

Modellgetriebenes Softwareengineering in der Industrieautomatisierung mit der IEC 61499



Univ.-Prof. Dr. Alois Zoitl
LIT | Cyber-Physical Systems Lab
Johannes Kepler University Linz



**Something is rotten in
the state of automation!**



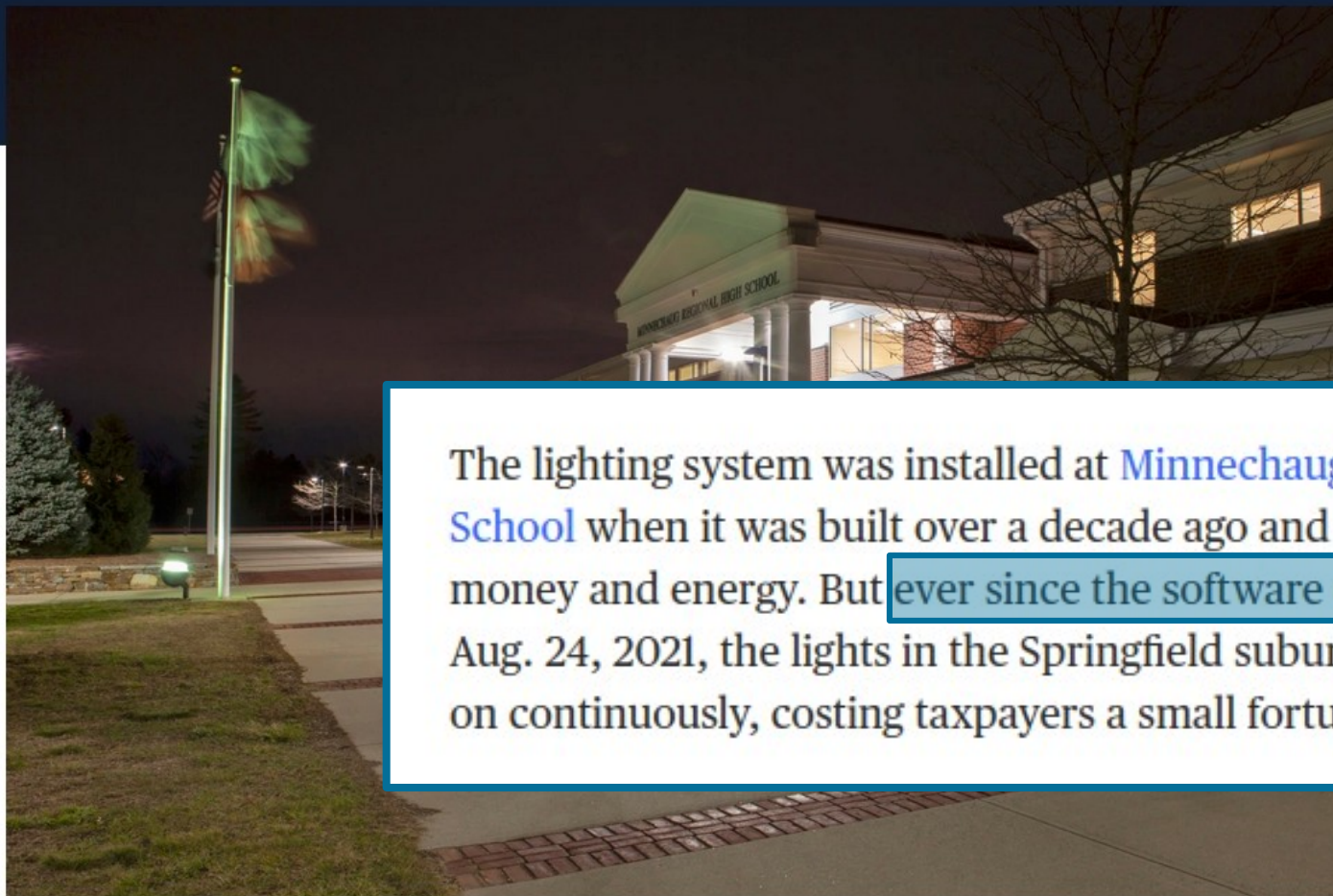
Source: Last Action Hero, Columbia Pictures

EXCLUSIVE

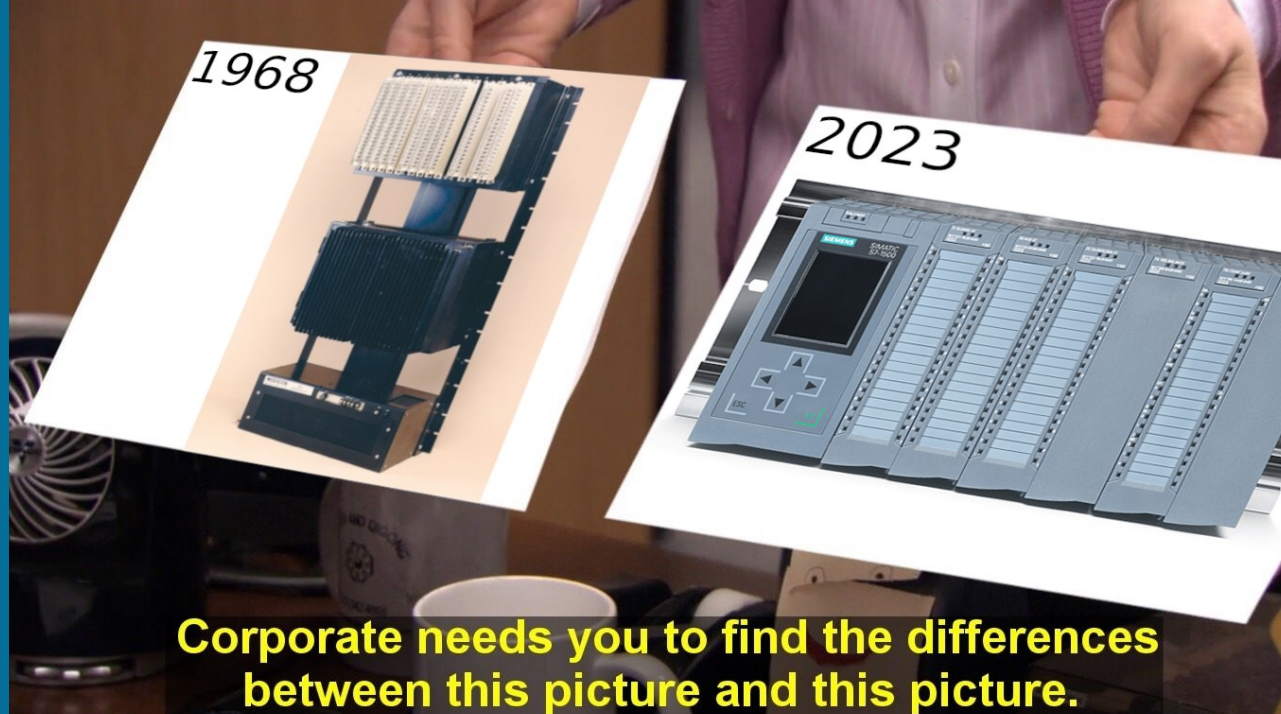
U.S. NEWS

The lights have been on at a Massachusetts school for over a year because no one can turn them off

Blame it on the pandemic and "supply chain problems," says the school district's assistant superintendent of finance.



The lighting system was installed at [Minnechaug Regional High School](#) when it was built over a decade ago and was intended to save money and energy. But **ever since the software that runs it failed** on Aug. 24, 2021, the lights in the Springfield suburbs school have been on continuously, costing taxpayers a small fortune.



Still the same programming model!

➔ Industry 3.0 control software!

They're the same picture.

Problem: Software Development Effort

„... increases the software-engineering portion of the overall Manufacturing costs of a machine: Starting from currently 50% share for electronics and software the share will rise in 2020 up to 80%.“

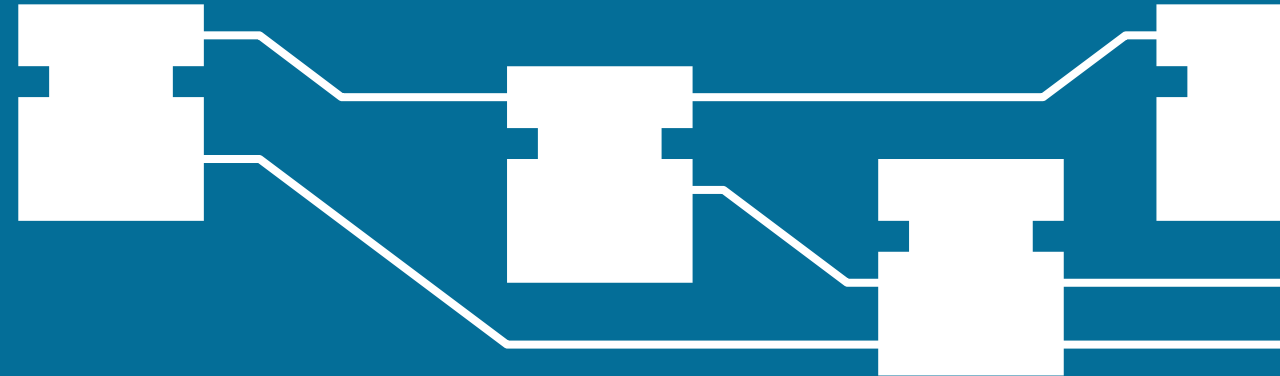
translated from IEE 01-2006

*„We have so far mastered most topics and could **save up to 70%** of the engineering effort. What makes us still problems is the **software effort.**“*

translated from SPS Magazin 08-2012

IEC 61499

Domain-specific Modeling Language for Distributed Industrial Process Measurement and Control Systems



