

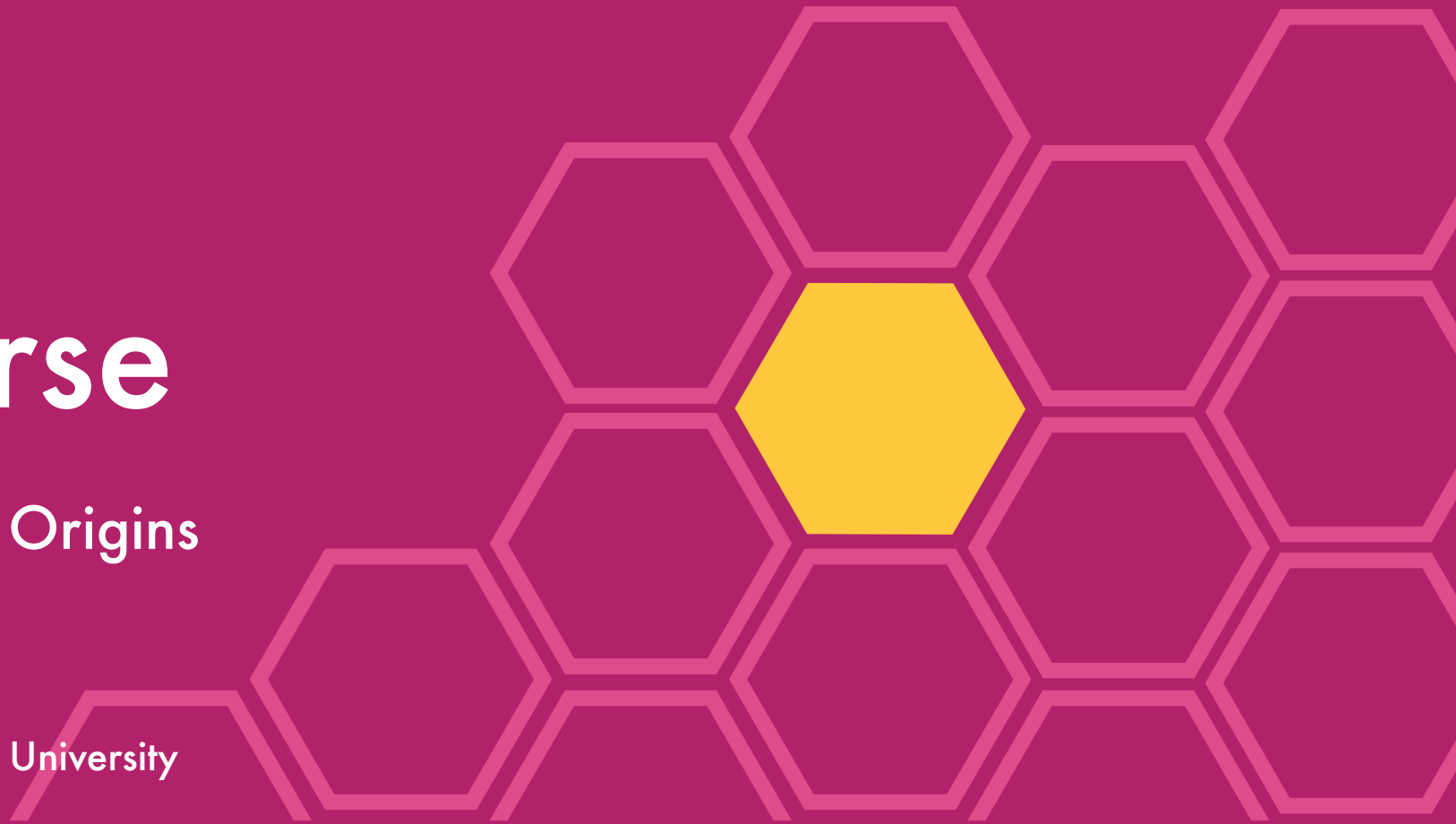
UNIVERSAL
AUTOMATION.ORG

IEC 61499: primer course

Module 0: Motivations and Origins

Valeriy Vyatkin

Luleå University of Technology and Aalto University



What is IEC 61499?

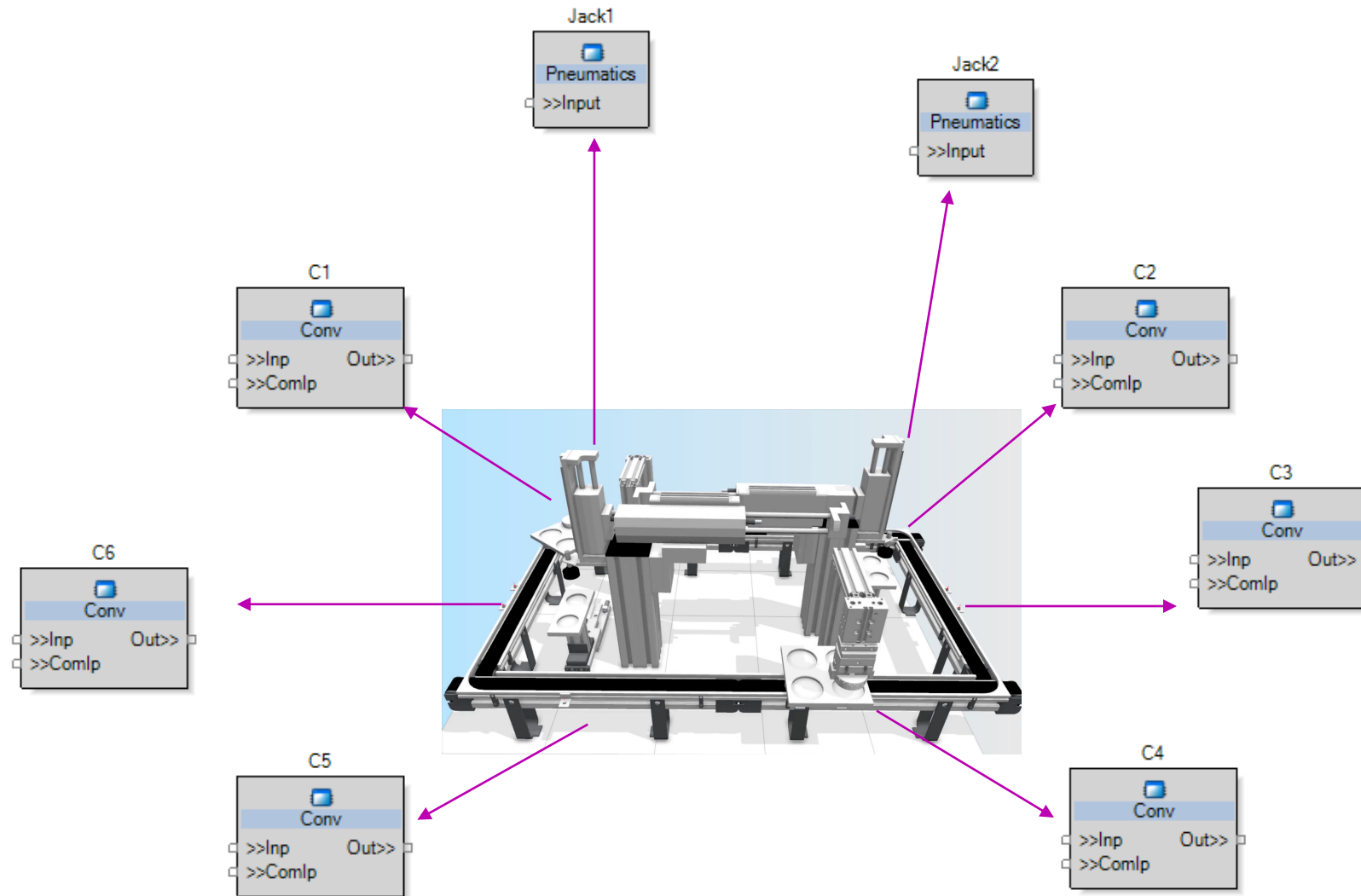
Example: Assembly system model

- Mechatronic modularity
- Layout changes during the production process to flexibly accommodate new order
- Wireless communication
- Totally distributed hardware control architecture



