Cyber-Physical Components as building blocks of more dependable industrial CPS

Valeriy Vyatkin
vyatkin@ieee.org

Aalto University
School of Electrical Engineering

ITMO UNIVERSITY
Industrial Cyber-Physical Systems Buzz

Supporting technologies

- Communicating embedded systems
- Cloud
- Service-oriented architecture

CPS science

- How to make them working in a predictable and robust manner?
- Cyber-Physical Engineering!

Aalto University
Cyber-Physical phenomena: cross influences
Components at any level of abstraction should be made predictable and reliable if this is technologically feasible. If it is not technologically feasible, then the next level of abstraction above these components must compensate with robustness.*

It is harder to make wireless links predictable and reliable. So we compensate one level up, using robust coding and adaptive protocols.


*Example: Network protocols, retransmissions...
Factory of the Future
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
- 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%
- 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
What about nuclear power plants?
Future of the PLC Technology: merger with IoT
AIC³ laboratory, Luleå

www.ltu.se/AICcube 

Automation, Industrial Computing, Communication and Control
Aalto Factory of the Future

- Assembly system with flexible physical layout
- Enabled by embedded wirelessly communicating controllers
Changing Layout

as a reaction on new product order
How do we test automation software?

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

Closed-loop formal model of CPS

EF (BACK and FWD)

State space of the CPS
Closed – loop Modelling

Controller

Start

sigFWD

End

sigBACK

Plant

FWD

Start

BACK

End
Model of Plant

Linear Drive with 2 Position Sensors
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

Each operation in controller corresponds to one state transition.

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

Reachability graph with a path to a state (counterexample) highlighted.
VEDA in work
New Challenges: Distributed Systems of PLCs

PROGRAM PLC_PRG1
VAR
  LCExtended : BOOL;
  LCRetracted: BOOL;
  LuggageArrived: BOOL;
  ExtendLC: BOOL;
  RetractLC: BOOL;
END_VAR
IF LuggageArrived AND NOT LCExtended THEN
  ExtendLC := TRUE;
  RetractLC := FALSE;
ELSIF LCExtended AND SharedVariable THEN
  ExtendLC := FALSE;
  RetractLC := TRUE;
END_IF

PROGRAM PLC_PRG
VAR
  TCRetracted: BOOL;
  TCEExtended: BOOL;
  LuggageRaised: BOOL;
  ExtendTC: BOOL;
  LuggageAway: BOOL;
END_VAR
IF LuggageRaised AND TCRetracted THEN
  ExtendTC := TRUE;
  LuggageAway := FALSE;
  SharedVariable := FALSE;
ELSIF TCEExtended AND NOT LuggageRaised THEN
  ExtendTC := FALSE;
  SharedVariable := TRUE;
END_IF
Distributed Systems of PLCs
New Opportunities: Use of Digital Twin for Testing

NxtStudio Engineering Environment

OPC-UA server

OPC-UA gateway

Soft PLC

OPC-DA client

CIROS simulator
Testing of Component-based Systems
Traditional Control Engineering Approach: Closed-Loop

Control Software

Plant Model
Cyber-Physical Component Architecture
From simulation to formal verification and back

CPCA

$\gamma_i(int) \leftrightarrow \gamma_i(sim)$

$\gamma_i(sim) \rightarrow \gamma_i(ctl)$

$\gamma_i(ctl) \rightarrow \gamma_i(view)$

$\gamma_i(view) \rightarrow \gamma_i(hmi)$

Simulation \hspace{2cm} Deployment

CPCA extended with formal model

$\Phi_i(mod) \leftrightarrow \Phi_i(ctl)$

$\Phi_i(ctl) \rightarrow \Phi_i(view)$

$\Phi_i(view) \rightarrow \Phi_i(hmi)$

Simulation \hspace{1cm} Formal verification \hspace{1cm} Deployment
Integration of CPCs
Distributed Component Architecture of IEC 61499
Function block

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

Behavior of a Basic Function Block is implemented by an execution control chart. Textual algorithms can be invoked upon entering a state.
IEC61499: seamless distribution
Communicating state-machines
Composition of CPCA
Why not the UML/SysML process?

UML diagrams? What?
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_3IlHimNyc](https://www.youtube.com/watch?v=kT_3IlHimNyc)
- Same, but live: [https://www.youtube.com/watch?v=QQwcIcrONMc](https://www.youtube.com/watch?v=QQwcIcrONMc)
CPSC implemented with FBs
Two independent processes
Intersecting cylinders: Mutual Exclusion
Four cylinders

Simulate and render
Central Orchestration

[Diagram of Central Orchestration with symbols and connections]

- WPS1, WPS2, WPS3, WPS4: Components labeled with WPSensor and ACTIVATE
- FB1, FB2, FB3, FB4: Functional blocks labeled with INIT, INTKN, INTFO, CControlTRAS, SRSP, MTXOUT, SREQ1, SCyIV, ADP1
- Connections illustrate the flow and interactions between components
Tool: fb2smv

IEC 61499 (XML) → Input model parser → Intermediary model

Modifiers

Intermediate model → Output model generator → SMV model

fb2smv project file

C# .NET 4.0 / MSVS 2015
Case study: FESTO MPS 500 processing station
Model-checking with fb2smv

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
Counter-example interpretation

1. Simulation Environment based on IEC 61499
2. PLC Code
3. Formal Verification Environment
4. Player Implant

- EF(BACK and FWD)
- State space Trace
Tool for visualizing requirements along counterexamples
Opportunistic verification framework