How the UML diagram describes the software



How the code is actually written



Comparing IEC 61131-3 and IEC 61499 with Code Metrics

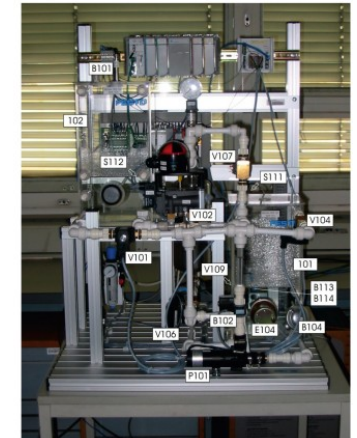
Sequential process

	IEC 61131-3	IEC 61499
Program vocabulary	587	217
Program length	581	439
Estimated length	4839.35	1503.98
Purity ratio	8.33	3.43
Program volume	5343.58	3407.32
Program difficulty	109.50	46.49
Program effort	585122.33	158408.48
Cyclomatic Complexity	33	45 (33)



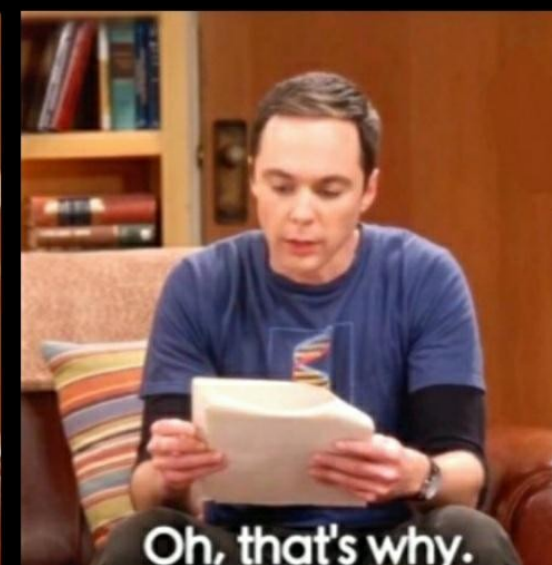
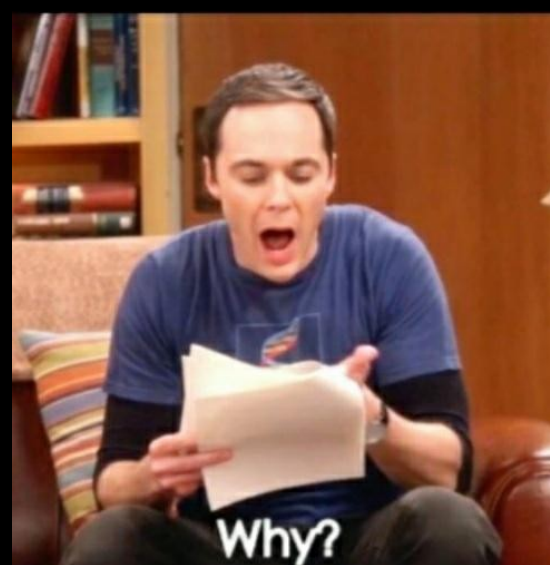
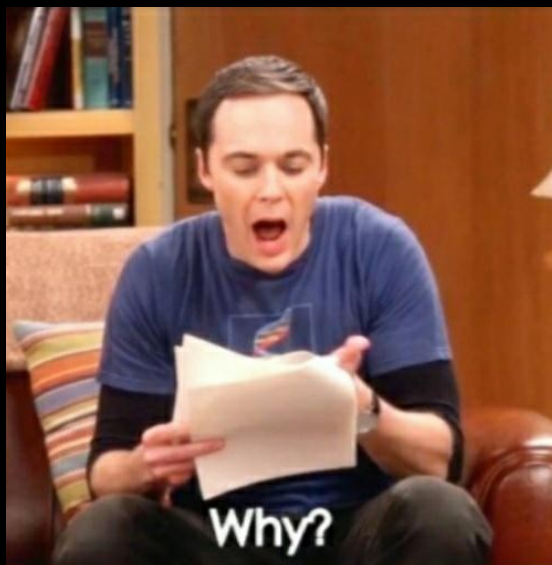
Continuous process

	IEC 61131-3	IEC 61499
Program vocabulary	68	82
Program length	61	117
Estimated length	346.63	455.60
Purity ratio	5.68	3.89
Program volume	371.34	743.83
Program difficulty	15	11.77
Program effort	5570.03	8758.04
Cyclomatic Complexity	3	6 (3)



P. Gsellmann, M. Melik-Merkumians and G. Schitter, "Comparison of Code Measures of IEC 61131–3 and 61499 Standards for Typical Automation Applications," 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Turin, 2018, pp. 1047-1050.

**Stronger Encapsulation!
Events!**



Events in PLC Programs?



We don't do that here!



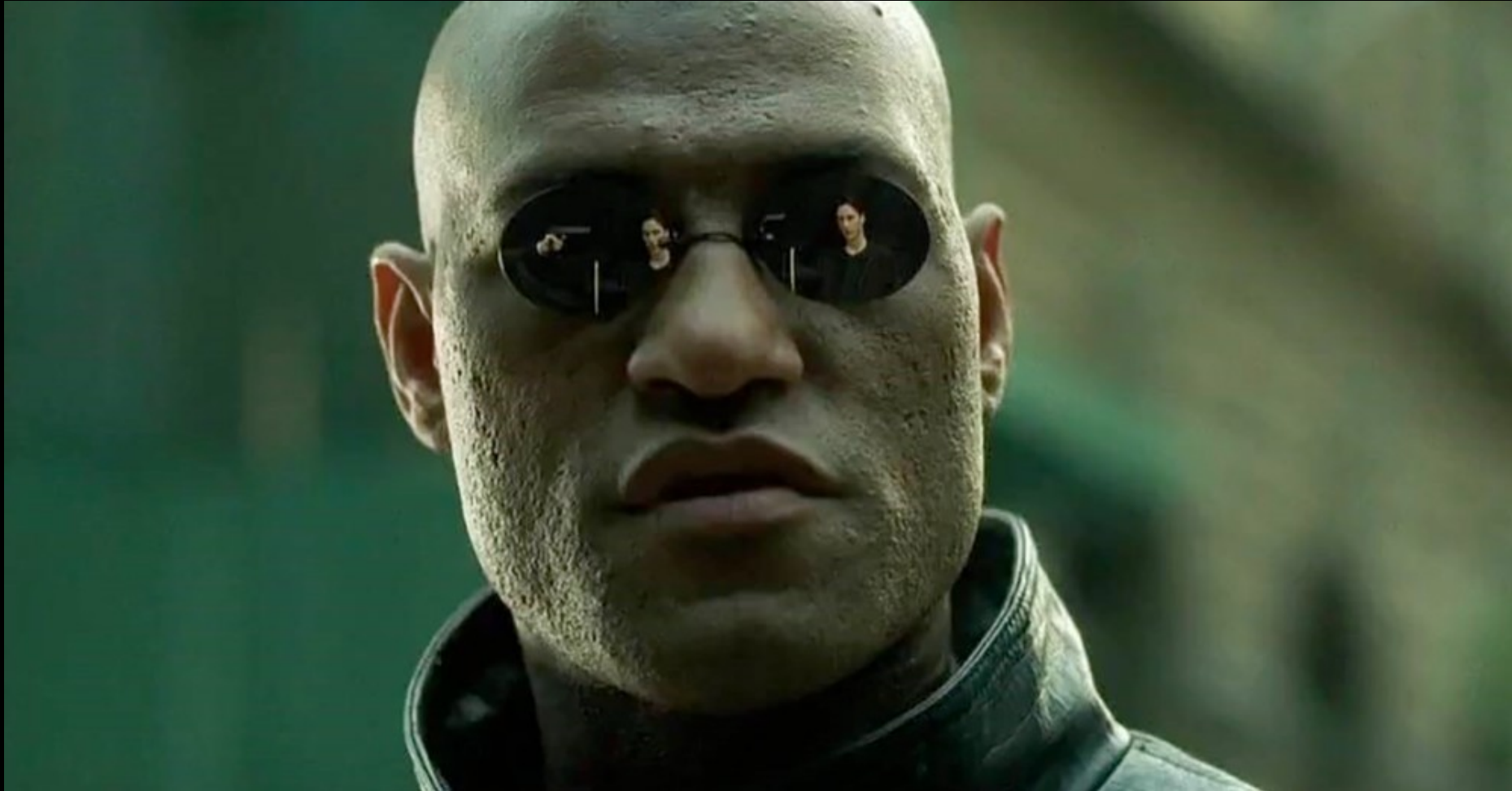
Source: Avengers: Infinity War, Walt Disney Studios Motion Pictures



Oh Really?

Source: Willy Wonka & the Chocolate Factory, Paramount Pictures

What if I told you ...



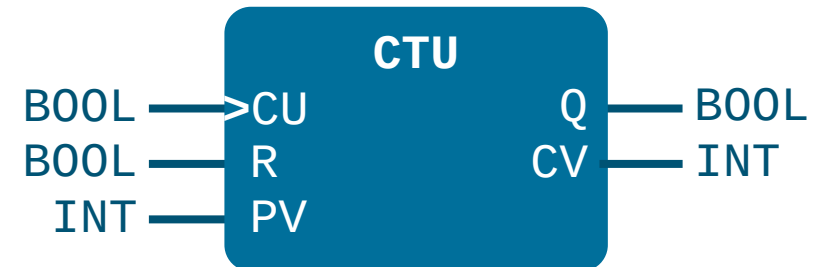
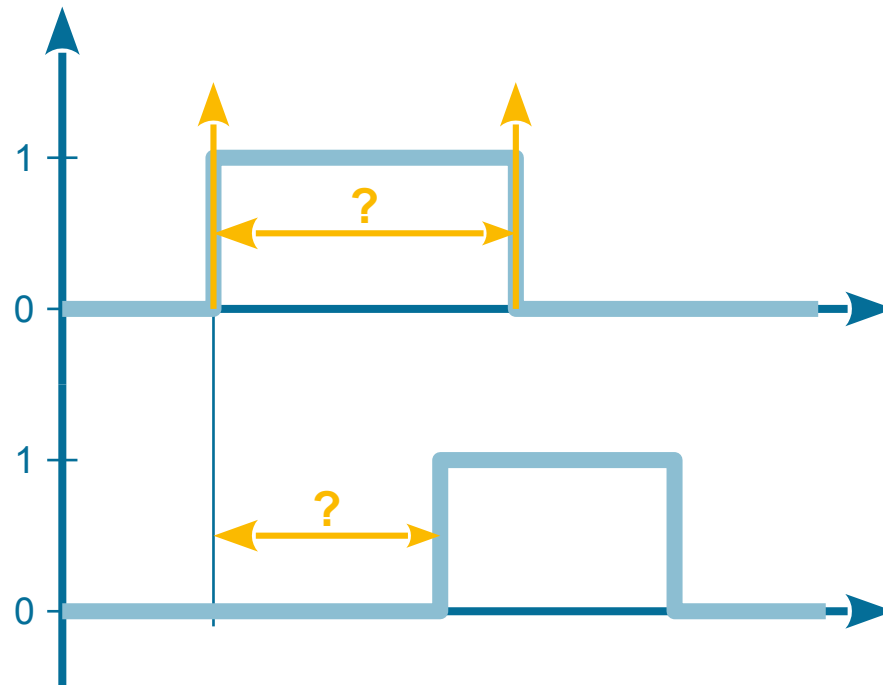
... PLC programs are full of events?