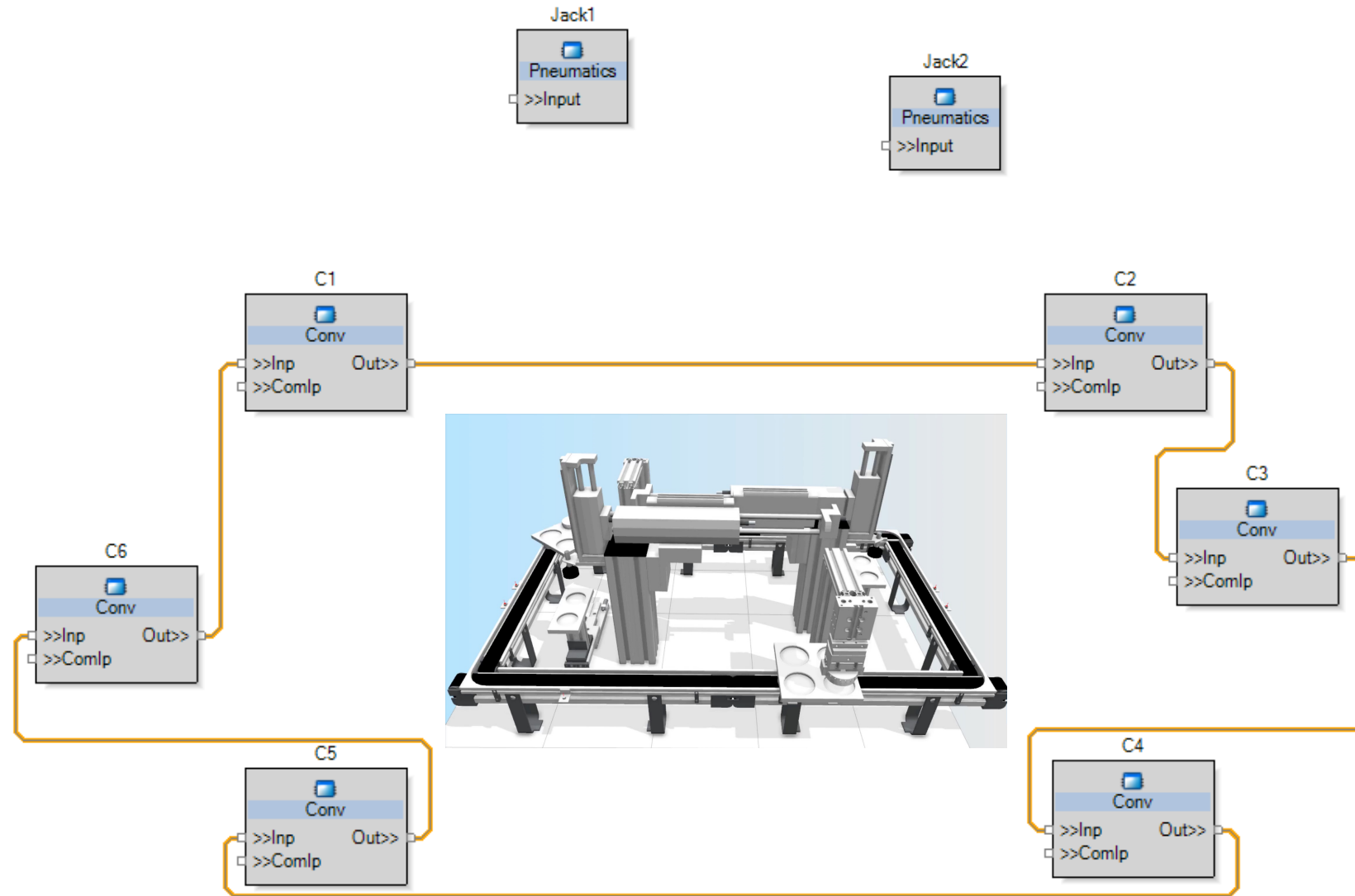
Object-based design

Each function block type corresponds to a mechatronic component type.

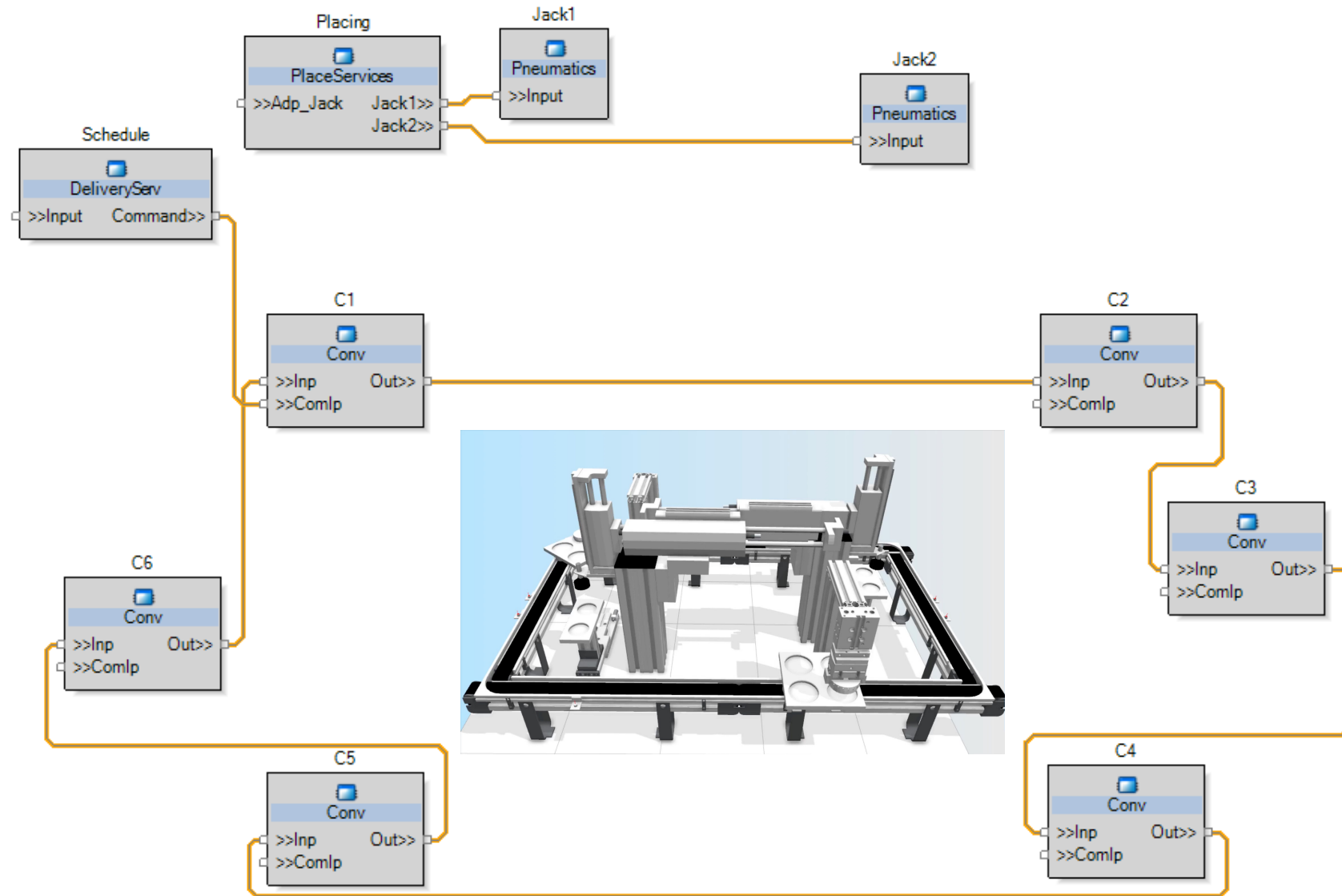


Each function block type implements basic control services for the mechatronic component.

