UNISA Current Research activities

F. Basile, [1]

[1]DIEM, Università di Salerno, Italy

UNISA Campus

- The largest University Campus in Italy (11 kms from the city of Salerno)
- About 45.000 students
- 16 Departments (now including Medicine)

UNISA Automatic Control Group

- 1 Full Professor (Pasquale Chiacchio)
- **1 Associate Professor (Francesco Basile)**
- 1 Assistant Professor (Alessandro Marino)
- 1 Post-Doc (Jolanda Coppola)
- 1 Phd student (Diego Gerbasio)

http://www.automatica.unisa.it

Università degli Studi di Napoli - Federico II (Italy)

- Chalmers University (Sweden)
- Aix Marseille Université (France)
- Universitè du Havre (France)
- Cankaya Universitesi (Turkey)
- Universidad de Zaragoza (Spain)
- Universiteit Gent (Belgium)
- Università di Cagliari (Italy)
- Politecnico di Milano (Italy)

Research topis

(a)

(b)

Theory

- Robot Modelling and Control
- **Discrete Event Systems Modelling and Control**

Applications

- Hyperflexible robotic cells
- **Automated material handling systems**

Theory

- **Process mining - DES Identification**
- Timed DES modelling and control
- **Cyber-Physical systems in Industrial Automation**

Applications

- **DES Fault detection/ Model repair**
- **Workflow Managment Systems**
- **Logistic Systems**
- Urban traffic modelling and control

Automated modeling of discrete event processes/systems from external observation of their behavior is a challenging problem that received a lot of attention in the last decade.

This problem has been addressed by the Discrete Event Systems (DESs) and Workflow Management Systems communities, under different approaches and names - *DES Identification* [Giua and Seatzu (2005)] and *Process mining* [van der Aalst (2004)].

[Giua and Seatzu (2005)] Giua, A. and Seatzu, C. (2005). Identification of free-labeled Petri nets via integer programming. 44th IEEE Conf. on Decision and Control (CDC05), Seville, Spain, 7639-7644.

[van der Aalst (2004)] van der Aalst, W. and Weijters, T. and Maruster, L., Workflow mining: discovering process models from event logs, IEEE Transactions on Knowledge and Data Engineeringvol.16, no.9, pp.1128,1142, Sept. 2004.

Process mining aims to discover, monitor and improve real processes by extracting knowledge from event logs that is a collection of sequential events and information about the system:

discovery - the model of the system is obtained starting from event logs, without any other knowledge of the system;

conformance checking - an existing process model is compared with an event log of the same process to check if reality, as recorded in the log, conforms to the model and vice versa;

enhancement - an existing process model is extended or improved using information about the actual process recorded in some event log.

Model repair is a particular case of enhancement; it consists in modifying the nominal model of a system as consequence of the occurrence of the observation of discrepancies between the system nominal behavior and the system observed behavior, in the manner that the modified model completely describes the observed behavior.

These discrepancies are called *anomalies*.

Anomalies can be due to different reasons:

- workers start handling activity differently,
- system components degrade,
- action of external agents, etc.

The occurrence of one of these circumstances modifies the system dynamic as well as the duration of the activities of the system and as consequence the nominal model needs to be modified.

The model repair technique has been introduced for the first time in [Fahland and van der Aalst (2012)].

New subprocesses are added to the nominal model (a logical PN) in the manner that the resulting model fits the observed behavior and it is as similar as possible to the original one.

The original net is modified by adding new transitions.

[Fahland and van der Aalst (2012)] Fahland, D. and van der Aalst,W.M. (2012). Repairing process models to reflect reality. In A. Barros, A. Gal, and E. Kindler (eds.), Business Process Management, volume 7481 of Lecture Notes in Computer Science, 229-245. Springer Berlin Heidelberg.

In [Cabasino et al. (2014).], anomalies are called *faults* and the model repair is presented as the identification of the *faulty model* of a logical PN system: the occurrence of a faulty firing sequence (i.e., a sequence that cannot be generated by the nominal model of the system) is associated to the unobservable firings of fault transitions, that must be opportunely added and linked to the nominal model of the system, to obtain the faulty model.