Events in PLC Programming

Variables:

- Init
- Start
- Stop
- Done
- Error
- Send


BOOL



Source: NASA/ESA

What is the Origin of Events in IEC 61499?

IEC 61499 Event Origin: Synchronize Distributed SFCs

 Pergamon

Control Eng. Practice, Vol. 4, No. 6, pp. 855-861, 1996
 Copyright © 1996 Elsevier Science Ltd
 Printed in Great Britain. All rights reserved
 0967-0661/96 \$15.00 + 0.00

PII:S0967-0661(96)00078-0

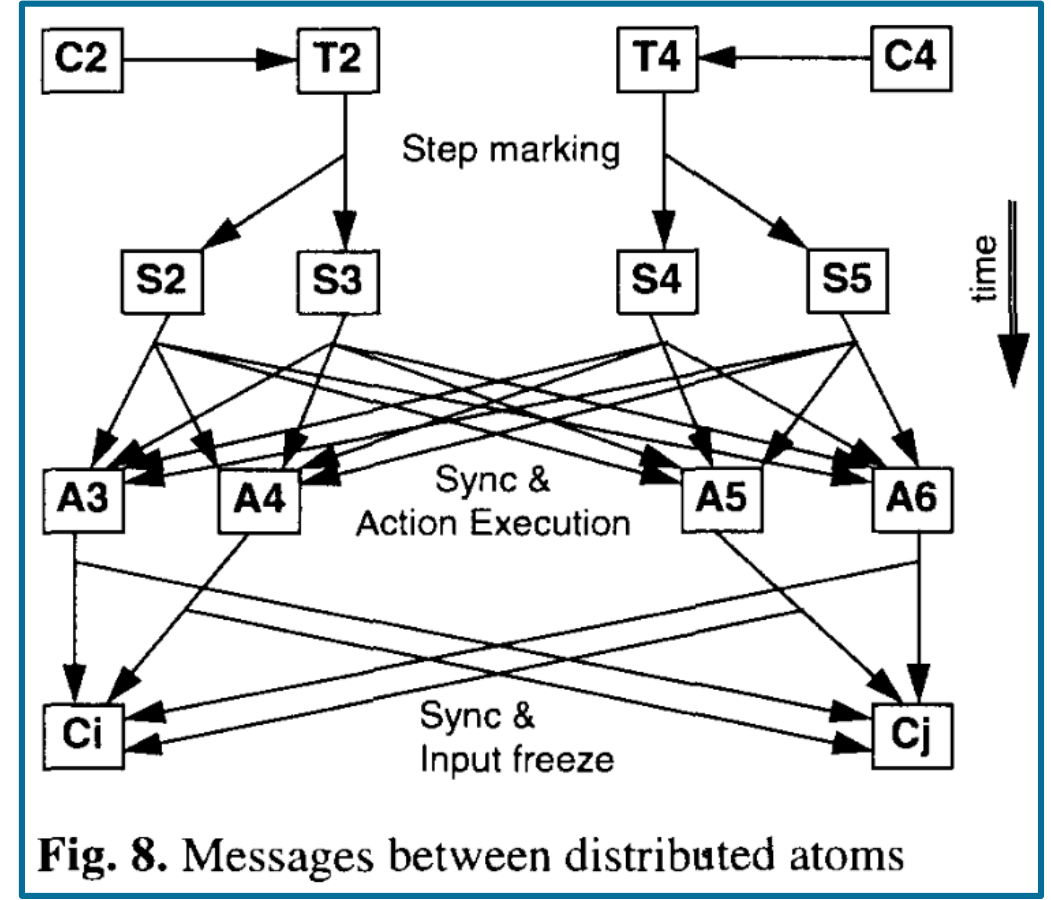
ASYNCHRONOUS AND SYNCHRONOUS APPROACHES FOR PROGRAMMING DISTRIBUTED CONTROL SYSTEMS BASED ON STANDARDS

R. Schoop* and A. Strelzoff**

*AEG Schneider Automation, Steinheimer Strasse 117, 63500 Seligenstadt, Germany (schoop@modicon.de)
 **AEG Schneider Automation, One High Street, North Andover, MA 08145-2699, USA

(Received October 1995; in final form March 1996)

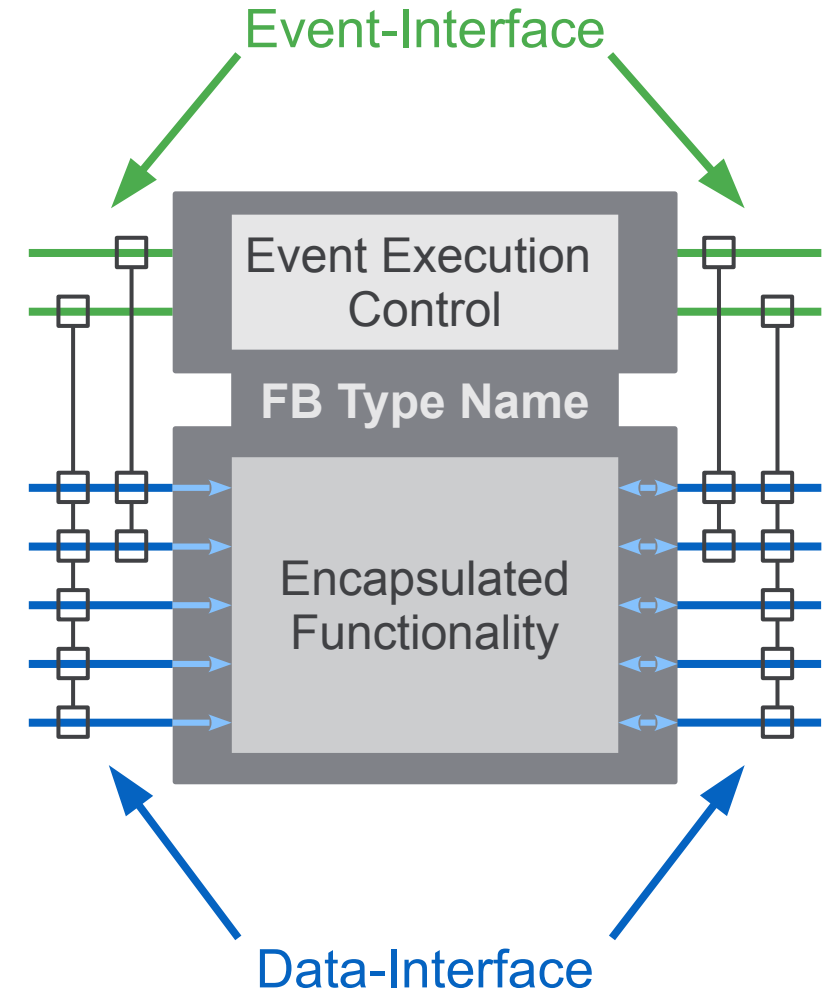
Abstract: Based on a general design model for distributed control systems, and using standardised languages of IEC 1131-3 for control, three approaches to programming are investigated. The first is based on IEC programs with extensions, the second is a decomposition of programs with SFC notations and the third approach uses function blocks corresponding to the IEC TC65 Function Block Standard. The approaches are specified and compared, and conclusions for their use and for further work are drawn. The intention of the contribution is to discuss possibilities for open programming models, rather than to present final results.



R. Schoop, A. Strelzoff Asynchronous and synchronous approaches for programming distributed control systems based on standards, *Control Engineering Practice*, Vol. 4, I. 6, 1996.

