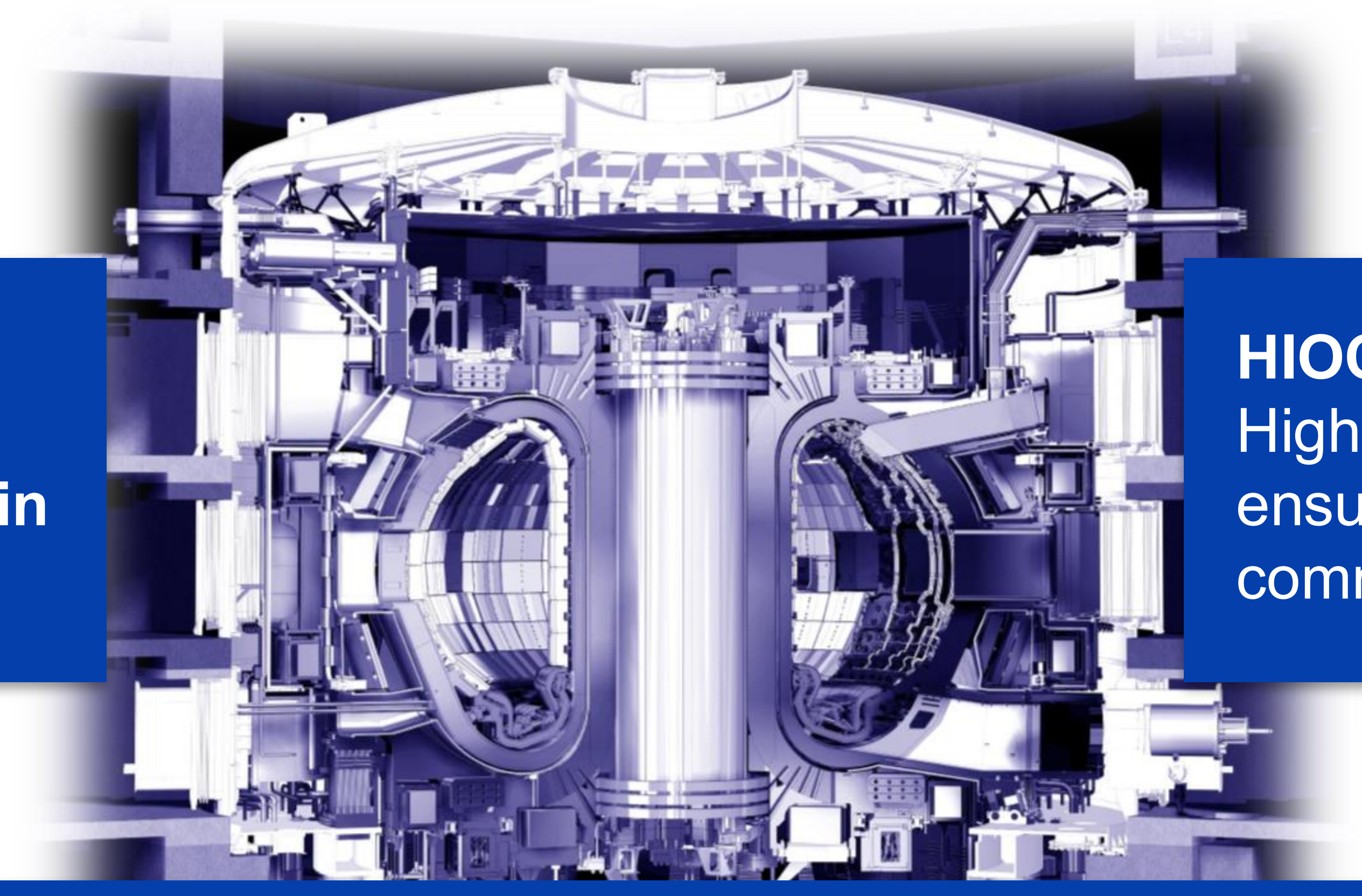
Applying model checking to critical PLC applications: An ITER case study

THPHA161

B. Fernández, D. Darvas, E. Blanco, CERN, Geneva, Switzerland
Gy. Sallai, BME, Budapest, Hungary
I. Prieto, IBERINCO, Madrid, Spain
G. Lee, Mobiis Co. Ltd., Seoul, South Korea
B. Avinashkrishna, Y. Gaikwad, S. Sreekuttan, Tata Consultancy Services, Pune, India
R. Pedica, Vitrociset s.p.a, Rome, Italy



ITER
World collaboration to build the first **fusion device** producing **net energy** and to **maintain fusion for long periods** of time.

