0	500	500	300	300
TurnTable_Do	500	500	500	500	500	300

Model-checking with fb2smv



FB2SMV Generator



```
MODULE main

VAR Z1DP : process DifferentialUnit4(Z1DP_INITO, Z1DP_REQO, Z1DP_INITO, Z1DP_Operate, Z1DP_current1, Z1DP_current2, [illegible]
[illegible]
CB4_beta, FalseInputEvent, alpha, beta, omega);

-- Input and output variables of component (basic) FB "Z1DP"
------------------------------------------------------------------------
VAR    Z1DP_INITO : boolean;
   Z1DP_REQO : boolean;
   Z1DP_INITO : boolean;
   Z1DP_Operate : boolean;
   [illegible]

-- Input and output variables of component (basic) FB "Z2DP"
------------------------------------------------------------------------
VAR    Z2DP_INITO : boolean;
   Z2DP_REQO : boolean;
   [illegible]
```

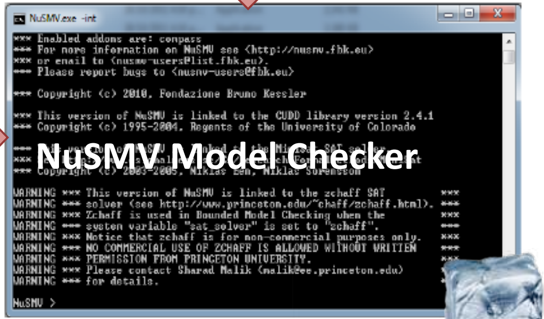
Model in SMV language

Property of the system (IFM) to safely cope with unpredicted changes, e.g.

- Equipment failure (IFM devices, AMU, communication, ROB3 ROB4, ROB5)
- Invalid/unknown inputs (sampling data, ROB3)
- Unexpected disturbances in the system (ROB1, ROB2)
- Intentional attacks (ROB3, ROB4)

Specifications

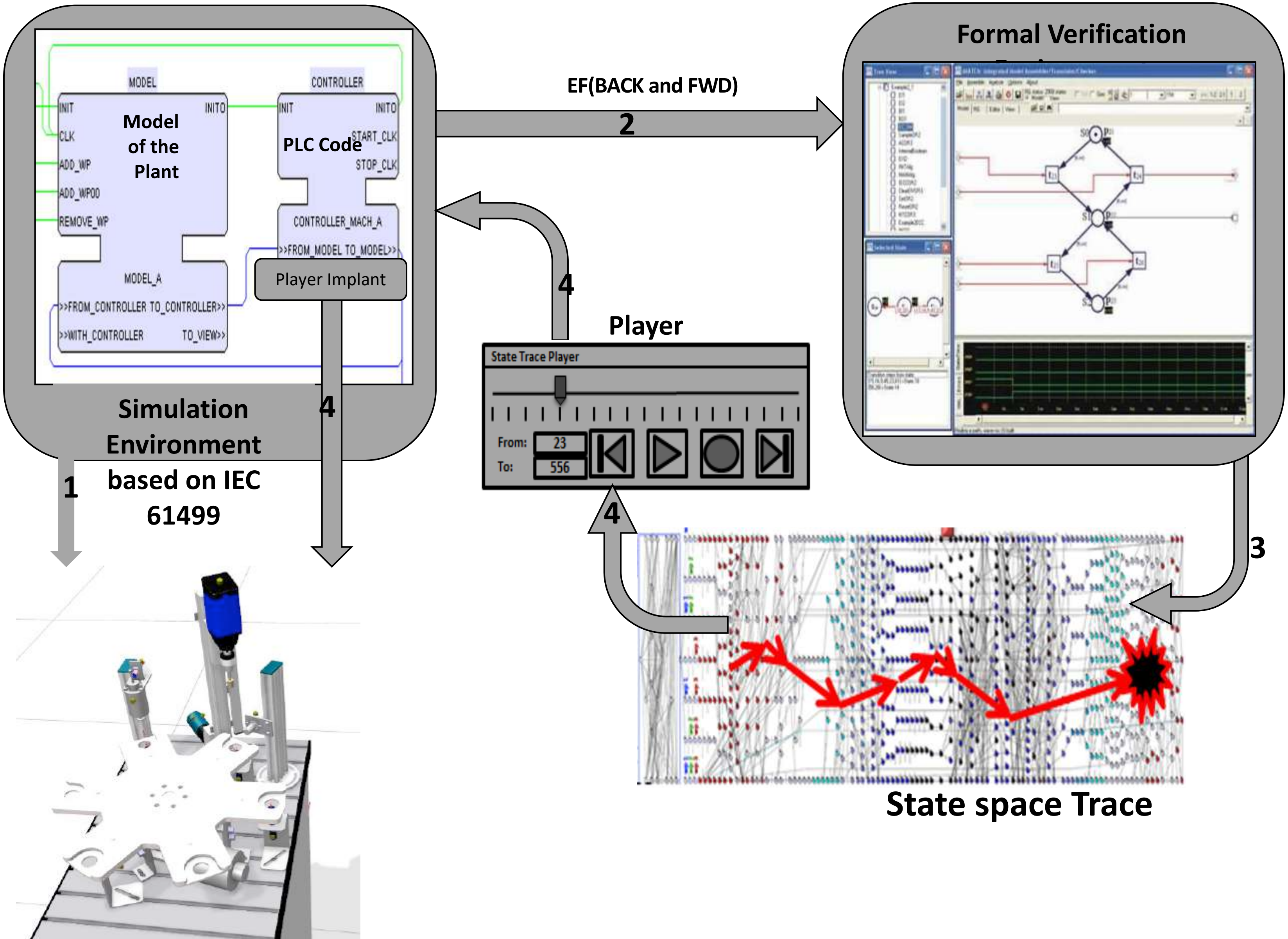
```
SPEC EF alpha
SPEC EF beta
SPEC AG (alpha -> AF (beta))
SPEC EF Z1DP_alpha
SPEC EF Z1DP_beta
SPEC EF Z2DP_alpha
SPEC EF Z2DP_beta
```



```
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <nusmv-users@fbk.eu>

*** Copyright (c) 2010, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2005, Niklas Een, Niklas Sorensson

WARNING *** This version of NuSMV is linked to the zchaff SAT         ***
WARNING *** solver (see http://www.princeton.edu/~chaff/zchaff.html). ***
WARNING *** Zchaff is used in Bounded Model Checking when the         ***
WARNING *** system variable "sat_solver" is set to "zchaff".          ***
WARNING *** Notice that zchaff is for non-commercial purposes only.   ***
WARNING *** NO COMMERCIAL USE OF ZCHAFF IS ALLOWED WITHOUT WRITTEN    ***
WARNING *** PERMISSION FROM PRINCETON UNIVERSITY.                     ***
WARNING *** Please contact Sharad Malik (malik@ee.princeton.edu)      ***
WARNING *** for details.                                              ***

NuSMV >
```

NuSMV Model Checker

Counter-example interpretation



Simulation Environment based on IEC 61499

EF(BACK and FWD)

Formal Verification

Player

State space Trace