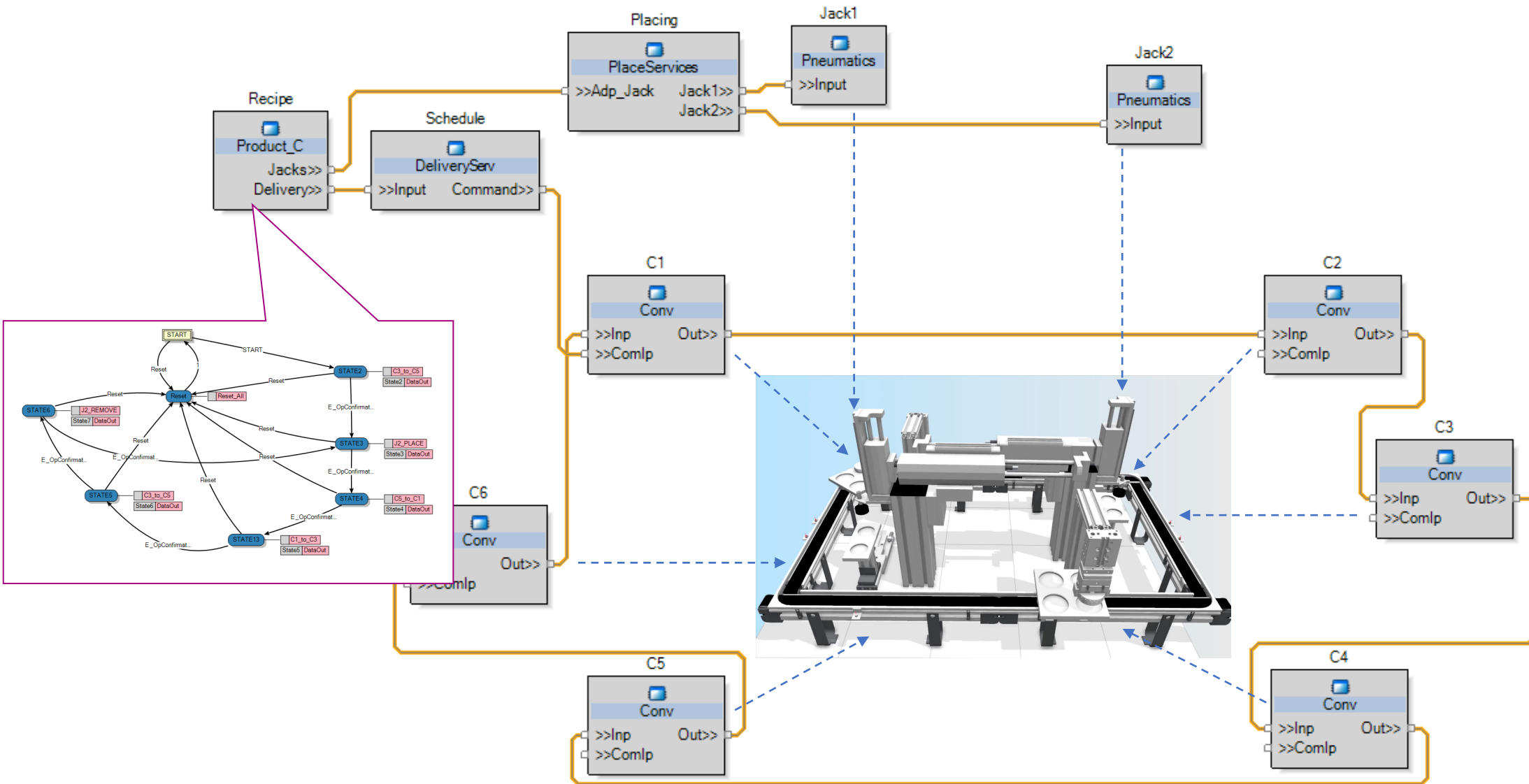
Programming with Function Blocks



Programming with Function Blocks



Programming with Function Blocks

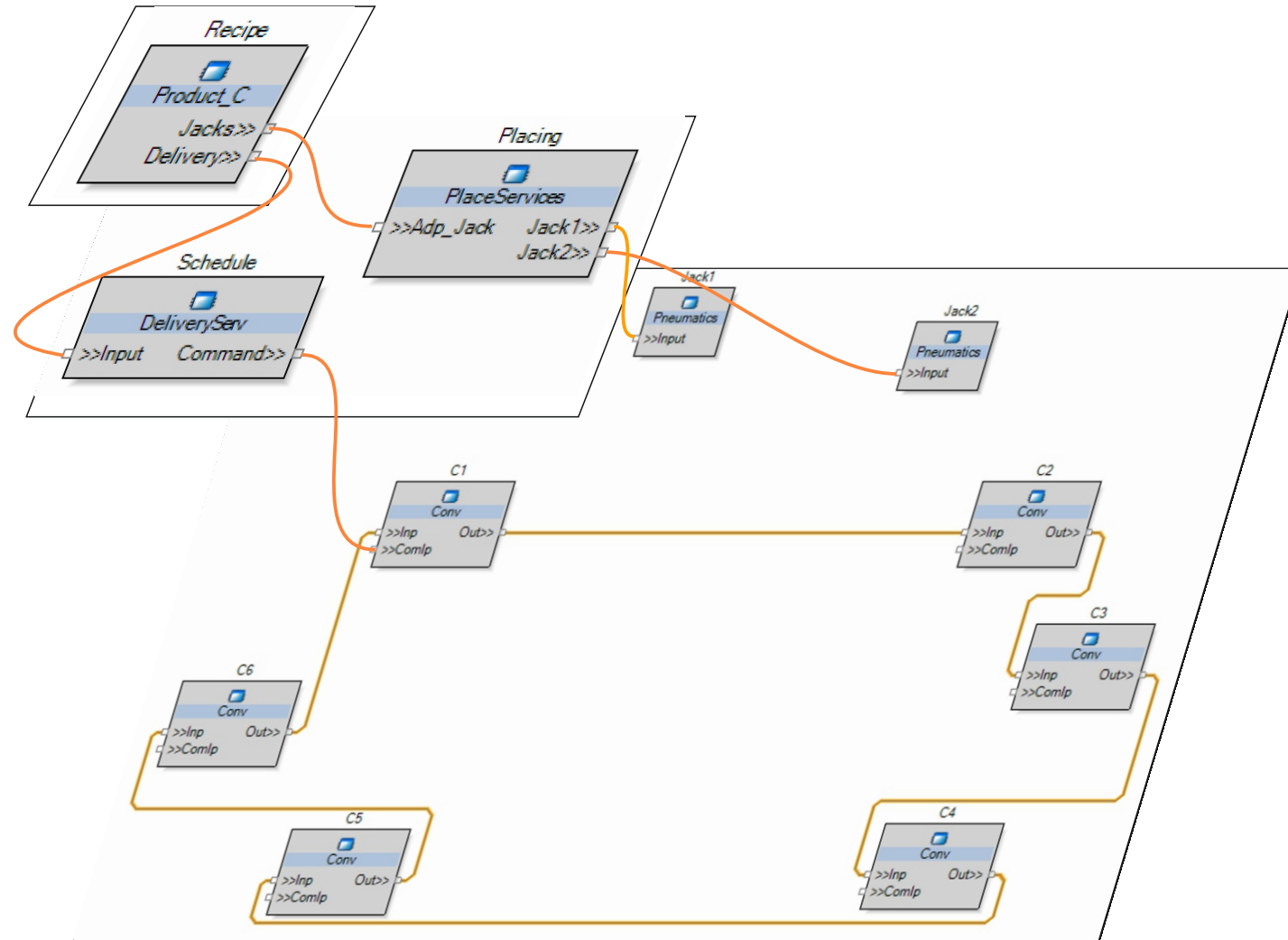


Layered services architecture

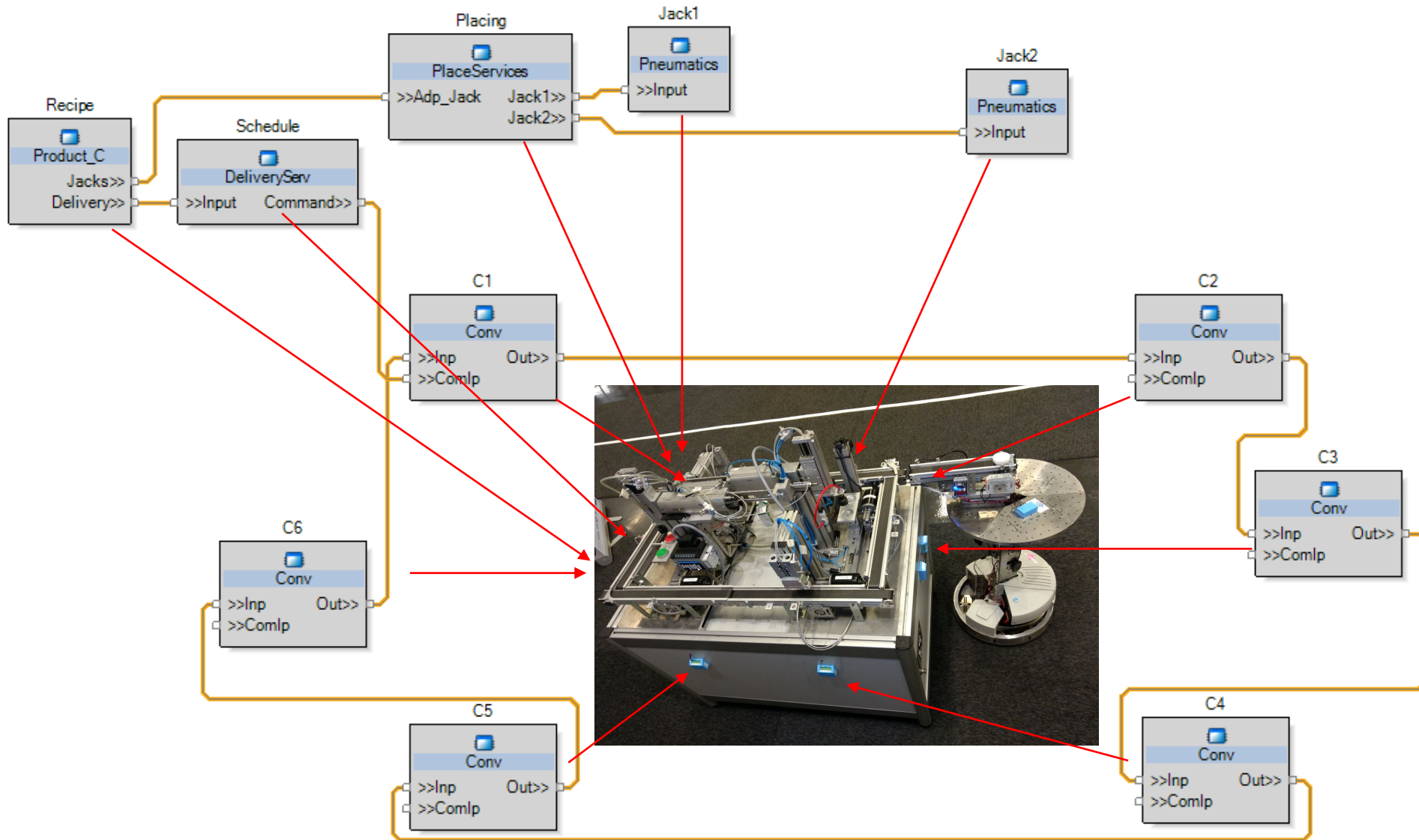
Product description layer

Planning services layer