Hence, also in this case, the structure of the nominal model is changed.

[Cabasino et al. (2014).] Cabasino, M.P., Giua, A., Hadjicostis, C.N., and Seatzu, C. (2014). Fault model identification and synthesis in Petri nets. Discrete Event Dynamic Systems, 1-22.

Definition (Time Petri net system)

Let $\mathcal{I}$ be the set of closed intervals with a lower bound in $\mathcal{Q}$ and an upper bound in $\mathcal{Q} \bigcup \infty$. A *Time* Petri net (TPN) system is the triple $S = \langle N, \boldsymbol{m}_0, I \rangle$, where N is a standard P/T net, $\boldsymbol{m}_0$ is the initial marking, and $I : T \rightarrow \mathcal{I}$ is the *statical firing time interval function* which assigns a firing interval $[l_j, u_j]$ to each transition $t_j \in T$.

Definition (Timed firing sequence)

A sequence

$$\sigma = (T_1, \tau_1) \dots (T_q, \tau_q) \dots (T_L, \tau_L) \, ,$$

where T_q is the set of transitions fired at time τ_q and $\tau_1 < \tau_2 \cdots < \tau_L$ denote firing time instants is called *timed firing sequence*. The position q the couple (T_q, τ_q) occupies in the sequence is called *time step*, so (T_1, τ_1) is associated with step 1, (T_2, τ_2) is associated with step 2 and so on; the number of triples (T_q, τ_q) in σ is called length $L = |\sigma|$ of the timed firing sequence.
The notation $\boldsymbol{m}[\sigma\rangle\boldsymbol{m}'$ is used to denote that $\boldsymbol{m}'$ is reached from $\boldsymbol{m}$ by firing σ. $\diamondsuit$

Definition (Timed Language)

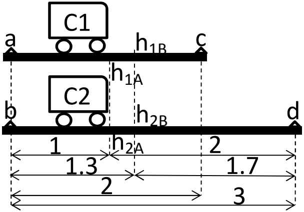
Given a TPN system $S = \langle N, \boldsymbol{m}_0, I \rangle$, its timed language, named $\mathcal{L}(S)$, is defined as the set of timed firing sequences generated by S from the initial marking $\boldsymbol{m}_0$.

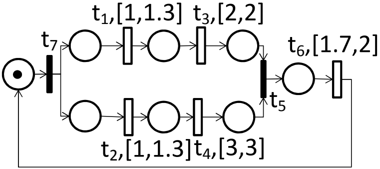
Let $S = \langle N, \boldsymbol{m}_0, I \rangle$ *be the known fault-free system, with* $N = (P, T, \mathbf{Pre}, \mathbf{Post})$*, that generates the nominal language* $\mathcal{L}(S)$*. For each timed firing sequence* $\sigma = \sigma_{prev}(T_q, \tau_q)$ *such that:*

- $\sigma \notin \mathcal{L}$*;*
- $\sigma_{prev} \in \mathcal{L}$*;*
- σ_{prev} *is a subsequence of* σ*, of length* $q-1$*;*

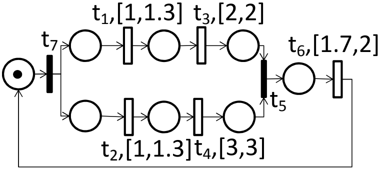
given the set of fault transition T^f*, with cardinality* n_f*, the goal is to identify the faulty incidence matrices* $\mathbf{Pre}_f$ *and* $\mathbf{Post}_f$*, with dimension* $m \times n_f$ *such that,* σ *belongs to the language generate by* $\tilde{S} = \langle \tilde{N}, \boldsymbol{m}_0, \tilde{I} \rangle$*, where* $\tilde{N} = (P, T \bigcup T^f, \left[\mathbf{Pre}\ \ \mathbf{Pre}_f\right], \left[\mathbf{Post}\ \ \mathbf{Post}_f\right])$ *and*