HIOC
High-integrity communication protocol to ensure safe masking of interlocks for commissioning and maintenance.

GOAL: Verification and better understanding of the **PLC program** implementing the **HIOC protocol**

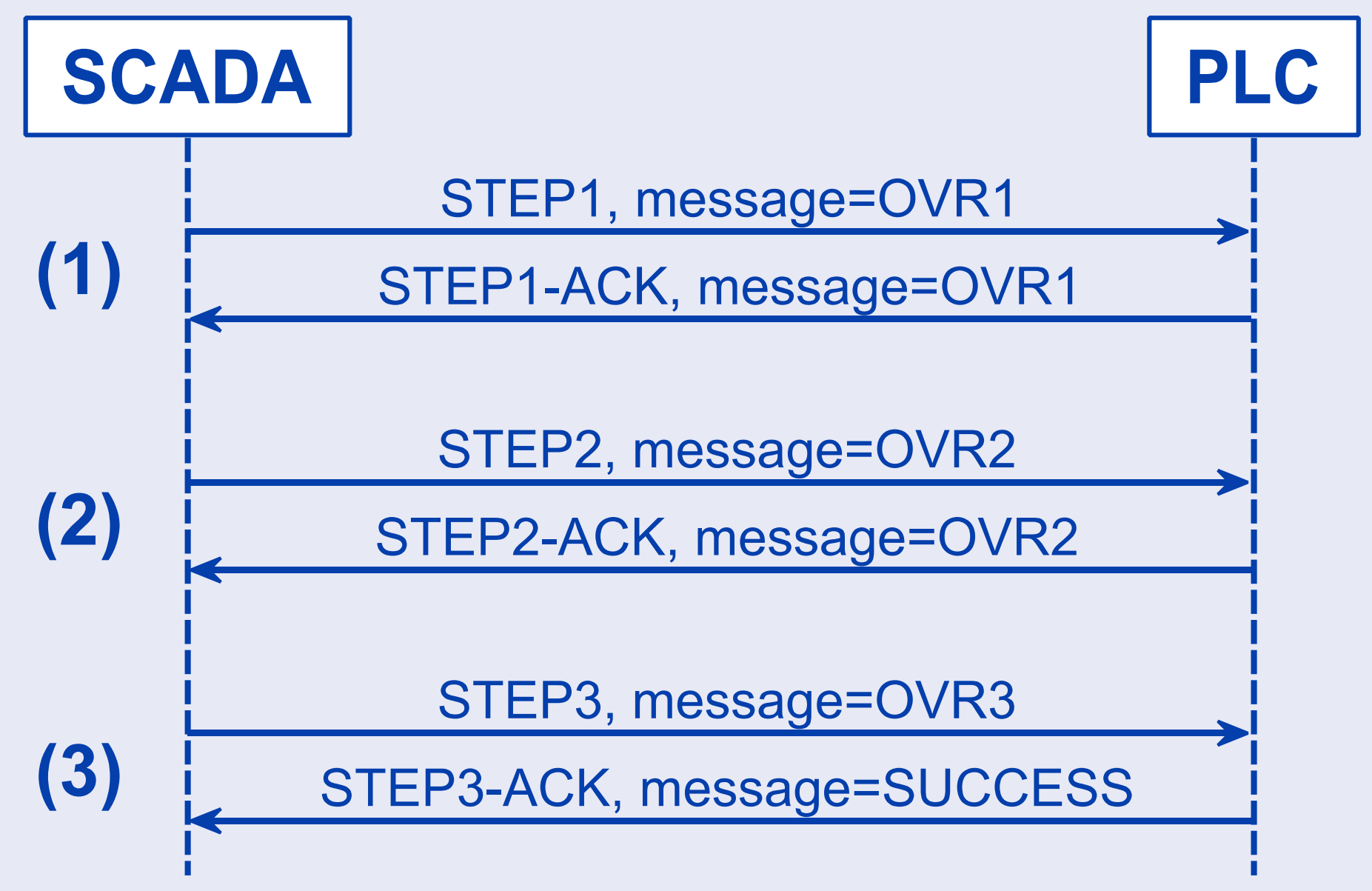
Protocol

Black channel approach

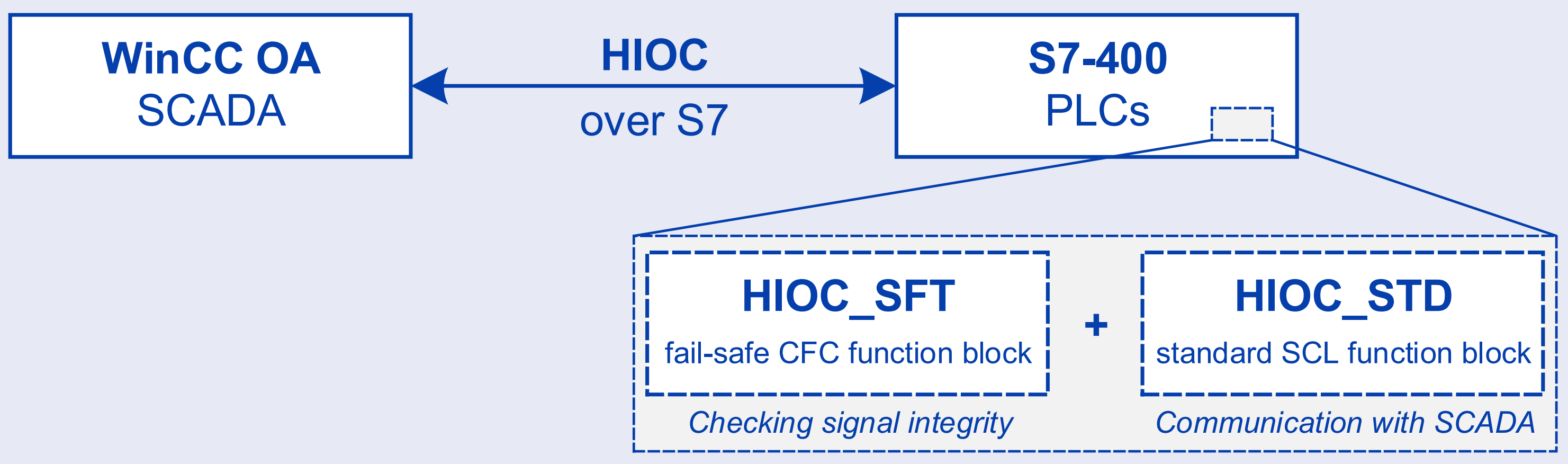
- As defined in IEC 61508
- No guarantees expected from the underlying communication protocols