- Source code of Controller
- Requirements to the system behaviour
- Back-end model checkers
  - Symbolic
  - Explicit state
  - Bounded
- Controller
- Plant
- Simulation model
- Requirements to plant
Nuclear systems verification
Case study: generic PWR model

Control Logic

Simulation Model

Generated using meta-heuristic algorithms

SMV model

Discrete state model

Model-Checker Tool
Default approach: open-loop model checking in MODCHK

- **Inputs are assigned arbitrary values from defined ranges**
- **Request-response properties, spurious actuation properties and invariants can be checked**
- **What if some input value combinations/sequences are impossible, and the controller does not behave well on them?**
- **What if we wish to check a property over inputs?**
Benefits of having the Plant Model

- Nuclear power plant
- Controller: I&C network
- Formal plant model
- MODCHK formal model
- NuSMV / SPIN code
  - Closed-loop model checking (of given temporal specification)

**Pros 1:** can check a wider class of temporal specifications

**Pros 2:** state space reduction (can be used in explicit-state verification, but not in symbolic)

**Cons:** if the model is incomplete, some behaviors can be missed
Where to get the plant models from?
Model Mining Method

Data measured from the real process

Simulation model of the closed-loop automation system

Initial simulation settings:
- plant initial condition
- controller mutations

Set of execution traces (around several hours or days of execution for large systems)

Symbolic model

```
TRANS
(next(output_pressure1) in & (next(output_voltage) in {
& (next(output_pressure1) = & (next(output_pressure2) = & (next(output_pressure2) = & (next(output_voltage) = 1 & (next(output_voltage) = 1 & (next(output_voltage) = 1 & (CONT_INPUT_1 in 0..50 & T & (CONT_INPUT_1 in 0..50 & n & (CONT_INPUT_1 in 0..50 & T
```
Why do we synthesise plant models and not closed-loop models?

Closed-loop model identification

- Plant
- Controller

- Simulation / execution
- Trace recording
- Model generation

Closed-loop model

- Only behaviors which are likely to appear in simulations, but we’d like to have some faults as well

Plant model generation

- Plant
- Mutated controllers

- Simulation only
- Trace recording
- Model generation

Plant model with faults

- Can compose with the controller model (assume that it is available) and model-check

Thus, we construct “semi-controlled” plant models

Simulating without the controller would lead to the lack of non-faulty behaviors
Model construction from behavior examples and specifications

- LTL synthesis may fail. In this case, the specification is incorrect and needs to be refined.
- If it does not fail and there is an example of different behaviors, refine either the model or the specification.
- If there is no such example, relax IO value boundaries.

Textual specification

Manual formalization

LTL specification

LTL synthesis tool

Synthesized formal model

Model-check equivalence

Example of different behaviors

Manually prepared formal model

Refine!
Case study 2: Redundancy in nuclear systems

- Basic network: interconnected basic blocks
- Graphical modeling in a verification tool
- A very typical scenario is to detect certain conditions on the input and generate output accordingly
- This may involve delays, storing information in flip-flops, etc.

Diagram:
- Containment pressure (1..4)
- Reactor pressure vessel water level (1..4)
- High water level in pump room (1..2)
- Pump room water level high (to ALU1..4)
Hardware view: 4 physically isolated redundant divisions

- APU (acquisition and processing unit) and ALU (actuation logic unit = voting unit) are two I&C networks
- There are four pairs of APU/ALU, corresponding to four isolated divisions
- Goal: verify the entire system, accounting for possible failures
Modeling approach: implementation in MODCHK / NuSMV

- MODCHK is a graphical frontend to the model checker NuSMV
- Faults are modeled as replacement of signals with nondeterministic values
- Asynchrony is modeled as communication delays
# Results of model checking

<table>
<thead>
<tr>
<th>Case</th>
<th>Model checking time (s)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BMC</td>
<td>LTL</td>
</tr>
<tr>
<td>No faults, no delays</td>
<td>98</td>
<td>3</td>
</tr>
<tr>
<td>Only delays</td>
<td>117</td>
<td>TL</td>
</tr>
<tr>
<td>Only faults</td>
<td>25125</td>
<td>38</td>
</tr>
<tr>
<td>Both faults and delays</td>
<td>138</td>
<td>TL</td>
</tr>
</tbody>
</table>

- TL (time limit) = Neither of the properties were checked within the time limit of 8 hours
- BDD-based LTL model checking does not scale to the case of delays, even if they are bounded by 1 cycle
- Bounded model checking (BMC) works better
Cyber-physical engineering is a new approach that is aiming at addressing *essentially new cyber-physical phenomena* of automation systems.

It relies on the newly developed cyber-physical science that is aiming at predicting and guaranteeing dependability properties of complex interconnected systems.
Cyber-physical engineering

• Wide use of the physical system models at all stages of system design and life-cycle

• Use of new cyber-physical formal languages that are capable of capturing efficiently properties of both physical world and cyber world

• Computations in the cyber part are enhanced with awareness of time, error, energy and other important physical parameters

• Use of design documents of both physical world and cyber world with ability to interconnect them and derive properties of one from another.

• Use of cyber power to create and validate models
Credits

Hans-Michael Hanisch

Igor Buzhinsky

Dmitrii Droz dov

Sandeep Patil

Cheng Pang

Edmund Clarke