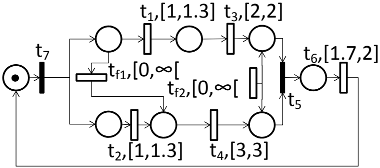
$$\tilde{I}(t) = \begin{cases} I(t) & \text{if } t \in T \\ [0, \infty[& \text{if } t \in T^f \end{cases}$$





$\sigma = (\{t_7\}, 0)\ (\{t_2\}, 1)\ (\{t_4\}, 3.5))$ and $\sigma' = (\{t_7\}, 0)(\{t_1\}, 1)(\{t_2\}, 1.3)(\{t_3\}, 3)\ (\{t_5, t_6\}, 3.5)$. Sequence σ is a faulty sequence since 1) an unexpected firing of t_4 and 2) the missing firing of t_1 are observed at time τ_3. Sequence σ' is a faulty sequence since an unexpected firing of t_5 is observed at time τ_5.





The firing of t_{f_1} can be due to the following faults: 1) an acceleration of C2 speed while the car is going from point *b* to point *d* and 2) the missing starting of C1.

The firing of t_{f_2} can be due to the wrong detection of both car at destination when actually only C1 is arrived (such a fault can be due, for example, to an accidental occupation of the sensor by an operator).

Improved integration of heterogeneous devices and systems, with particular emphasis on platform independence, real-time requirements, robustness, security and stability of solutions, among other major requirements.

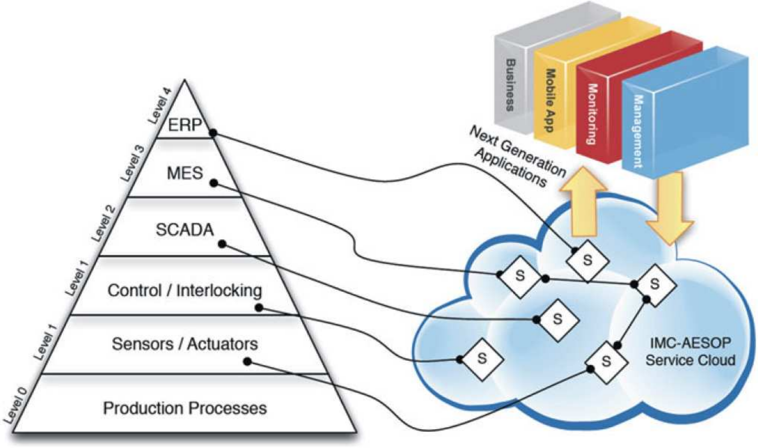
CPSs arise from this integrations of computation and physical processes.

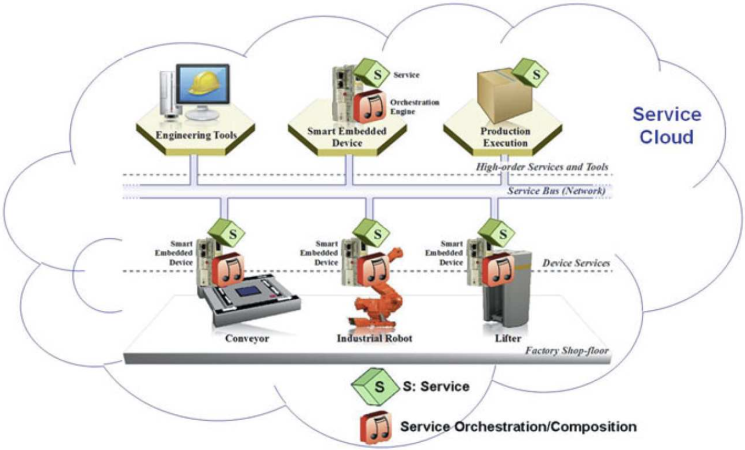
Embedded computers and networks monitor and control the physical processes usually with feedback loops where physical processes affect computations and vice versa.