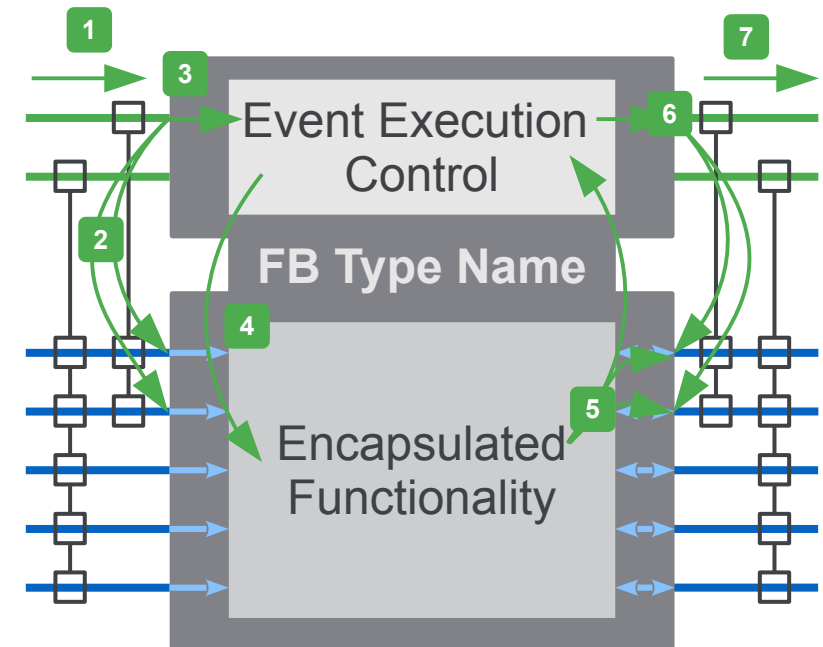
Core Element: Event-driven Function Block

- Function Blocks extended with event interface
- Pure **event-driven** execution model
- Data types based on **IEC 61131-3**
- Focus on **encapsulation** and **reuse**
- No global or directly addressed variables
- Hardware access with special function block type: **Service Interface Function Block**

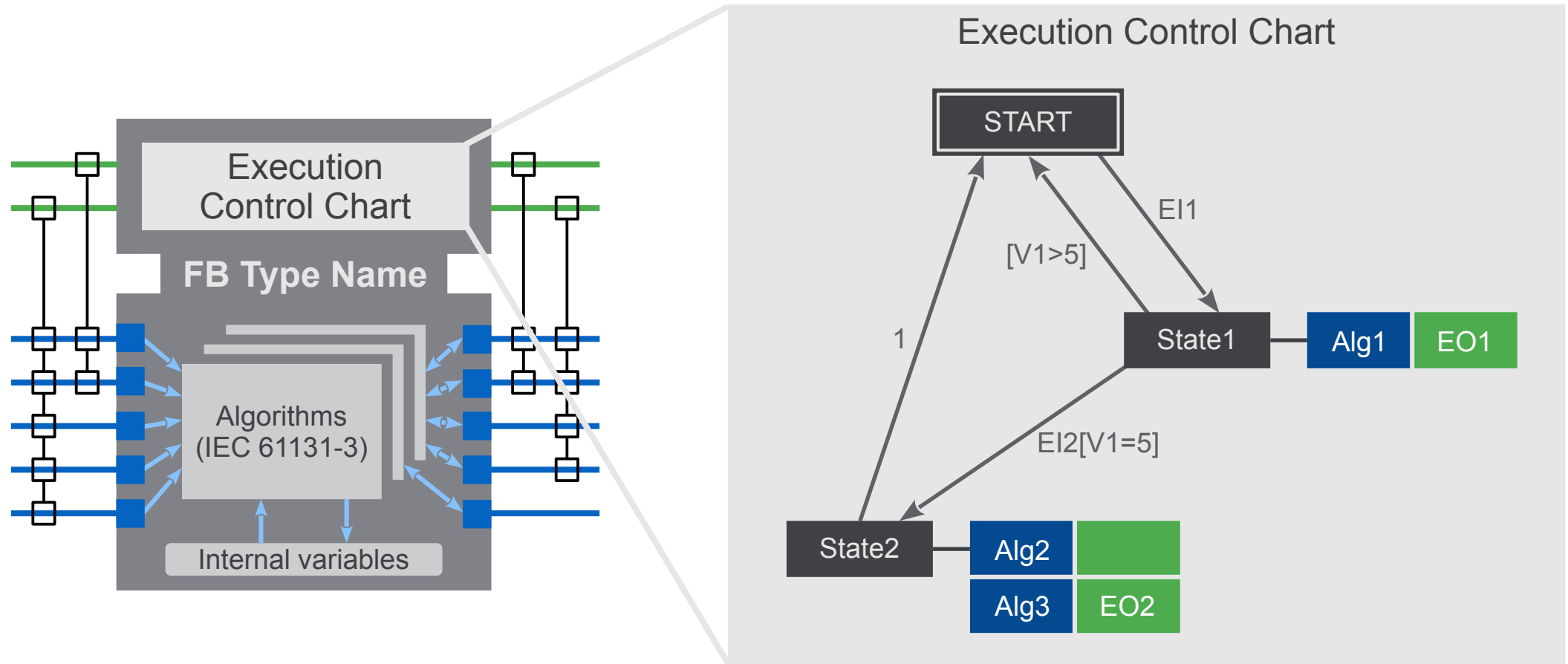


General Function Block Execution Behavior

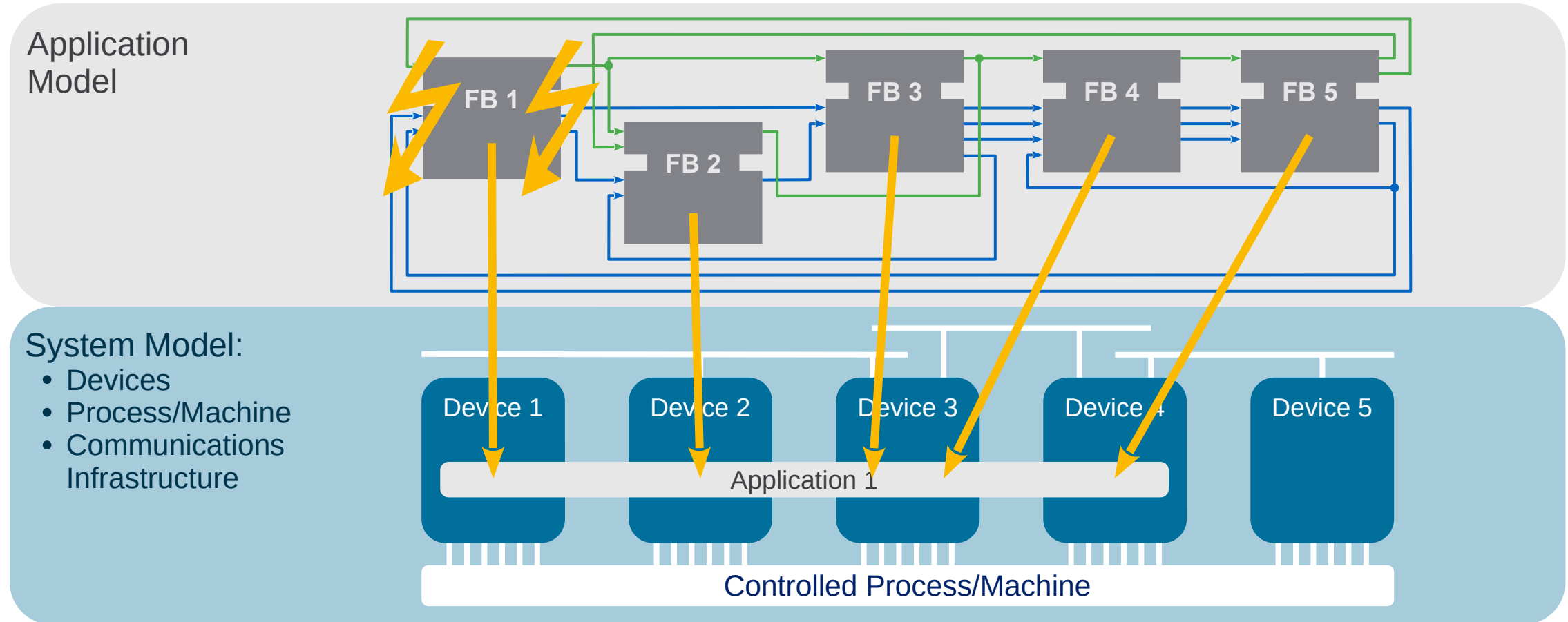
1. Input event gets delivered
2. Associated input data is sampled
3. Event execution control is notified
4. Depending on the type and execution control internal functionality is triggered for execution
5. Internal functionality finishes execution and provides new output data
6. Output event is ready for sending, associated output data is updated
7. Output event is sent
8. Step 4 to 7 may repeat several times