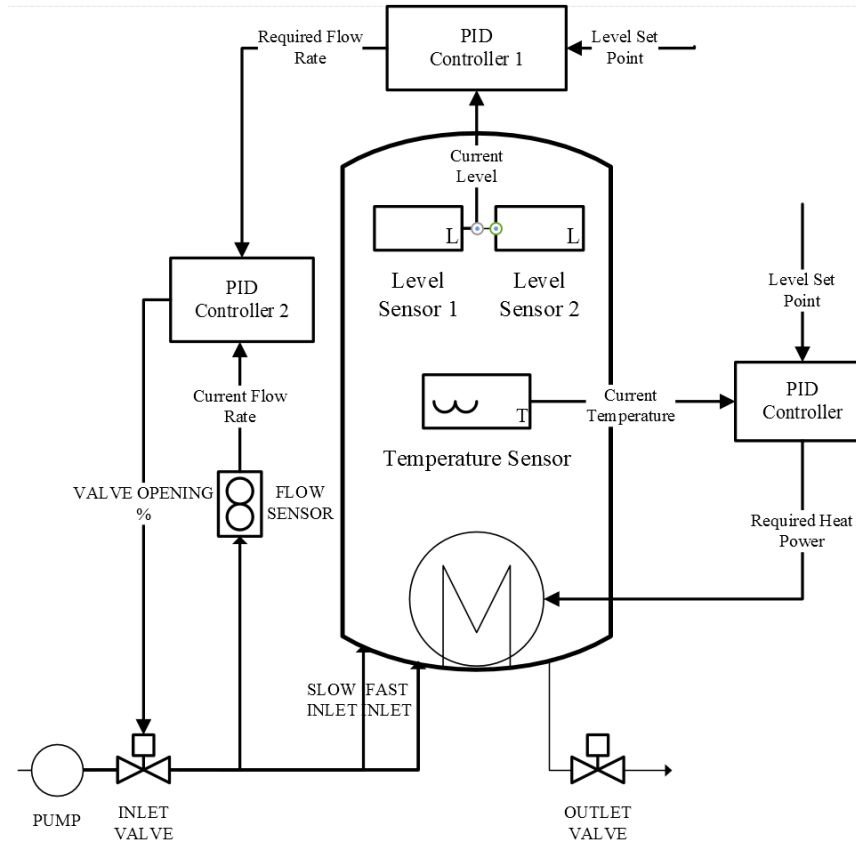
Execution services layer



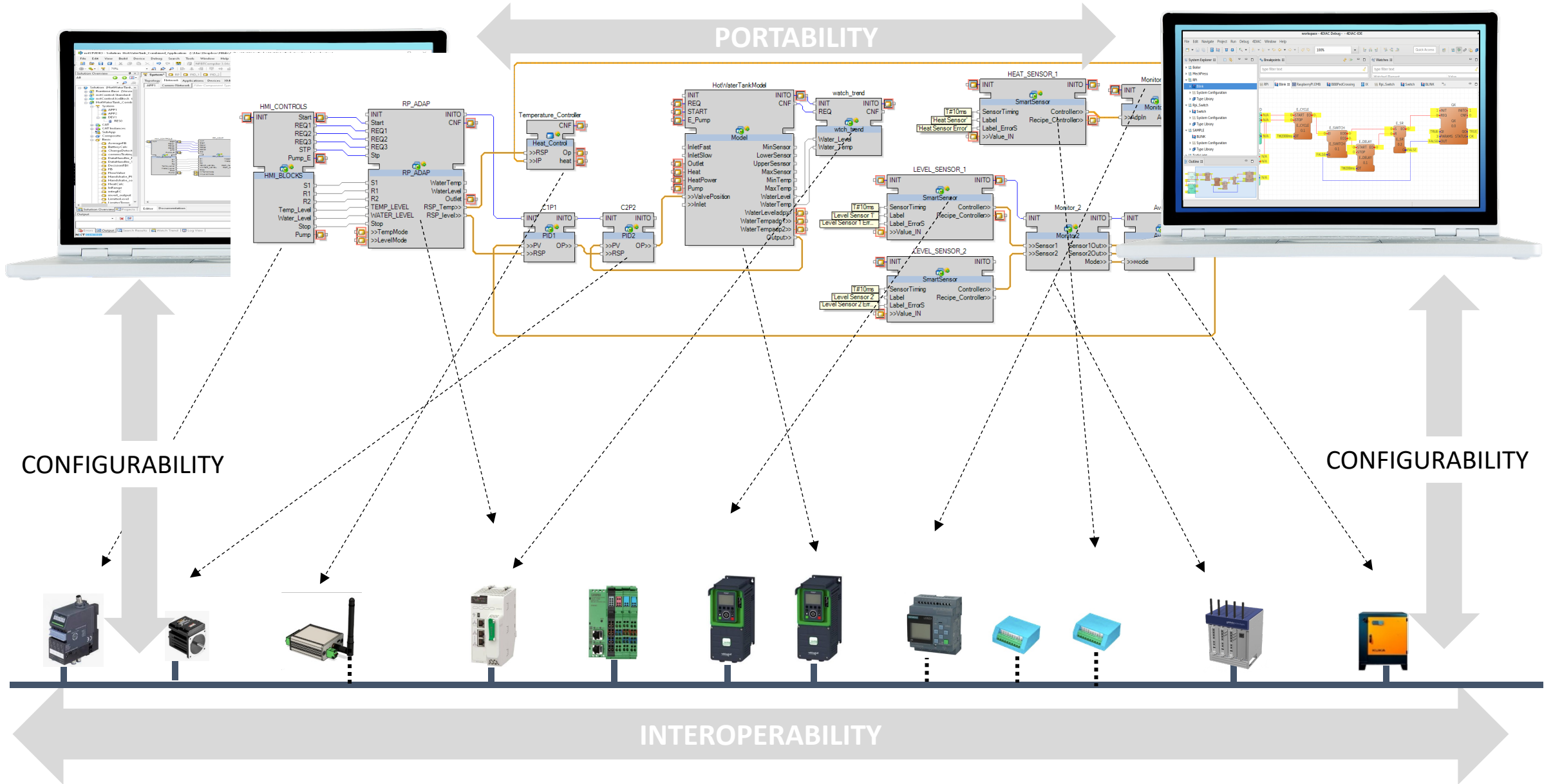
Distributed deployment



What is IEC 61499?



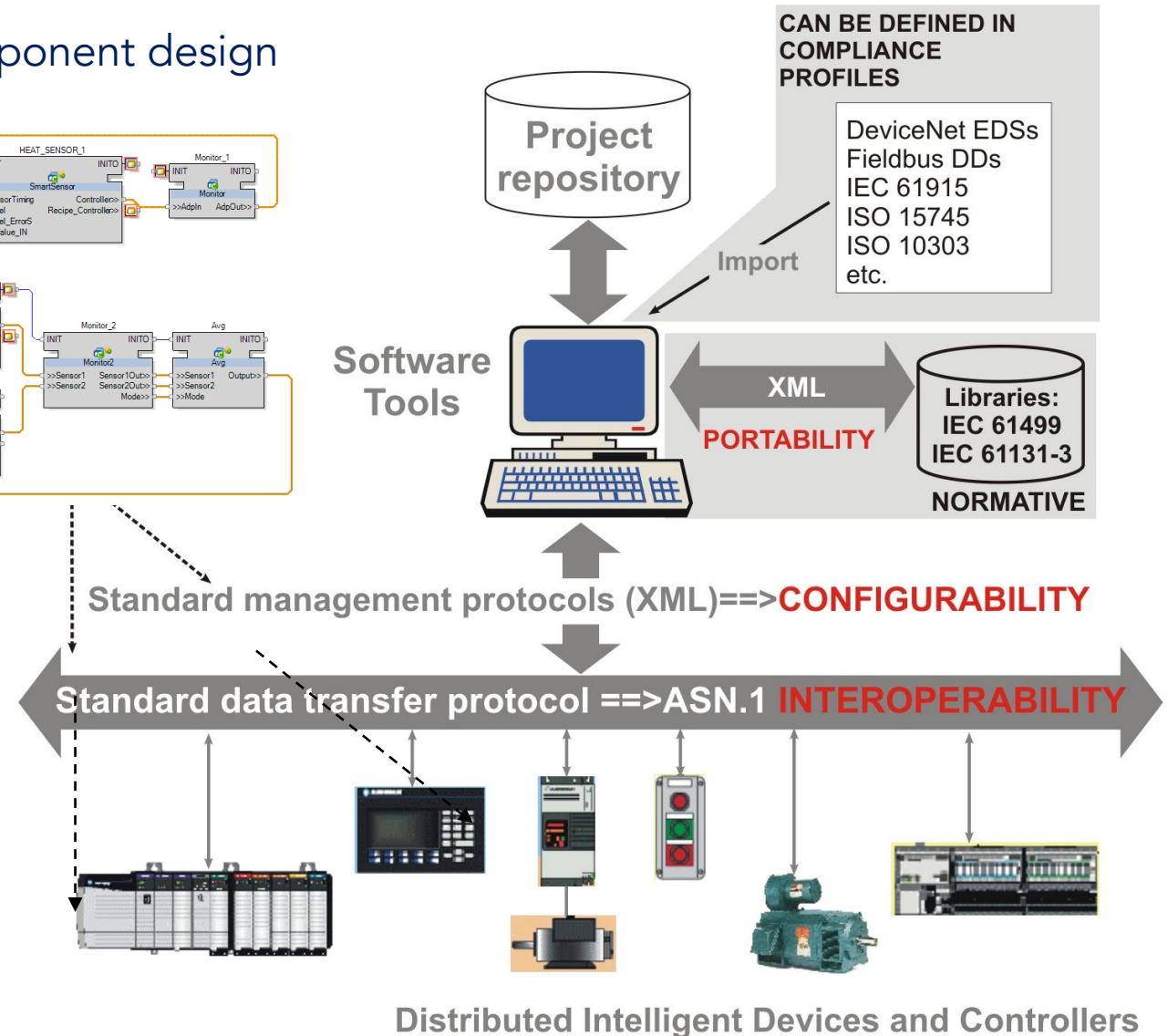
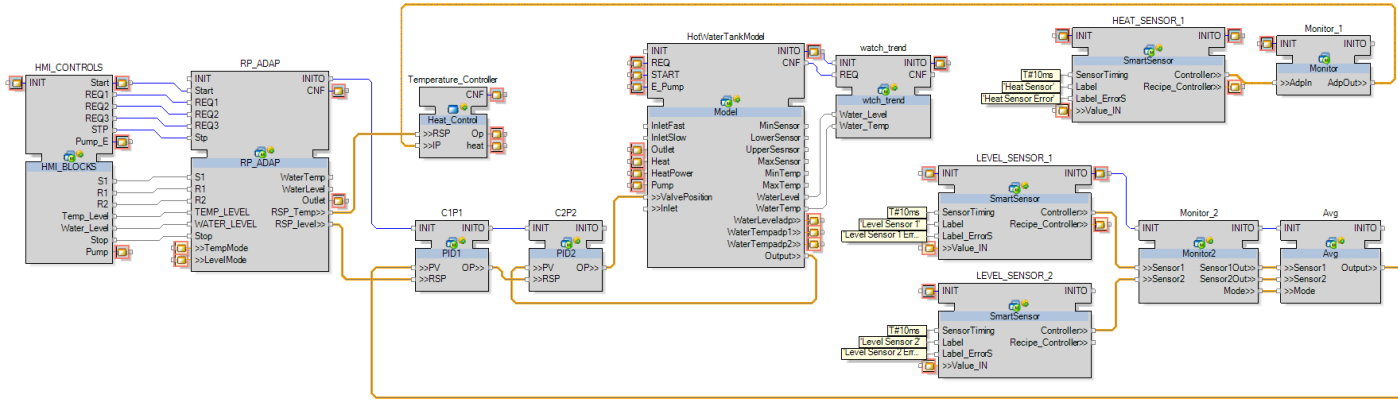
What is IEC 61499?



How does IEC61499 address the needs?