The umbrella paradigm underpinning novel collaborative systems is to consider the set of intelligent system units as a conglomerate of distributed, autonomous, intelligent, proactive, fault-tolerant and reusable units, which operate as a set of cooperating entities.

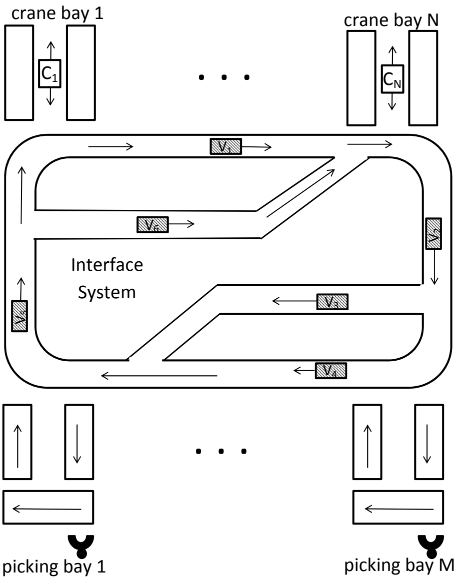
These entities are capable of working in a proactive manner, initiating collaborative actions and dynamically interacting with each other to achieve both local and global objectives.

This evolution towards global service-based infrastructures indicates that new functionality will be introduced by combining services in a cross-layer form, i.e. services relying on the enterprise system, on the network itself and at device level will be combined.


Level 4
ERP
Level 3
MES
Level 2
SCADA
Level 1
Control / Interlocking
Level 1
Sensors / Actuators
Level 0
Production Processes
Business
Mobile App
Monitoring
Management
Next Generation Applications
S
IMC-AESOP Service Cloud



Distributed path planning...


crane bay 1
crane bay N
C1
CN
V1
V2
V3
V4
V5
V6
Interface System
picking bay 1
picking bay M

The cyber part of the control architecture in automated warehouse systems reduces to the control algorithms implemented according to International Electrotechnical Commission (IEC) programming standard.

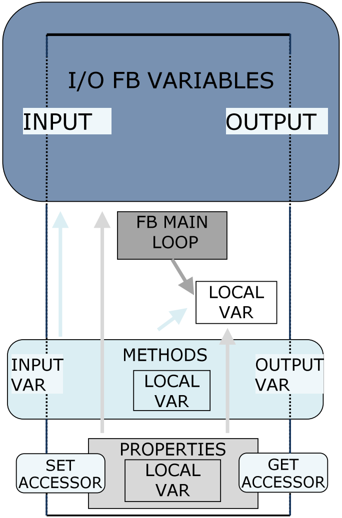
As for the physical part, it can be assumed that, at a certain level of abstraction, there is a software component for each physical unit, e.g. a conveyor, an elevator, and so on.

However, the complexity of modern warehouse systems, requires big interfaces (e.g. a carousel, shuttles, rail guided vehicles) between cranes and picking area and so many vehicles must be used.

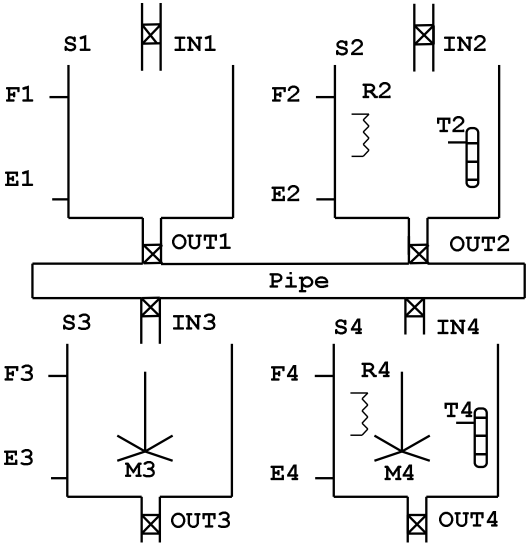
When each crane cycle involves more than one picking and deposit, the number of SUs moved by vehicles at a time in the interface area grows, and then a significant time is required to cover the interface guidepath.