Basic Function Block

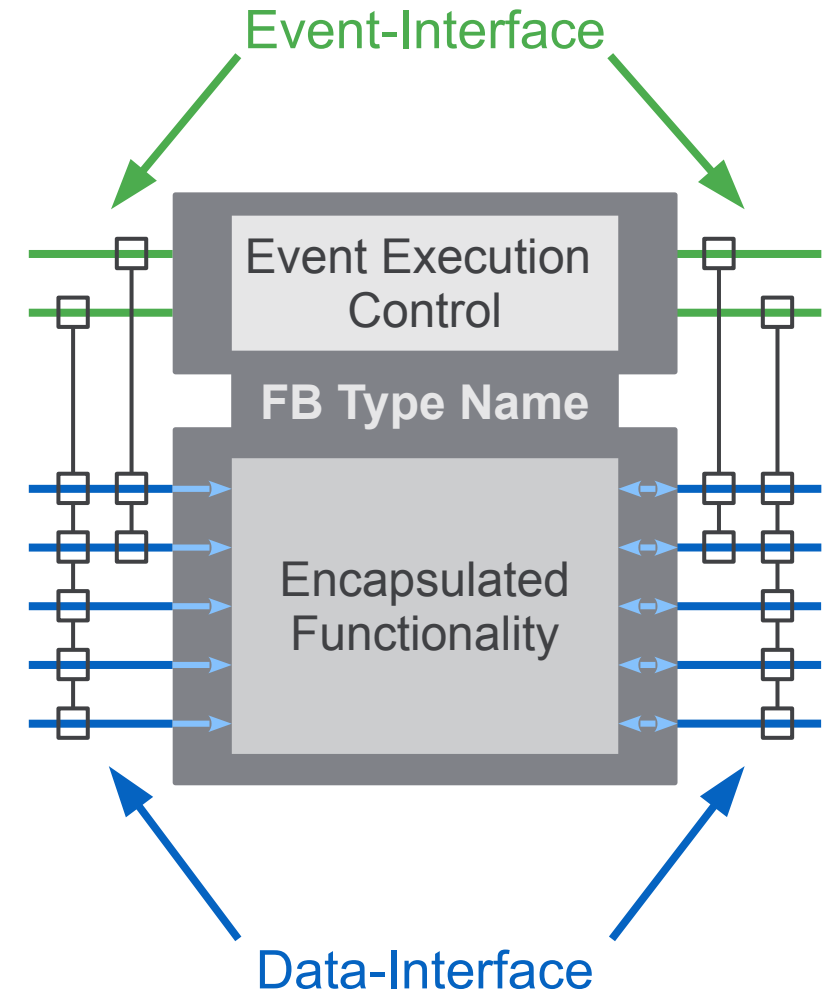


Application, System, and Distribution Model



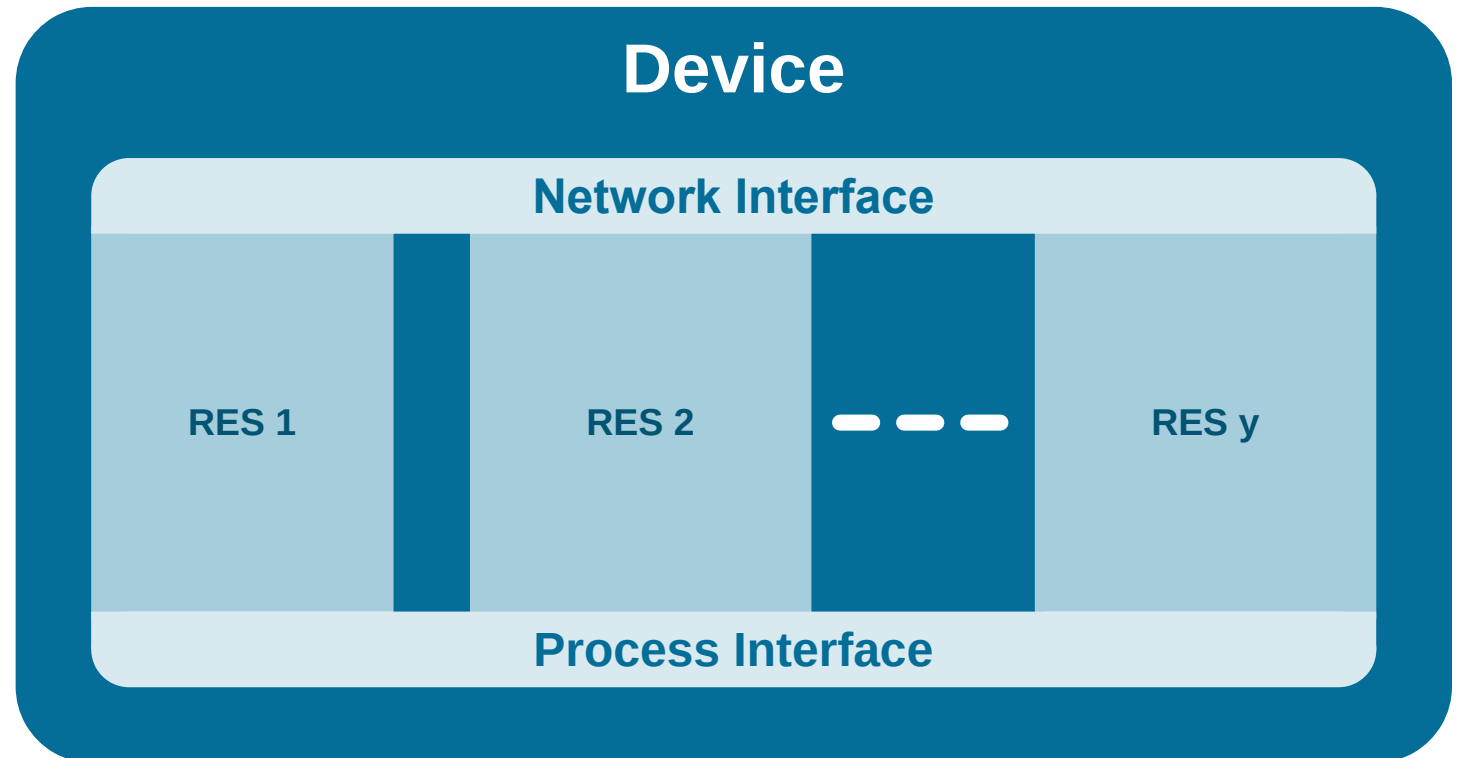
Core Element: Event-driven Function Block

- Function Blocks extended with event interface
- Pure **event-driven** execution model
- Data types based on **IEC 61131-3**
- Focus on **encapsulation** and **reuse**
- No global or directly addressed variables
- Hardware access with special function block type: **Service Interface Function Block**



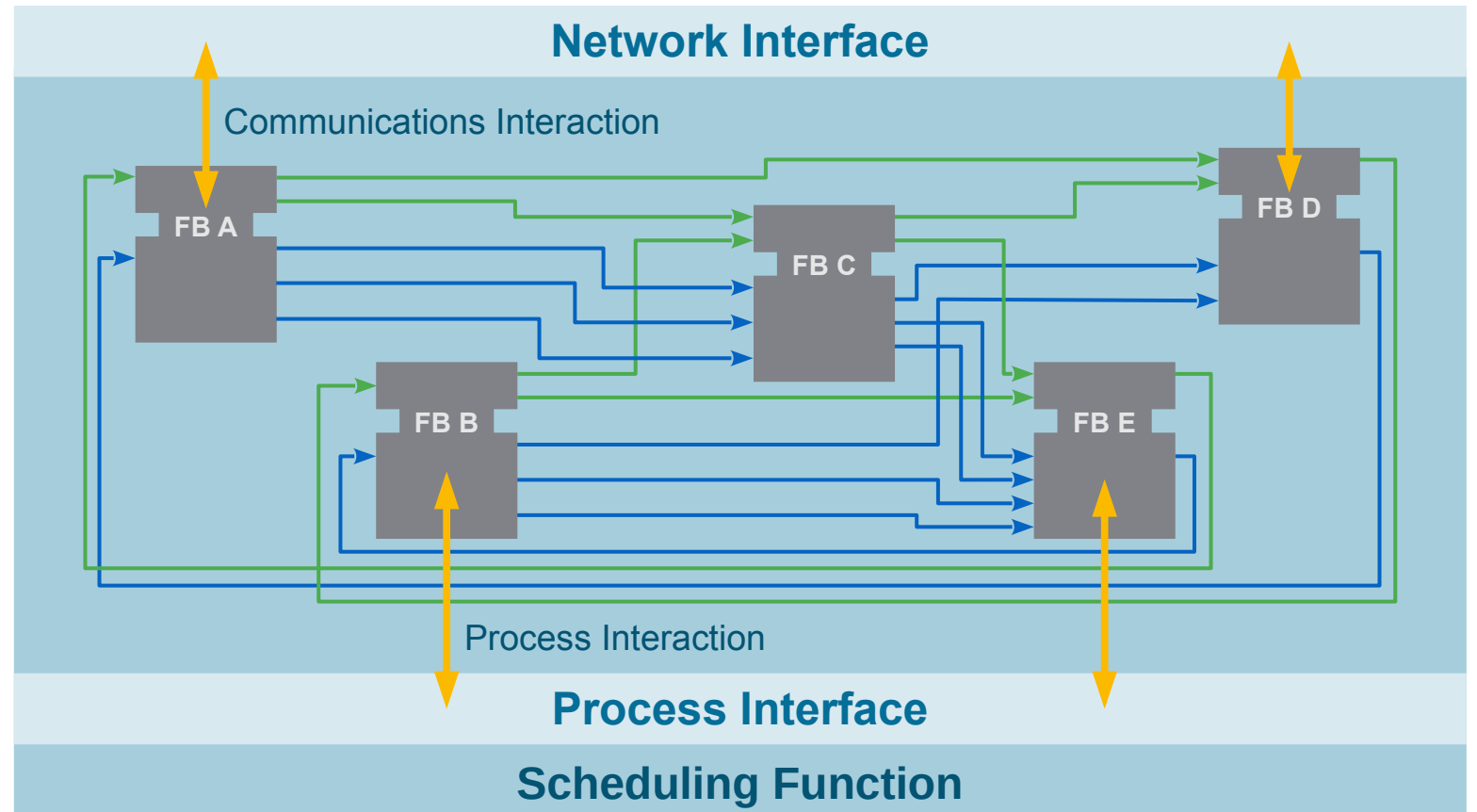
Device Model

- Device:
Container for Resources
- Provides
 - Network interface
 - Process interface



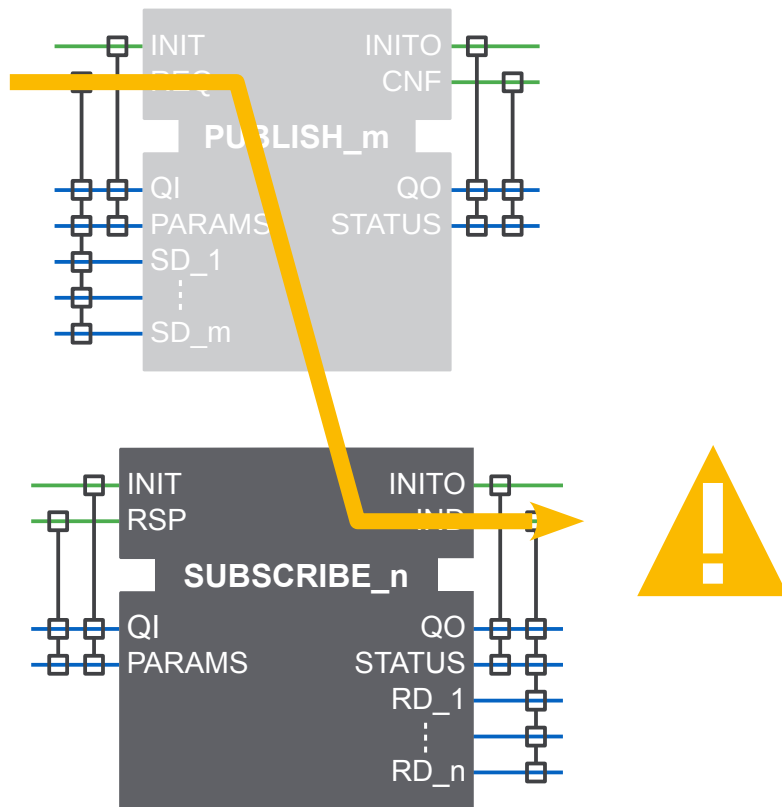
Resource Model

- Resource executes Function Blocks
- Resource provides access to
 - Communication
 - Processfor Service Interface Function Blocks