Open Distributed Automation Architecture

Extends IEC 61131 PLC programming with a component design language for distributed automation systems



Distributed Intelligent Devices and Controllers

IEC 61499 International Standard

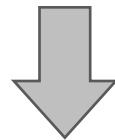
International Electrotechnical Commission IEC TC 65B/ WG7/ MT15

A component-based, open reference architecture for
Distributed Industrial - Process Measurement & Control Systems (IPMCS)
which can meet both current and future requirements for intelligent automation

1992 – project started

2005 – first edition

2011 – second edition



Based on and
extends the
standards



PLC Function Blocks (IEC 61131-3)

DCS Function Blocks (IEC 61804 project)

IEC61499 ingredients

- Extends PLC programming languages of IEC 61131-3 to distributed systems design
- Uses visual block-diagram representation for component-based design
- Uses state-machines for defining components' logic
- Components communicate via message passing

- All these features are common in engineering of automation and embedded systems, in IEC 61499 they are brought making powerful engineering framework for modern automation needs

Origins

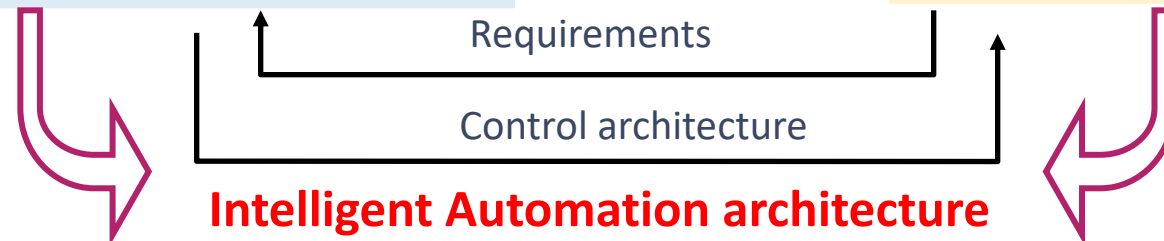
Architectural Co-Evolution

IEC 61499

- **Parent organization:** IEC
- **Working group:** TC65/WG6
- **Goal:** Standard model (function blocks) for control encapsulation & distribution
- **Started:** 10/90
- **Active development:** 3/92
- **Trial period:** 2001-03
- **Completion:** 2005

Holonic Manufacturing Systems (HMS)

- **Parent organization:** IMS
- **Working group:** HMS Consortium
- **Goal:** Intelligent manufacturing through holonic (autonomous, cooperative) modules
- **Feasibility study:** 3/93-6/94
- **First phase:** 2/96 - 6/00
- **Second phase:** 6/00-6/03



Origins/block diagram programming

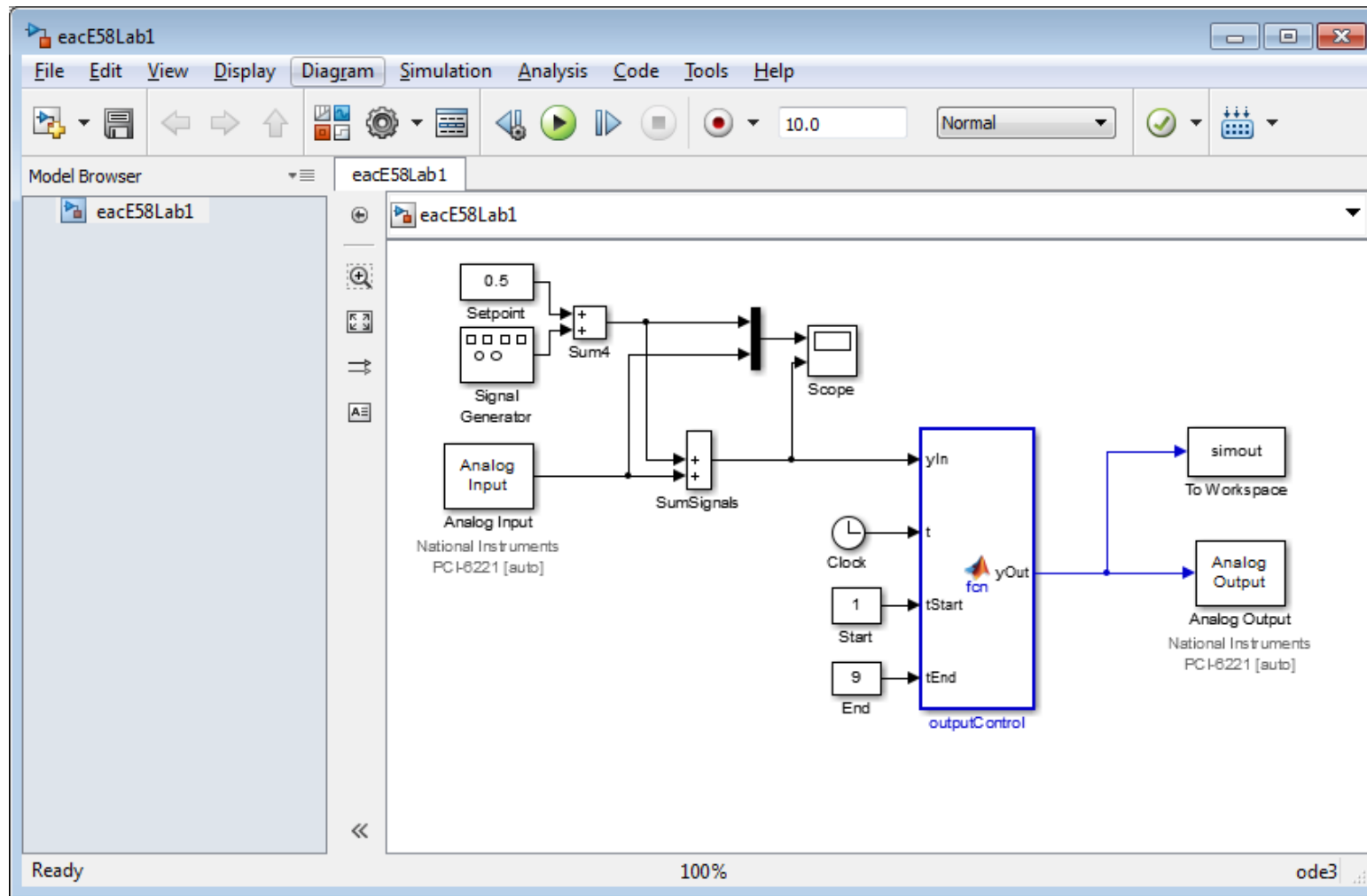
Block diagrams as a programming language

- Block diagrams is a convenient abstraction for representing component-based engineering
- It is used widely in Industrial Automation, Embedded systems, Cyber-Physical Systems, including automotive and aerospace

Block Diagram Programming in Automation

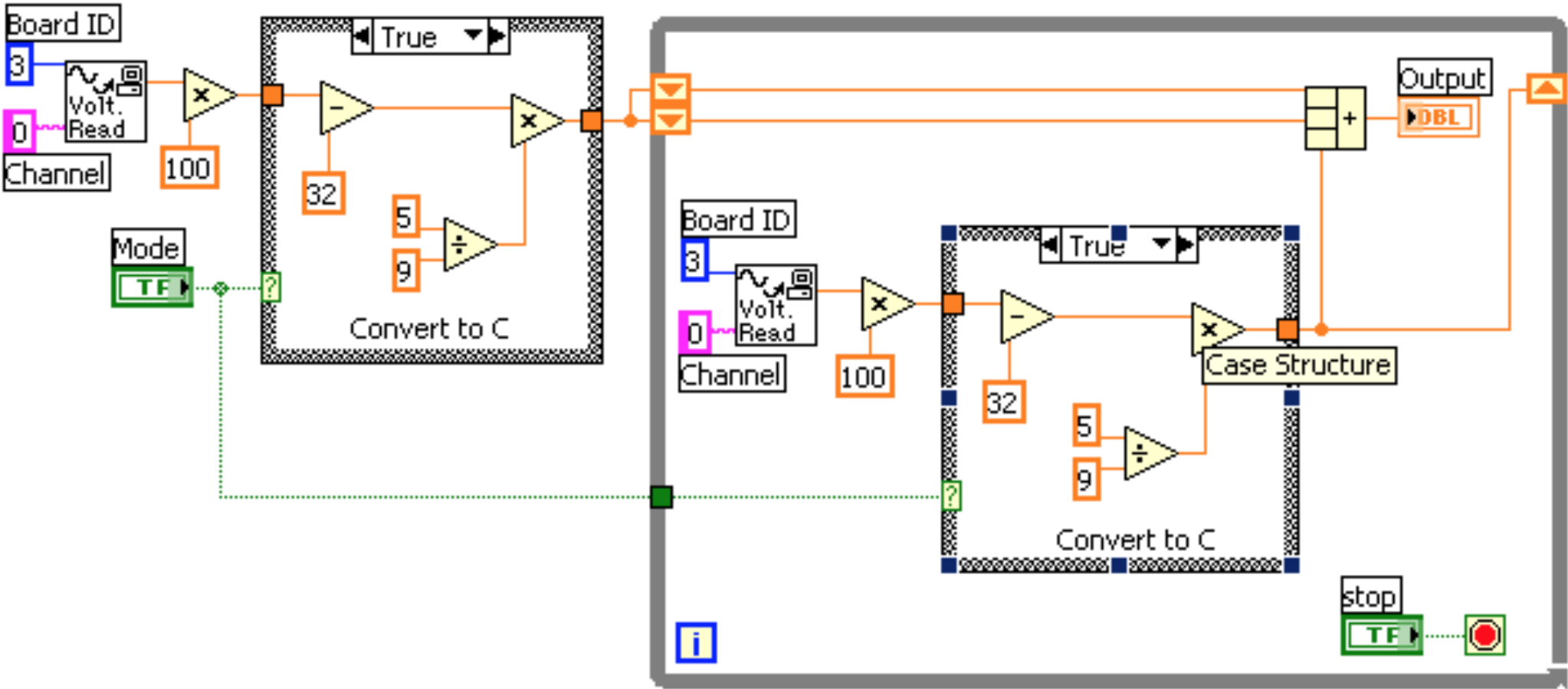
The screenshot displays the STRATON - FBDdemo software interface. The main workspace shows a block diagram for a blinker function. The diagram starts with a 'bCommand' input block connected to a 'RETURN' block. Below this, a comment states: "Dont execute this program if bCommand is FALSE". The main logic is enclosed in a 'lab:' block with the comment: "Blink main command is always TRUE - Select fast or slow period for blinking (fast is 2 times faster)". Inside the lab, there are two input blocks: 'Fast command bFast' and 'Slow period tPerio'. The 'bFast' block is connected to the 'IN1' input of a 'sel' (selector) block. The 'tPerio' block is connected to the 'IN2' input of the 'sel' block. The 'sel' block is also connected to the 'IN' input of an 'any_to_time' block, which has a 'Converts to TIME' label. The 'any_to_time' block is connected to the 'IN' input of a 'Blink1' block. The 'Blink1' block has a 'TRUE' input and a 'CYCLE' output. The 'CYCLE' output is connected to the 'main output blinking signal bOut' block. The diagram ends with a 'The end' comment. The software interface includes a menu bar (File, Edit, View, Insert, Project, Tools, Window, Help), a toolbar, a workspace with a 'Workspace' pane on the left, a 'Build' pane at the bottom, and a 'Block selection' pane on the right. The status bar at the bottom shows 'OffLine 127.0.0.1' and other system information.