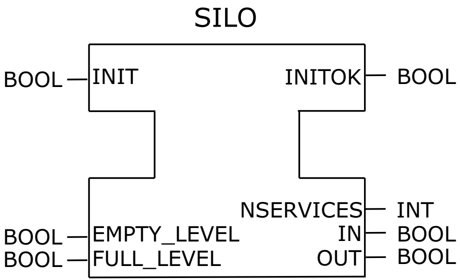
In practice, vehicles as well as conveyors can be moved with constant speed, can stop at the interface points to load (unload) a SU from (to) another subsystem bay, and can vary their speed with constant acceleration if a particular condition occur (distance between two vehicles goes under a threshold, a particular point is reached...).

To conclude, physical dynamic influence the optimization, developed essentially in the cyber part.



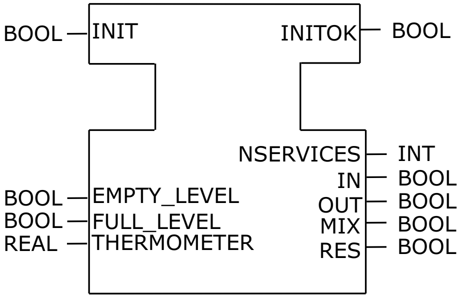
IEC 61131 3rd edition (2013) - OOP has been introduced


S1
IN1
S2
IN2
F1
F2
R2
T2
E1
E2
OUT1
OUT2
Pipe
S3
IN3
S4
IN4
F3
F4
R4
T4
E3
M3
E4
M4
OUT3
OUT4

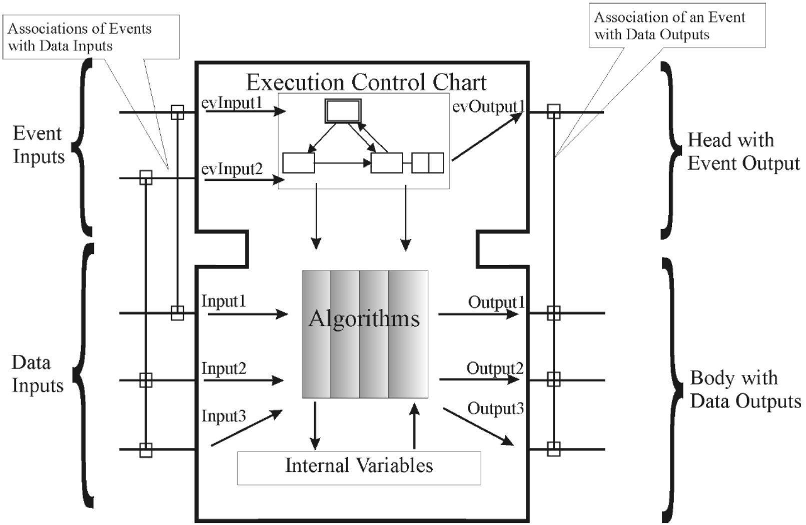

SILO
BOOL
INIT
INITOK
BOOL
NSERVICES
INT
BOOL
EMPTY_LEVEL
IN
BOOL
BOOL
FULL_LEVEL
OUT
BOOL

```
FUNCTION_BLOCK SILO
  VAR_INPUT
      EMPTY_LEVEL:BOOL; (* sensor of empty level *)
      FULL_LEVEL:BOOL; (* sensor of full level *)
      INIT:BOOL; (* initialization event *)
  END_VAR
  VAR_OUTPUT
      IN:BOOL; (* input valve *)
      OUT:BOOL; (* output valve *)
      N.SERVICES:INT; (* number of active services *)
      INITOK:BOOL; (* initialization completed event *)
  END_VAR
  VAR
      ...
  END_VAR
  METHOD Set_Init
      ...
  END_METHOD
      ...
END_FUNCTION_BLOCK
```