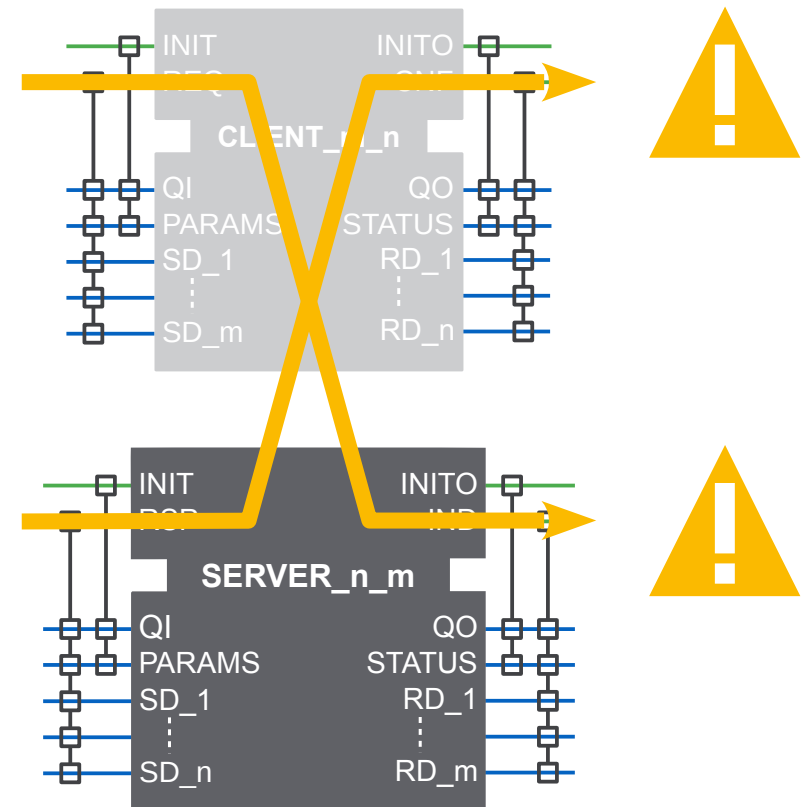


IEC 61499 Communication Patterns

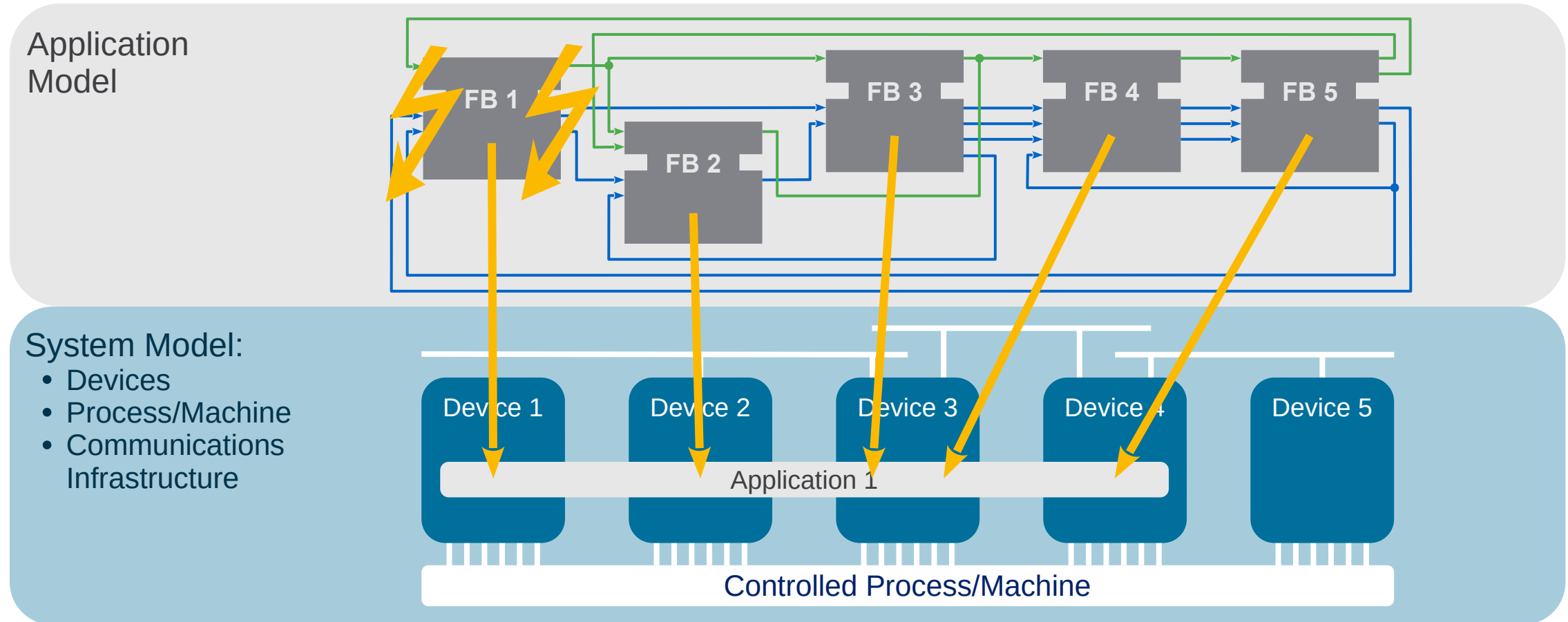
Unidirectional Communication



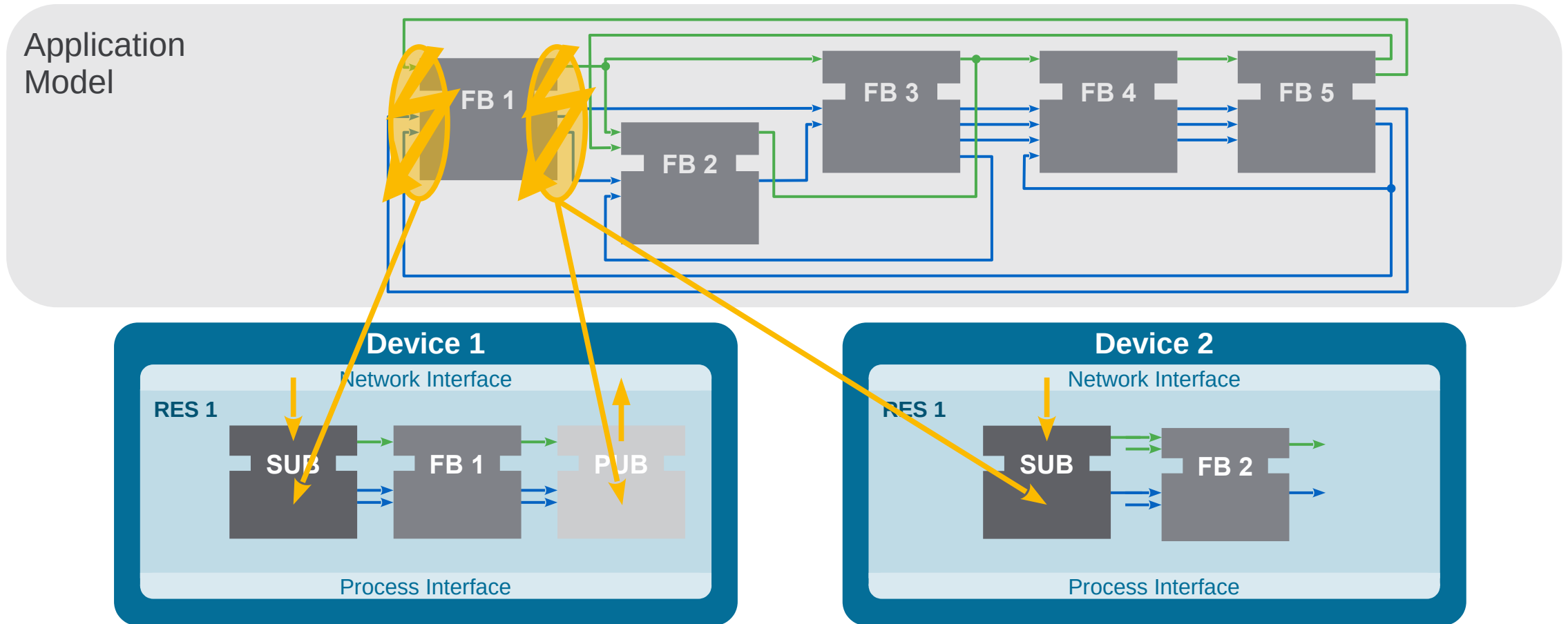
Bidirectional Communication



Application, System, and Distribution Model

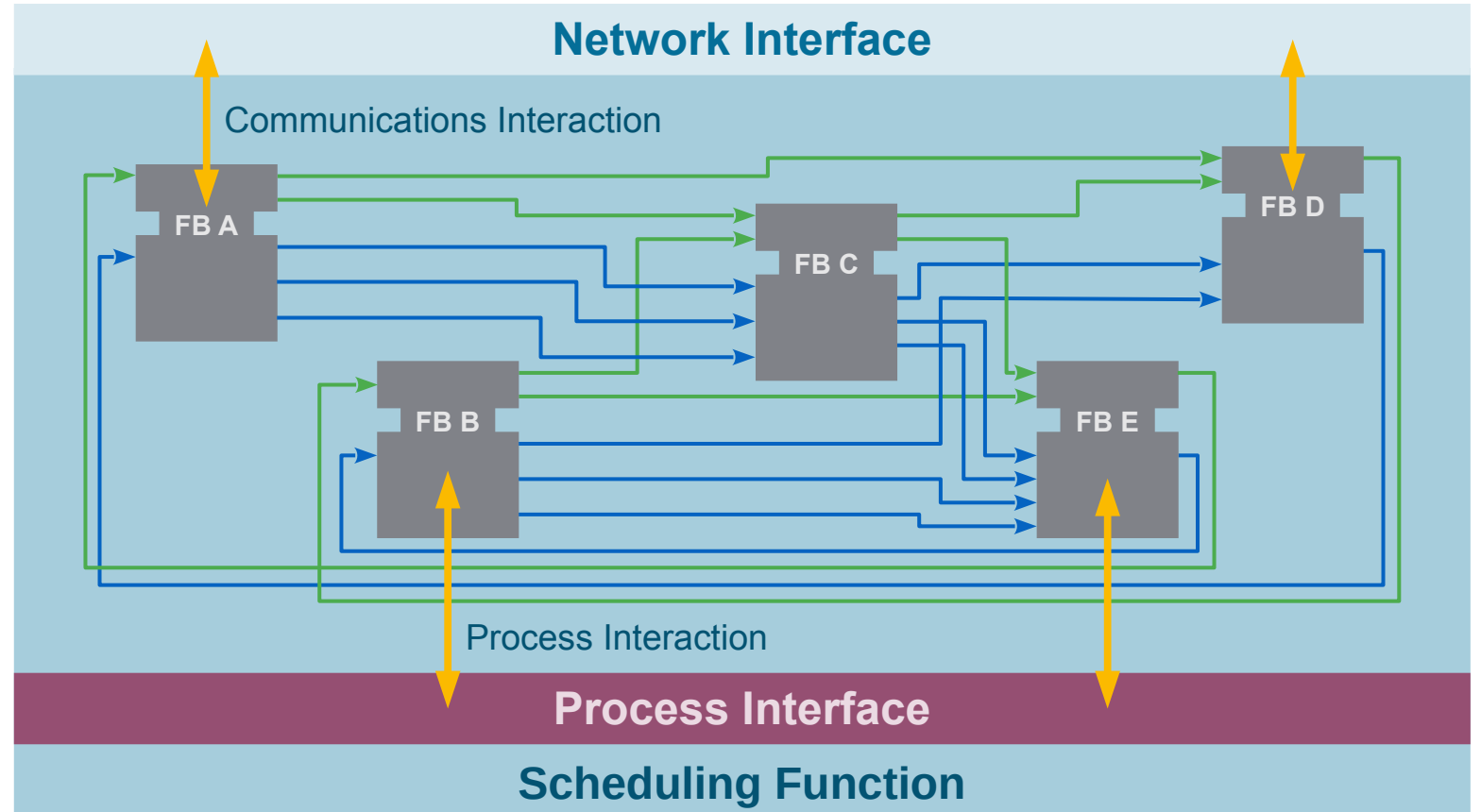