Matlab / Simulink

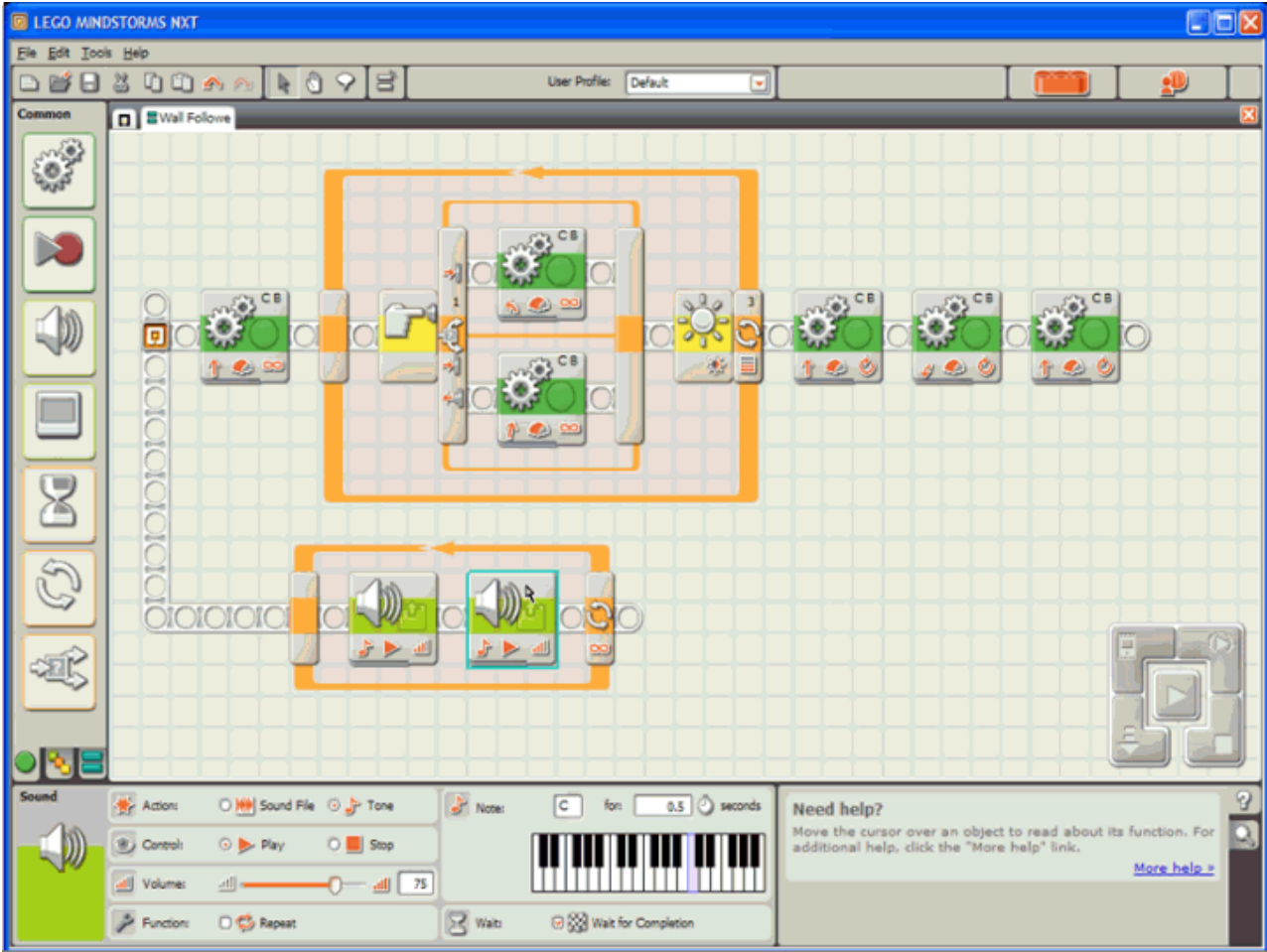
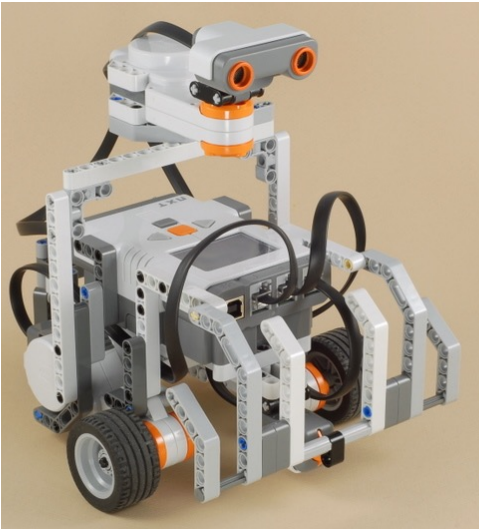


- Block diagrams is a convenient way of defining systems' functionality.
- It is quite standard in control systems engineering and in modelling.