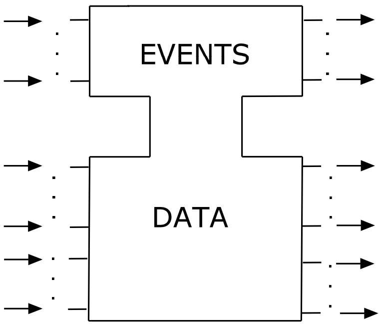
SILO WITH MIXER
AND HEATER



```
FUNCTION_BLOCK SILO_WITH_MIXER_AND_HEATHER EXTENDS SILO
  VAR_INPUT
        THERMOMETER:REAL; (* temperature sensor *)
  END_VAR
  VAR_OUTPUT
        MIX:BOOL; (* mixer motor *)
        RES:BOOL; (* heater resistance *)
  END_VAR
  (* local variables *)
        ...
  METHOD Set_Mixer
        ...
  END_METHOD
        ...
  PROPERTY TEMPERATURE : REAL
        GET
          TEMPERATURE:=(THERMOMETER*1.8)+32;
        END GET
  END_PROPERTY
END_FUNCTION_BLOCK
```



IEC 61499 (2005) - a new FB model, event based execution has been introduced.


EVENTS
DATA

```
         +------+
         | CTU  |
BOOL--->CU    Q|---BOOL
BOOL---|R       |
 INT---|PV   CV|---INT
         +------+
```

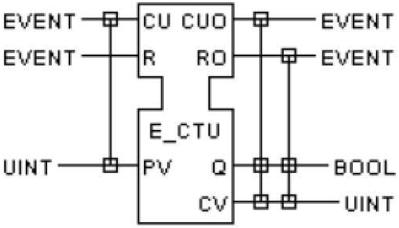
(a)

Graphical Representation

```
IF R
THEN CV := 0;
ELSIF CU AND (CV < PVmax)
     THEN
     CV := CV+1;
END_IF;
 Q := (CV >= PV);
```

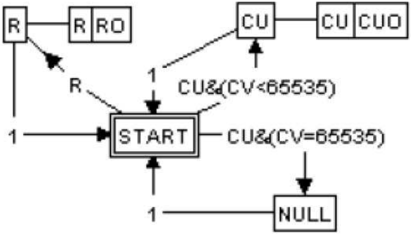
(b)

FB body in ST language



(a)

Graphical representation



(b)

Execution control chart

```
ALGORITHM R IN ST:
    CV := 0 ;
    Q := 0 ;
END_ALGORITHM
```

(c)

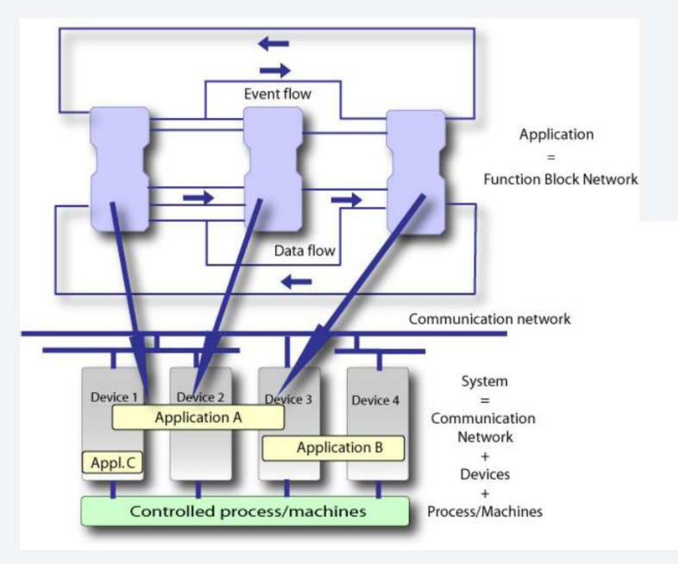
Reset algorithm

```
ALGORITHM CU IN ST:
  CV := CV+1;
  Q := (CV = PV);
END_ALGORITHM
```