Handling Communication for Distributed Applications

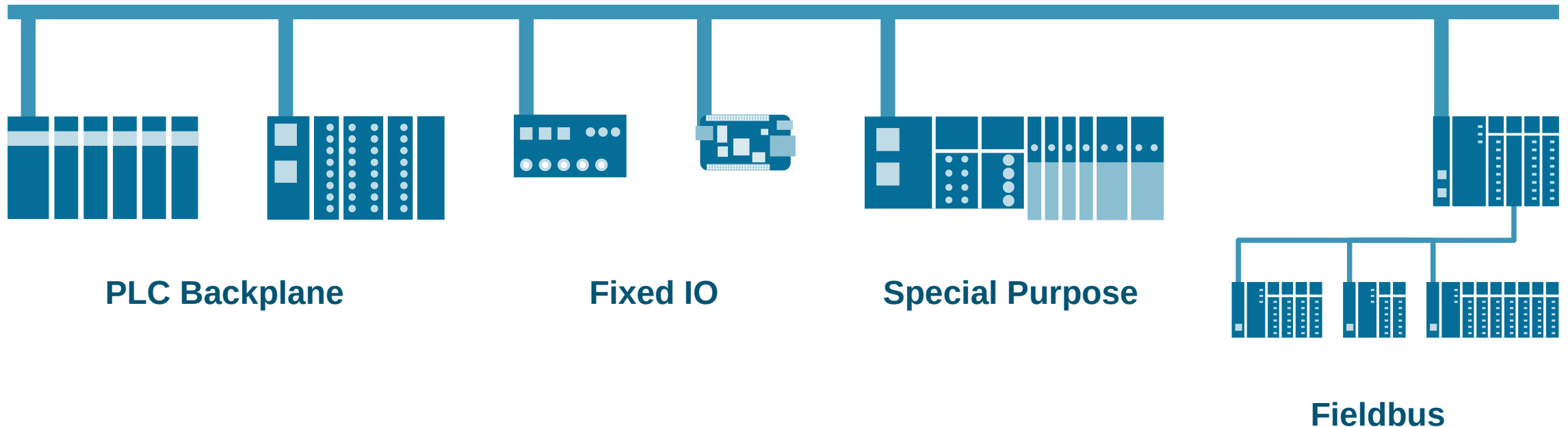


Resource Model

- Resource executes Function Blocks
- Resource provides access to
 - Communication
 - Processfor Service Interface Function Blocks

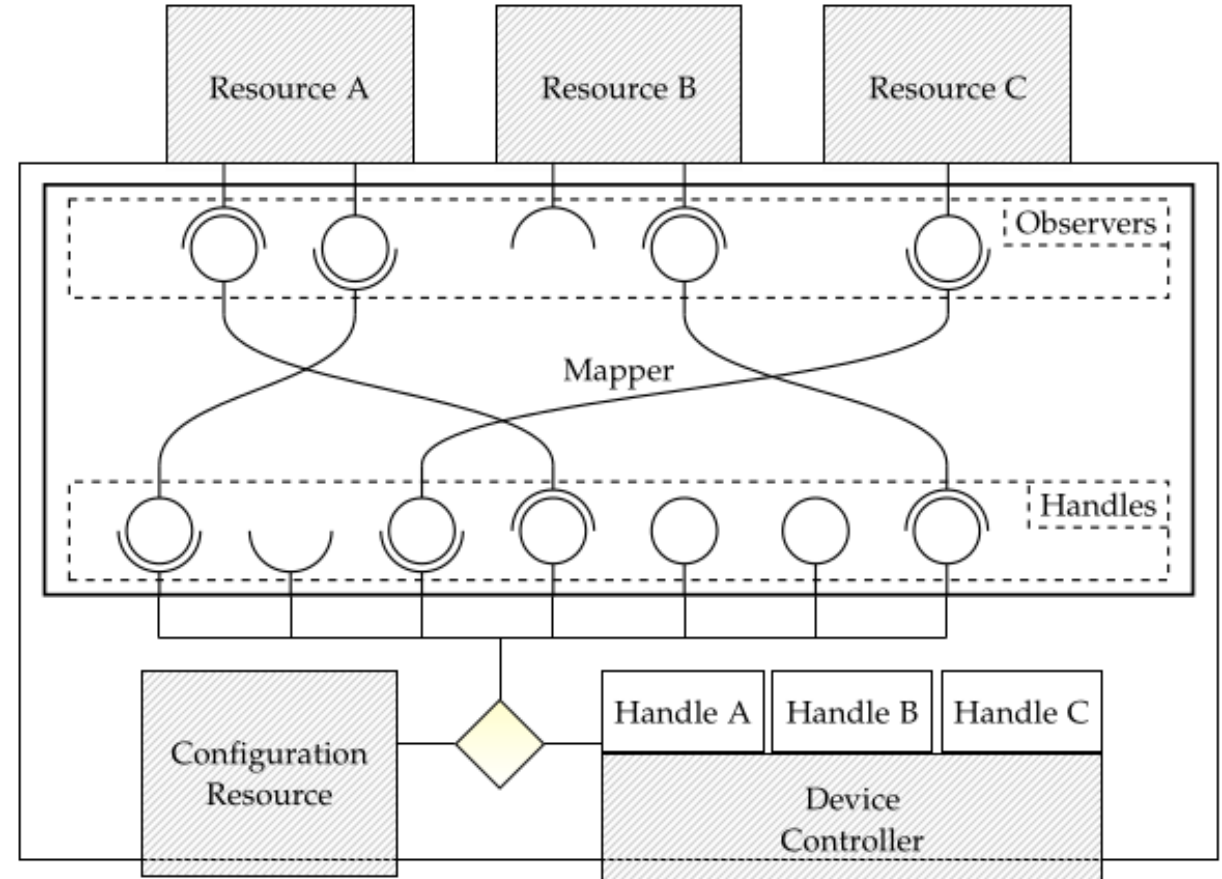
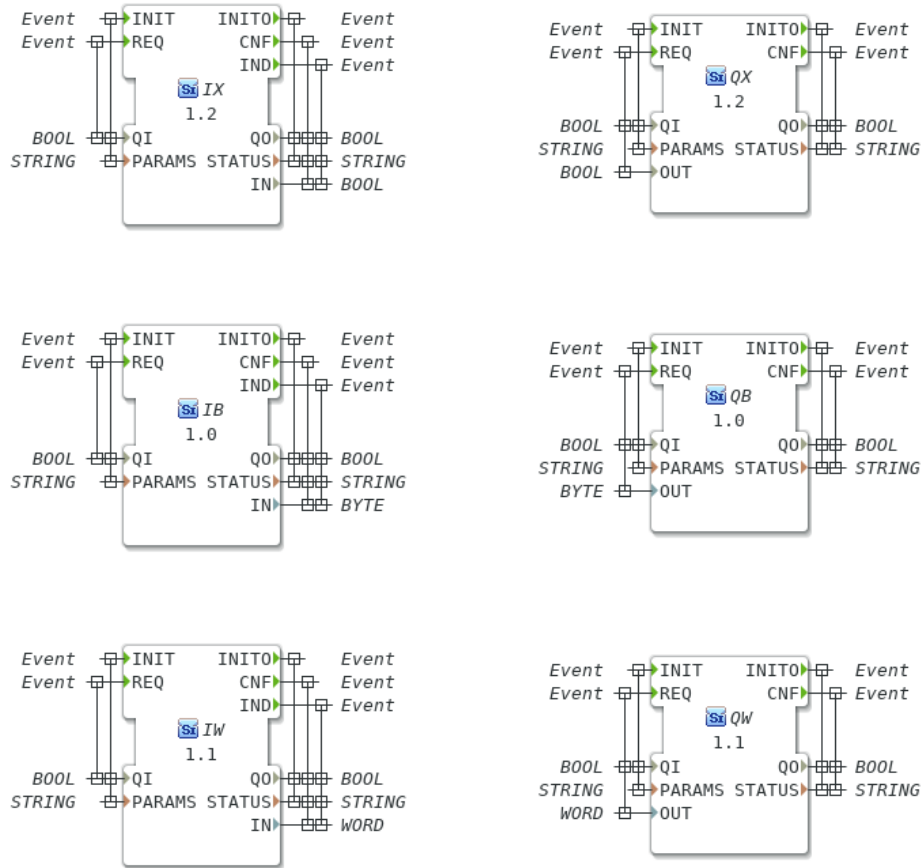