LabView

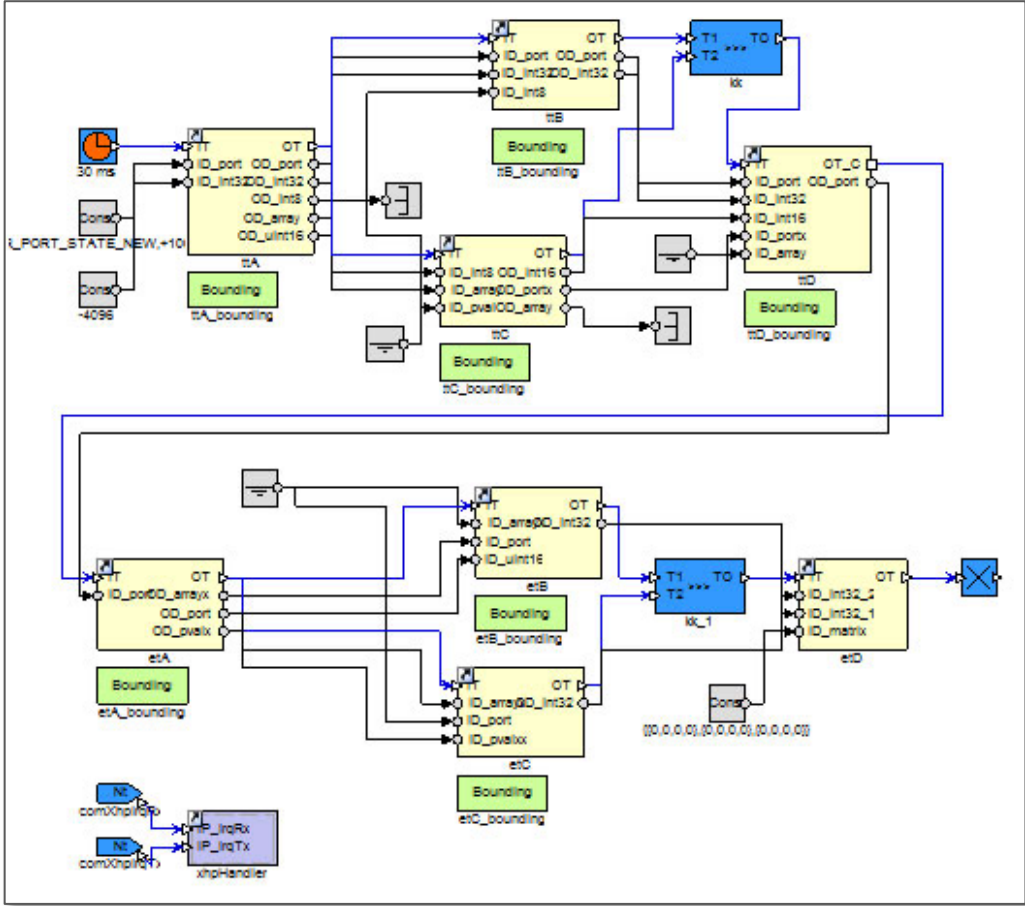


LEGO MindStorm

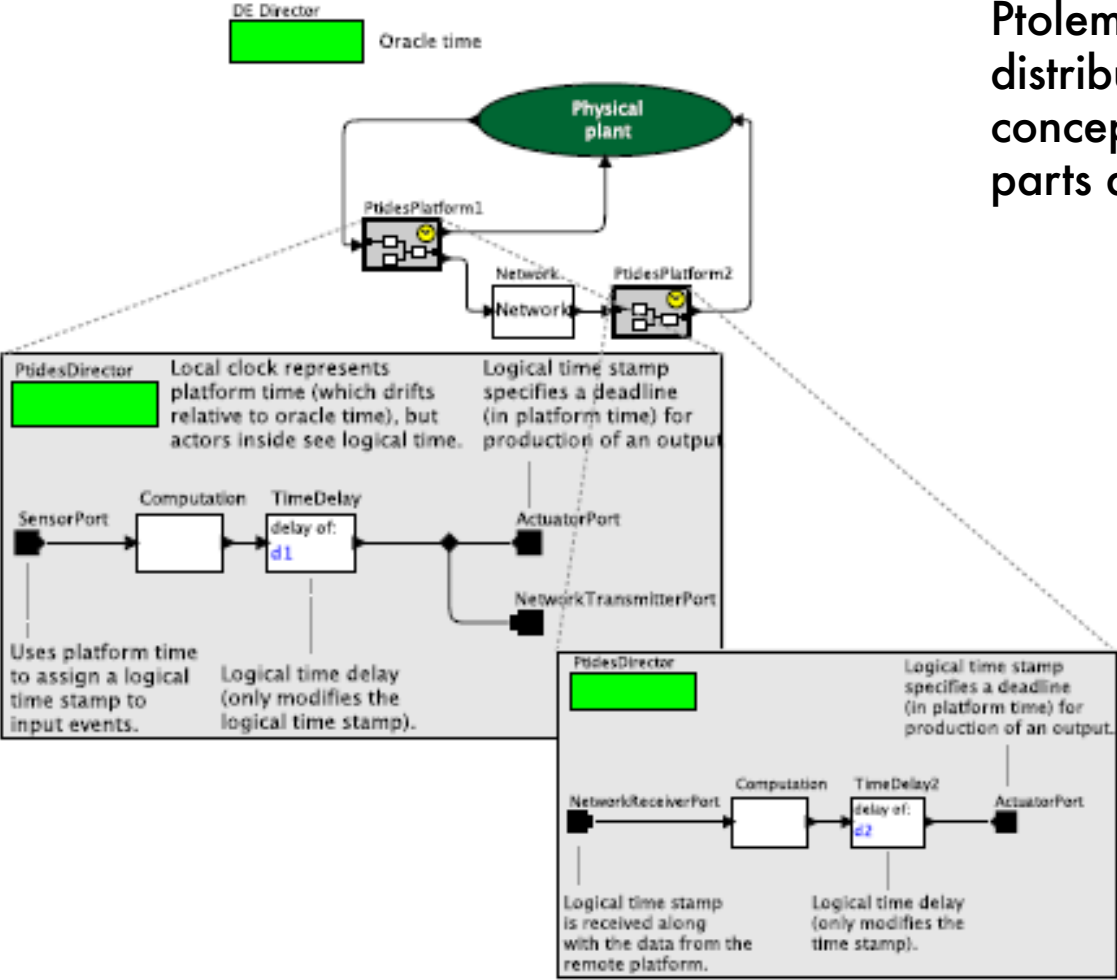


The screenshot shows the LEGO MINDSTORMS NXT software interface. The main workspace contains a sequence of programming blocks: a 'Wait' block, a 'Sound' block, a 'Wait' block, a 'Sound' block, a 'Wait' block, a 'Sound' block, a 'Wait' block, a 'Sound' block, a 'Wait' block, a 'Sound' block, and a 'Wait' block. The blocks are connected in a sequence, with some blocks having a 'Repeat' icon. The interface includes a menu bar (File, Edit, Tools, Help), a toolbar, and a 'Sound' panel at the bottom with various controls like 'Action', 'Sound File', 'Tone', 'Note', 'Volume', 'Function', and 'Wait'. A 'Need help?' section is visible in the bottom right corner.

Rubus: embedded systems programming

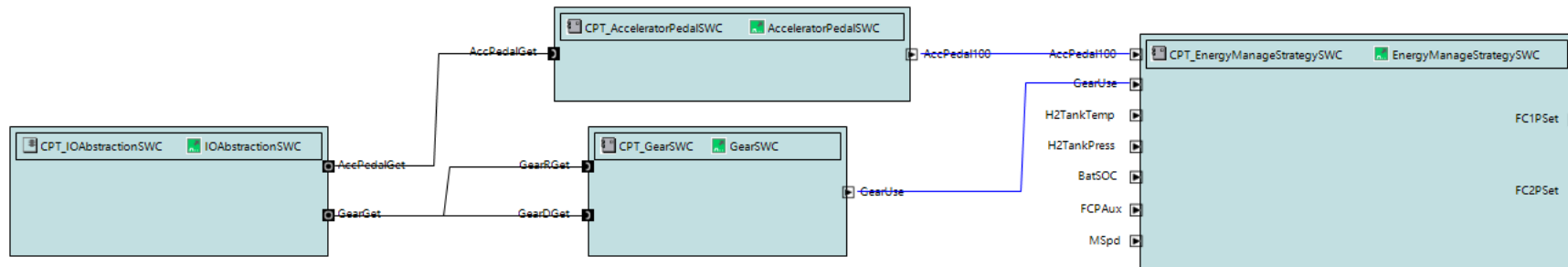


Ptolemy II



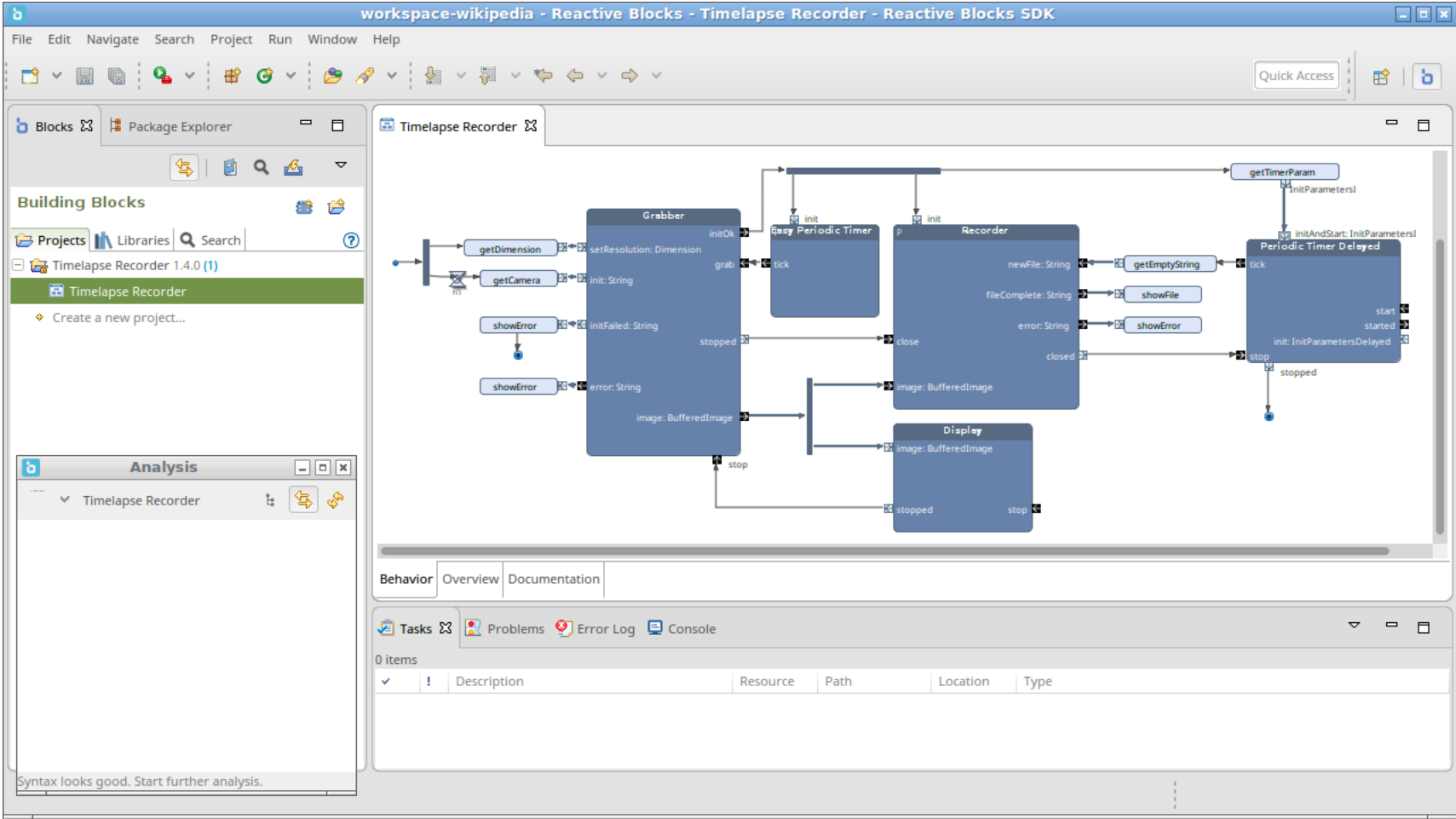
Ptolemy and pTides is a semantic framework for distributed CPS that uses a uniform time concept for both physical and computational parts developed at UCLA Berkeley.