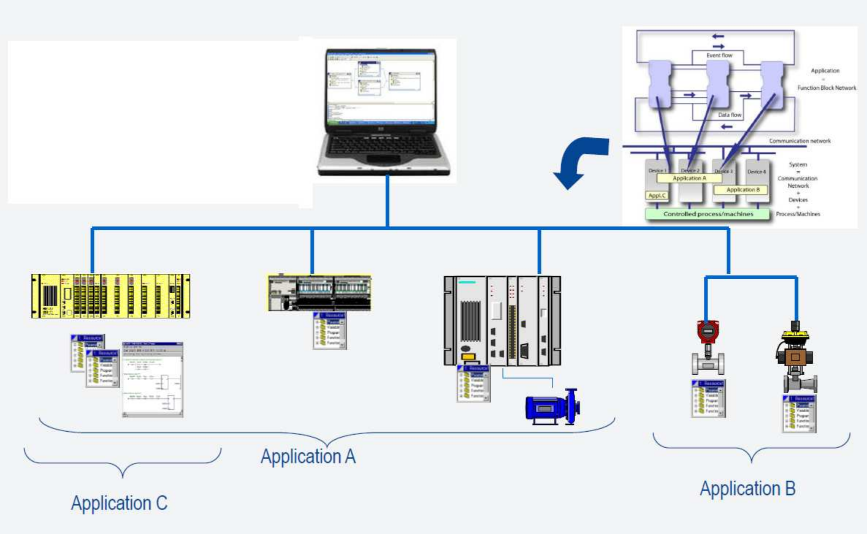
(d)

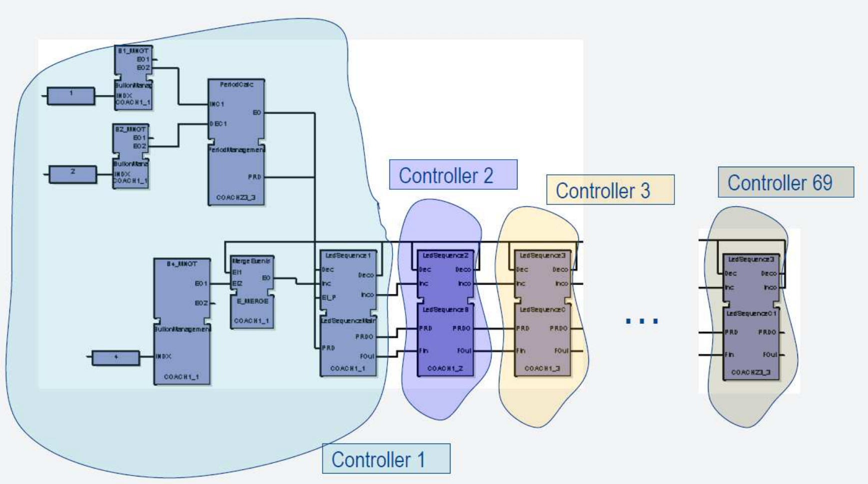
Count up algorithm

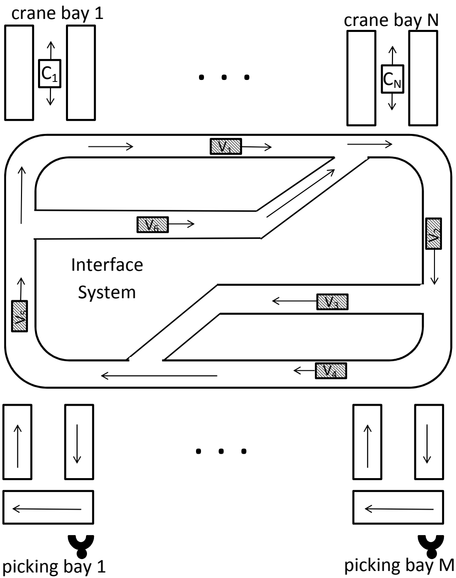
Cyber-Physical systems in Industrial Automation