Process Interface: Where Events Meet the Scanned World



W. Eisenmenger, J. Meßmer, M. Wenger and A. Zoitl, "Increasing control application reusability through generic device configuration model," 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 2017

Generic Process Interface



J. Meßmer, "Handling a modular IO System in embedded IEC 61499 controllers", BA Thesis, TU München, Mar 2017

Research Topic: Automatically Derive Events from Process Values

The CIP Method: Component- and Model-Based Construction of Embedded Systems

Hugo Fierz

Computer Engineering and Networks Laboratory TIK
Swiss Federal Institute of Technology ETH
CH-8092 Zürich, Switzerland
fierz@tik.ee.ethz.ch

Abstract. CIP is a model-based software development method for embedded systems. The problem of constructing an embedded system is decomposed into a *functional* and a *connection* problem. The *functional* problem is solved by constructing a formal reactive behavioural model. A CIP model consists of concurrent clusters of synchronously cooperating extended state machines. The state machines of a cluster interact by multicast events. State machines of different clusters can communicate through asynchronous channels. The construction of CIP models is supported by the CIP Tool, a graphical modelling framework with code generators that transform CIP models into concurrently executable CIP components. The *connection* problem consists of connecting generated CIP components to the real environment. This problem is solved by means of techniques and tools adapted to the technology of the interface devices. Construction of a CIP model starts from the behaviour of the processes of the real environment, leading to an operational specification of the system

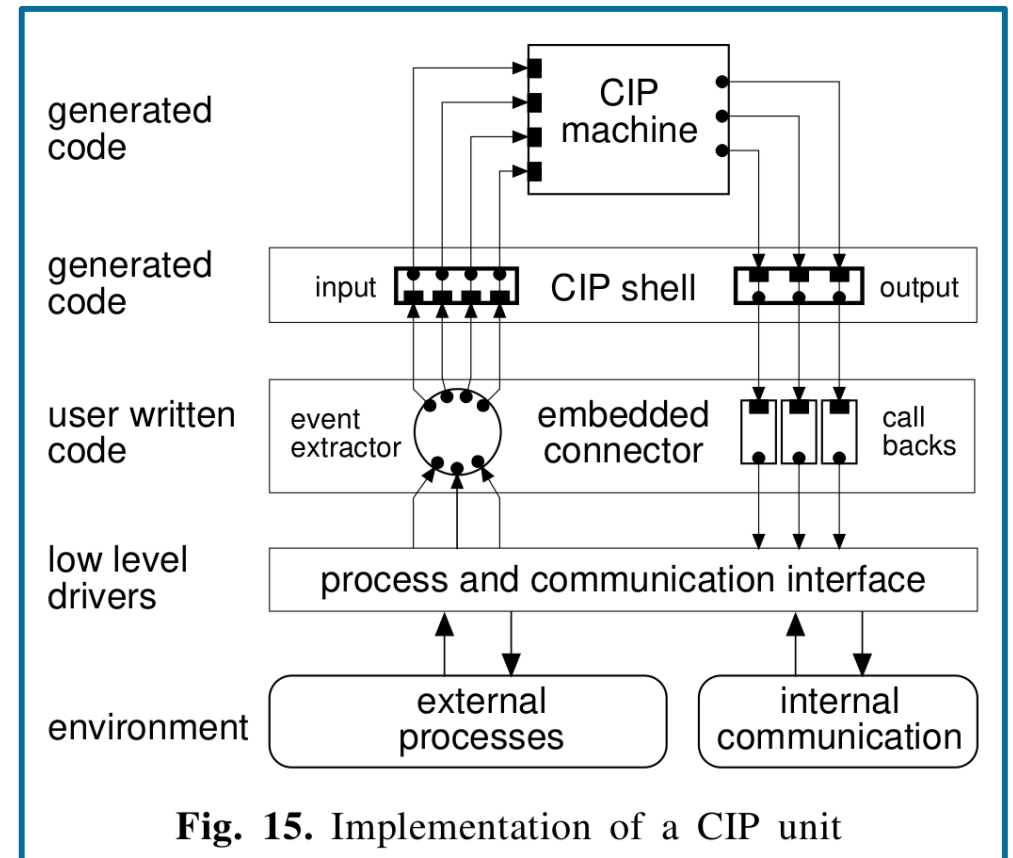


Fig. 15. Implementation of a CIP unit

Fierz, H. (1999). The CIP Method: Component- and Model-Based Construction of Embedded Systems. In: Nierstrasz, O., Lemoine, M. (eds) Software Engineering — ESEC/FSE '99. ESEC SIGSOFT FSE 1999 1999. Lecture Notes in Computer Science, vol 1687. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48166-4_23



Ali Spittel  **#CodeLand**

@ASpittel



Why would you ever spend a few minutes reading the documentation when you can spend a few hours randomly trying things?

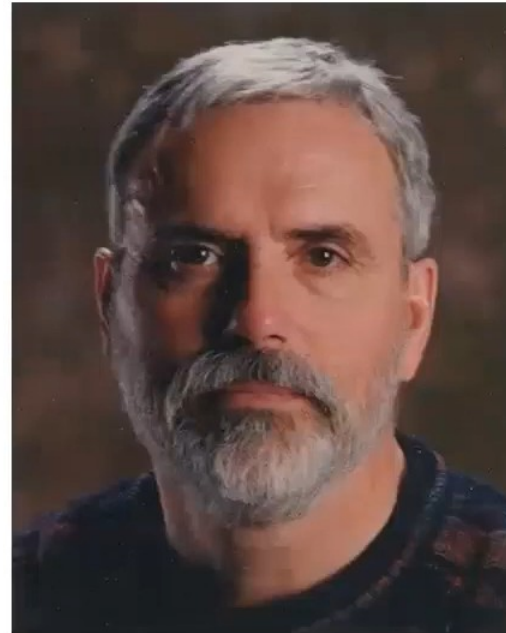
10:55 AM · 23 Jul 19 · [TweetDeck](#)

206 Retweets **1,217** Likes



Keynote

What's Wrong with "Users"?



Bran Selic
Malina Software Corp., Canada

A large red circle with a white background inside, containing text in red. The text is centered and reads: "A USER INTERFACE IS LIKE A JOKE. IF YOU HAVE TO EXPLAIN IT, IT'S NOT THAT GOOD."

A USER
INTERFACE
IS LIKE A JOKE.
IF YOU HAVE TO
EXPLAIN IT, IT'S
NOT THAT
GOOD.

Challenge: Languages Like IEC 61499 Require Highly Sophisticated Tools

Inputs	Outputs			
Name	Type	Name	Type	Comment
1 cmd	AConveyor	1 motor	AMotor	
		2 inletBlocker	ABlocker	
		3 mainBlocker	ABlocker	
		4 index	ACylinder	

```
ALGORITHM REQ
IF NOT init OR RST THEN
  init := BOOL#TRUE;
  in_last := in;
  t_last := ULINT_TO_UDINT(TIME_IN_US_TO_ULINT(NOW_MONOTONIC()));
  i := REAL#0.0;
  tc := REAL#0.0;
ELSE
  (* read last cycle time in Microseconds *)
  tx := ULINT_TO_UDINT(TIME_IN_US_TO_ULINT(NOW_MONOTONIC()));
  tc := UDINT_TO_REAL(tx - t_last);
  t_last := tx;

  (* calculate proportional part *)
  p := KP * IN;

  (* run integrator *)
  i := (IN + in_last) * REAL#5.0E-7 * KI * tc + i;
  in_last := IN;

  (* calculate output Y *)
  Y := d + i;
```

Name	Value
THIS	MySuperSimple
IN	20
Adding	28600
Subtracting	-28600
INTERNALVAR1	FALSE
INTERNALVAR2	FALSE
INTERNALVAR3	FALSE
INTERNALVAR4	FALSE
INTERNALVAR5	FALSE
INTERNALVAR6	FALSE
INTERNALVAR7	FALSE
INTERNALVAR8	FALSE
INTERNALVAR9	FALSE

Thank you!



LIT | Cyber-Physical Systems Lab
Johannes Kepler University Linz

IEC 61499, a system so fine,
Controls processes, bringing efficiency in line.
A network built of functions divine,
It offers us control, we need it all the time.

Networks of functions, a beautiful view,
Flowing together, with seamless control too.
They interact, to bring forth what is true,
And with IEC 61499, everything comes through.

In factories and plants, near and far,
Its presence is felt, a shining star.
Safe and reliable, never any lag,
It's the backbone of our era, a timeless piece of art.

Real-time control, is what we desire,
With IEC 61499, our dreams won't tire.
Functions always up-to-date, forever anew,
In a world that's constantly changing fast, that's true.

So let us celebrate, IEC 61499 so pure,
Our system that helps us, the future secure.
With efficiency, control and bravery in tow,
Together, let us reach our goal, in real-time, and grow.

ChatGPT in the style of Goethe, instructed by Bianca Wiesmayr, 2023