AUTOSAR



ReactiveBlocks

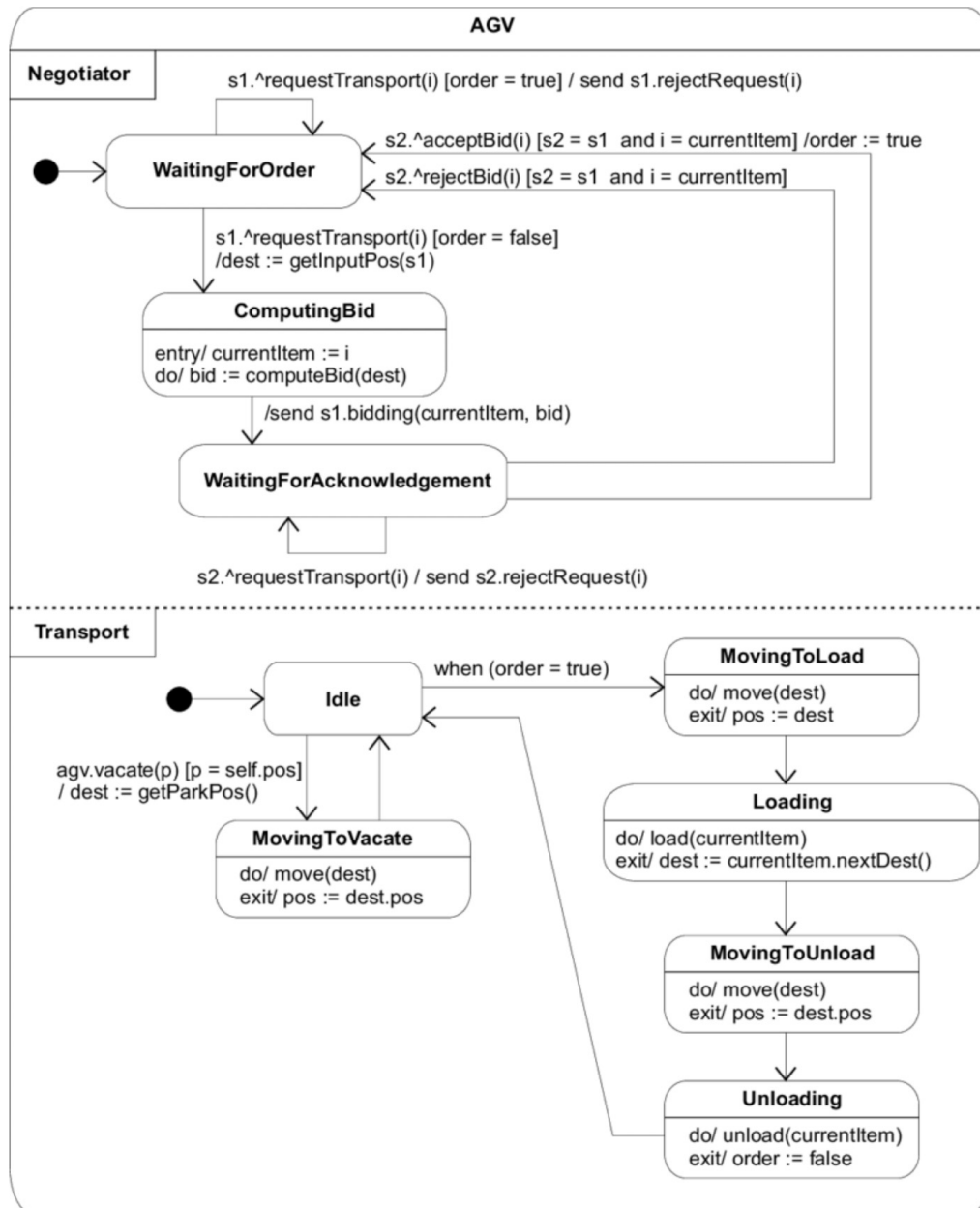
Compiled to Java



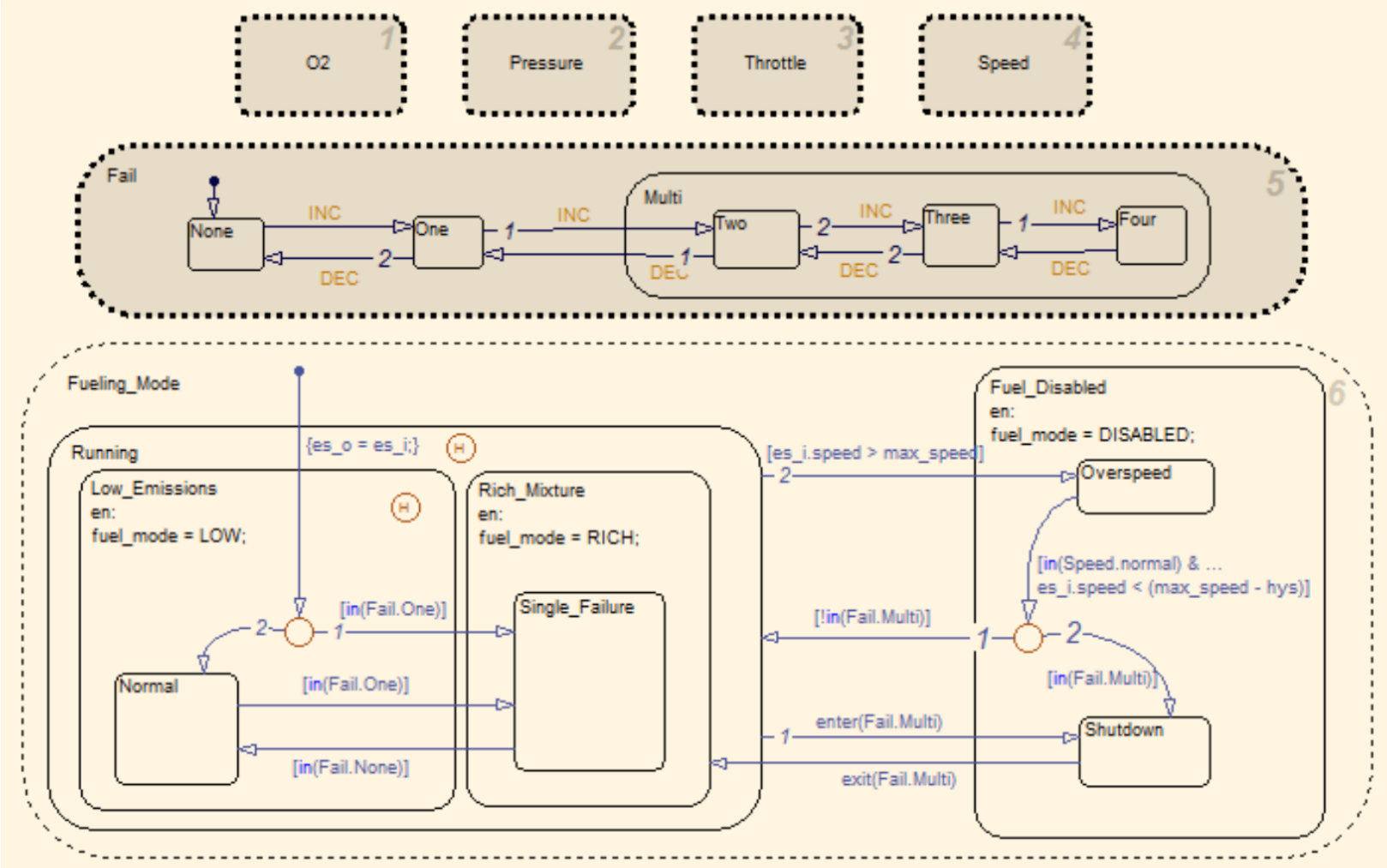
State machines

UML State Charts

- An important supporting tool for Model-Based Engineering in software and embedded systems engineering.



Matlab/Symulink StateFlow



CoDeSys State Charts

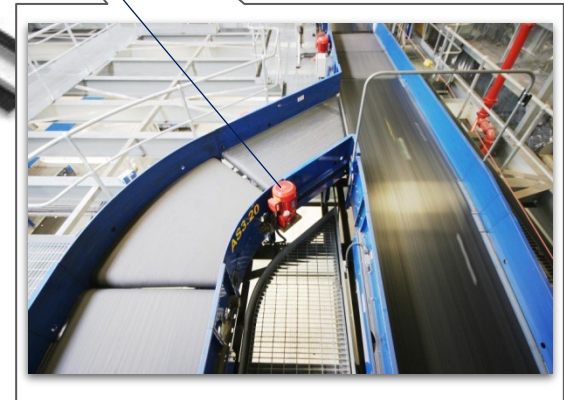
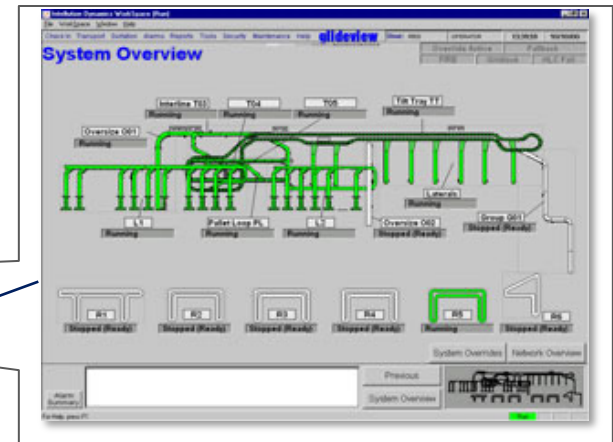
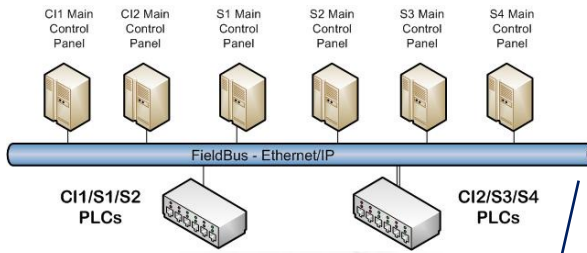
Dev.Application.Kaffemaschine

Expression	Type	Value	Prepared value	Comment
bMakeCoffee	BOOL	TRUE		
bShutDown	BOOL	FALSE		
bPowerUp	BOOL	TRUE		
nGrind	INT	30		
_UML_SC_Kaffemaschine	_UML_SC_7cc42eb0...			

State Chart Description:

- Power_Off** (Start State) transitions to **Init** on event **[bPowerUp]**.
- Init** transitions to **Idle** on event **[TRUE]**.
- Idle** transitions to **Shutdown** on event **[bShutDown]**.
- Shutdown** transitions to **Power_Off** on event **[TRUE]** and action **ActShutdown**.
- Idle** transitions to **CreateCoffee** on event **[bMakeCoffee]**.
- CreateCoffee** (Composite State):
 - Water** (1) transitions to **Pour** on event **[TRUE]** and action **Boil**.
 - Milk** (2) transitions to **Pour** on event **[TRUE]** and action **Heat**.
 - Coffee** (3) transitions to **Pour** on event **[nGrind >= 100]** and actions **Grind** (ENTRY / ActEntry, DO / ActDo).
- Pour** transitions to **Idle** on event **[TRUE]** and action **CoffeeReady**.

Structure of the course



UNIVERSAL
AUTOMATION.ORG