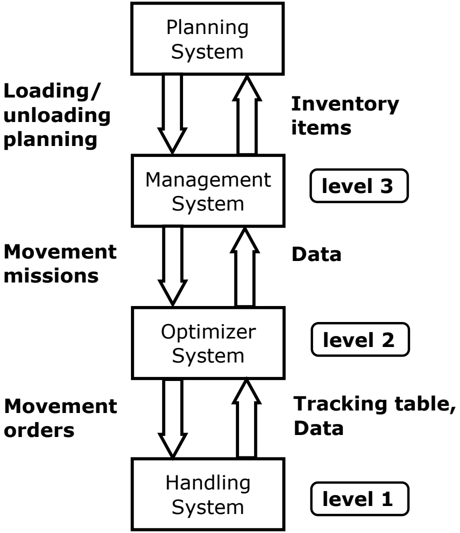


Cyber-Physical systems in Industrial Automation




Controller 2
Controller 3
Controller 69
Controller 1
...


crane bay 1
C1
crane bay N
CN
. . .
V1
V2
V3
V4
V5
V6
Interface
System
. . .
picking bay 1
picking bay M


Planning System
Loading/ unloading planning
Inventory items
Management System
level 3
Movement missions
Data
Optimizer System
level 2
Movement orders
Tracking table, Data
Handling System
level 1

Cyber-Physical systems in Industrial Automation

To obtain a Cyber-Physical Component (CPC) each FB is associated to a physical component (a conveyor, an elevator, etc.) to implement its cyber part where traditional and collaborative/intelligent functionalities are implemented.
As for example, traditional functionalities are:

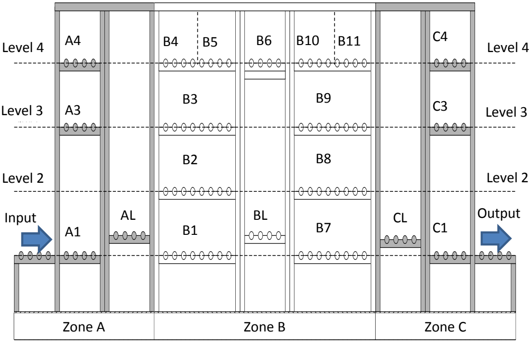
- Ability to move objects from one end to the other. At this aim each component is equipped at least with a photocell at its beginning plus one at the end, and a motor drive.
- Ability to indicate readiness to receive/deliver SUs in order to avoid inappropriate SU transfers from/to upstream/downstream component.
- To manage the shifting of tracking data according to the real SU movements in the plant.

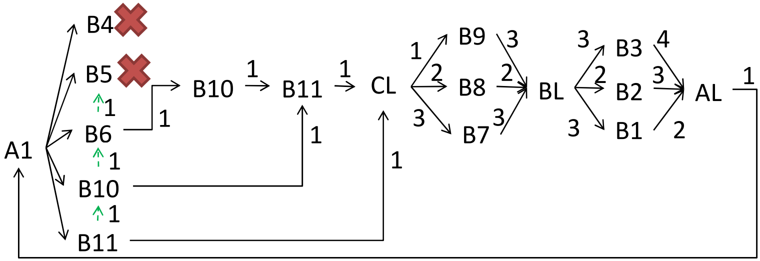
As for example, collaborative/intelligent functionalities are:

- Ability to change speed/acceleration according to the SU weight to save energy and/or to avoid breaking the loaded SUs.
- Prediction of SUs position on the component through time.
- Ability to cooperate with other components to implement a dynamic path building algorithm.
- Ability to choose an optimal path for a certain SUs.

Traditional functionalities represent what is called HS, while collaborative/intelligent functionalities make possibile to implement OS and MS at a device level in a completely distributed way. Just the warehouse location map must be centralized.

Cyber-Physical systems in Industrial Automation





Distributed path planning...

Codesys Development System, compliant to IEC 61131 standard, third version;

Forte, IEC 61499 Compliant Runtime Environment;

NXT Control, hybrid - IEC 61499 and IEC 61131 2nd ed. Compliant Runtime Environment (not just FB programming).

Thanks for your attention!