Three-phase protocol

- Ensures the integrity of the transmitted message

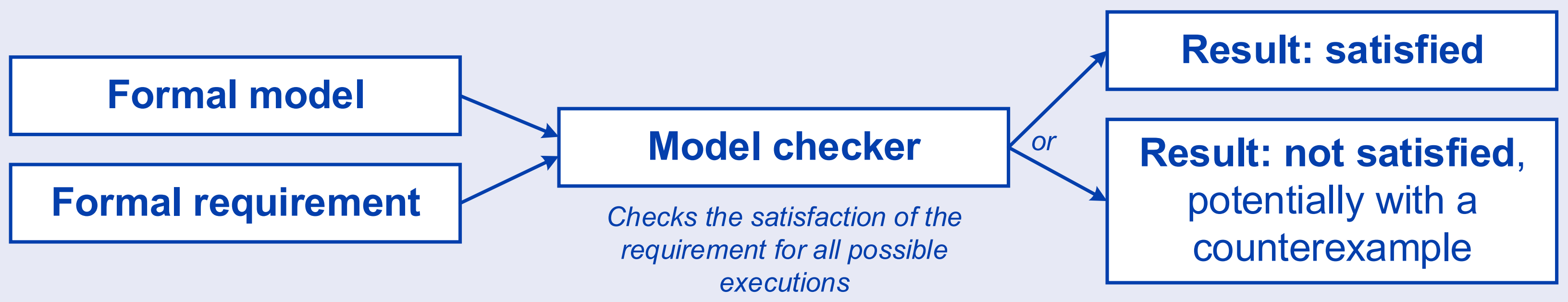


Implementation


WinCC OA
SCADA
HIOC
over S7
S7-400
PLCs
HIOC_SFT
fail-safe CFC function block
+
HIOC_STD
standard SCL function block
Checking signal integrity
Communication with SCADA

Model checking

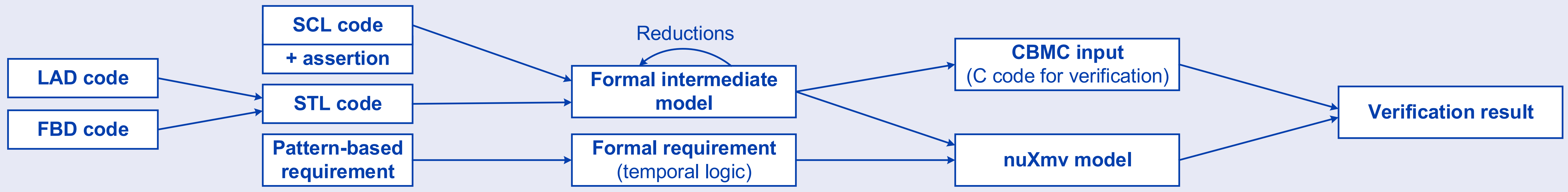
Model checking is a formal verification method that checks the satisfaction of a formal requirement on a formal model with mathematical precision for all possible executions.
In case a violation of a requirement is found, often a counterexample can be provided that shows a trace leading to the violation.


Formal model
Formal requirement
Model checker
Checks the satisfaction of the requirement for all possible executions
or
Result: satisfied
Result: not satisfied, potentially with a counterexample

PLCverif

- **Model checking solution** for PLC programs
- **Hides the formal details** from the users
- Integrates **multiple** model checking **engines**
- Developed at **CERN**

PLCverif s verification workflow


LAD code
FBD code
SCL code
+ assertion
STL code
Pattern-based requirement
Reductions
Formal intermediate model
Formal requirement (temporal logic)
CBMC input (C code for verification)
nuXmv model
Verification result

Pattern-based verification

- **Pattern-based requirement**: Fixed English sentence with place-holders to be filled by the verifier
- **nuXmv**: State-of-the-art symbolic model checker tool

+ **Can check rich, complex requirements**

– **Performance can be a bottleneck**

Assertion-based verification

- **Verification assertion**: Logic expression in the code that must always be satisfied
- **CBMC**: Bounded model checker to check assertion violations in C code

+ **Fast verification**

– **Assertions can only represent simple requirements**

– Bounded model checking ensures correctness only for certain length

Outcome

Formal proof of correctness

- Ongoing work
- Formalising and checking all important requirements is an ongoing work
- Difficult to ensure completely: All tools in the toolchain must be verified

Improved understanding

Via counterexamples

- A counterexample can show a **witness of an incorrect behaviour**
- Similarly, counterexamples can be used to provide examples (traces) of any behaviour
- Such trace may reveal **peculiar, unexpected functionality**

Via requirement formalisation

- Model checking requires formal requirements
- **Removing all ambiguity** from informal specifications is difficult and often reveals **interesting corner cases**
- Needs collaboration of specifiers, developers and verifiers

You can find the paper and more information at
http://cern.ch/plcverif
http://iter.org

Photo of the TOKAMAK: © ITER Organization, http://www.iter.org/, included for informational use.
We thank the ITER interlock team for their support of this work.