Transactions on Machine Learning
and Data Mining
Vol. 7, No. 1 (2015) 41-58
ISSN 1864-9734
ISBN: 879-3-942952-33-0

ibai Publishing
www.ibai-publishing.org

Transductional Holonic Formalisms for Control Systems Applications

Calin Ciufudean, Constantin Filote, and Valentin Vlad

Stefan cel Mare University
University Str 9, 720225, Suceava, Romania
calin@eed.usv.ro

Abstract. In this chapter the expression "holonic cyclic manufacturing systems" refers to complex assembly/disassembly systems or fork/join systems, kanban systems, and in general, to any discrete event system that transforms raw material and/or components into products. Such a system is said to be cyclic if it provides the same sequence of products indefinitely. This chapter considers the scheduling of holonic cyclic manufacturing systems and describes a new approach using Petri nets formalism. We propose an approach to frame the optimum schedule of holonic cyclic manufacturing systems in order to maximize the throughput while minimize the work in process. We also propose an algorithm to verify the optimum schedule.

Keywords: Petri net, event graph, holonic cyclic manufacturing system, minimum cyclic time, kanban system.

1 Introduction

The explosive development of information and communication technology networks (ICTNs) is designing and implementing more sophisticated and flexible approaches for monitoring and controlling systems; i.e. generically called manufacturing processes as we assume that every known system is dealing with inputs internal processing parameters and resulting outputs. It does this by linking high level systems, such as enterprise research planning (ERP) and customer relationship management (CRM) to low-level, real-time control devices such as digital signal processing (DSP) programmable logic controllers (PLCs), robots, conveyors, milling machines, etc [1-3]. For these approaches, the supervisory controllers can be distributed across several different computational devices. Following the holonic approach, the supervisory controller can be

considered as a single holon associated with a physical device, although atomically it consists of several sub-holons, one for each computational device involved in the control of physical device [3]. For a whole-parted system Koestler denoted the term "holon" , derived from the Greek word *holos* = whole with the suffix on (like neutron or proton) pointing out the part characteristic [4]. A holon could compile strategy out of its rules, which fits to its intentions and goals and with the interpolation of the environment. Typical holonic applications or models are applied mostly to the production processes, called holonic manufacturing systems (HMSs); we notice that similar to these approaches, they are modeled domain specifically [5, 6]. A complex holonic structure has a hierarchical order, rules and strategies framed by communication and reaction abilities. Most of the time, the hierarchies are looked at as rigid and inflexible shapes. However, some approaches are not following this like the approaches described in [7], and the one proposed here. An interesting situation occurs in both designing and controlling flexible manufacturing systems (FMSs), where there is no hardware support to ensure synchronization of the common resources flows (e.g. practical situations where the outcome that changes the states of FMS depends on the global sequence of events). In order to model and to ensure an optimized control of these situations, we propose an approach based on fundamental class of Petri nets that are well known to produce safe models and safe protocols for common flows of resources in large and complex systems. PNs modelling is more compact than the automata approach and is better suited for modelling systems with parallel and concurrent activities. In addition, PN has a friendly graphical representation with a powerful algebraic formulation and models clearly the synchronization involved in concurrent systems. Thus, it has generated intense interest from scientists and researchers. In this chapter, we focus our discussion on techniques for the predicting and verifying the performance of holonic distributed systems. We consider a holonic distributed system as a loosely or a tightly coupled processing element working co-operatively and concurrently on a set of related tasks. In general, there are two approaches for performance evaluation: deterministic models and probabilistic models. In deterministic models, it is usually assumed that the task arrival times, the task execution times, and the synchronization involved are known in advance to the analysis. This approach is very useful for performance evaluation of real-time control systems with hard deadline requirements. In probabilistic models, the task arrival rates and the task service time are usually specified by probabilistic distribution functions. Probabilistic models usually give a gross prediction on the performance of a system and are usually used for the early stages of system design when the system characteristics are not well understood. Another recently developed tool for modelling and simulating distributed control systems is the IEC 61499 standard [8, 9]. This standard defines a software paradigm built on the basis of FB (Function Block) that encapsulates data along with its behaviour in a form that is similar to physical entities, i.e. electronic circuits or hardware elements. In designing and implementing state machine control, Petri nets are suitable for defining the desired sequence of operations for the controlled machine or process, as well as possible abnormal sequences which may occur. The IEC 61499 standard may be used for informal modelling of the machine or process behaviour, using simulation models, or to define the appropriate state-machine controllers, typically as the ECCs (Execution Control Charts) and

algorithms of basic function block types. This chapter presents the proposed some PN based model for holonic systems and proposes an under work modelling application based on the IEC 61499 standard. Some examples will emphasize our approach. This approach is very useful for performance evaluation of real-time control systems with hard deadline requirements. In probabilistic models, the task arrival rates and the task service time are usually specified by probabilistic distribution functions. Probabilistic models usually give a gross prediction on the performance of a system and are usually used for the early stages of system design when the system characteristics are not well understood. We analyze the performance of holonic distributed systems and, in order to model clearly the synchronization involved in concurrent systems, the Petri net model is chosen. In this chapter we consider event graphs as Petri nets in which each place has one input transition and one output transition. It has been shown that distributed systems can be modeled as event graphs [8, 9]. When the manufacturing times are deterministic (respectively stochastic), the cycle time (respectively the mean cycle time) of the model is the period (respectively the mean period) required to manufacture a given set of parts which fits with the required ratios. The smaller the cycle time (respectively the mean cycle time) the higher the productivity of the system. When the firing times of transitions are deterministic, it is possible to define the cycle time of an elementary circuit. This is given by the ratio between the sum of the firing times associated with the transitions of the circuit and the number of the tokens in the places of the circuit, which is constant (we consider strongly connected graphs with the number of tokens in any elementary circuit, constant):

$$C_i = \frac{T_i}{N_i} \tag{1}$$

Where i = 1, 2, ... , n number of elementary circuits of the graph; Ci = cycle time of elementary circuits i; $T_i = \sum_i^n r_i$ = sum of the execution times of the transition in circuit i; $N_i = \sum_i^n M_i$ = total number of tokens in the places in circuit i.

In this case it has been proven that the cycle time of a strongly connected event graph is equal to the greatest cycle time of all elementary circuits. Furthermore, given a value C* greater than the largest firing time of all transitions, an algorithm has been proposed in [10] to reach a cycle time less than C*, while minimizing a linear combination of the token number in the places. The coefficients of the linear combination are the elements of a p-invariant. When the event graph is the model of a ratio-driven distributed system (such as manufacturing system), C* has to be greater than the largest cycle time of all command circuits [11]. A command circuit is an elementary circuit, which joins the transition that models the operations performed on the same machine [23 - 25]. Such a circuit contains one token to prevent more than one transition firing at any time in each elementary circuit. In other words, C* must be greater than the time required by the bottleneck machine to perform a sequence of parts which fits with the production ratios. In the case of random firing times, it is no longer suitable for the elementary circuits to evaluate the behaviour of the event graph in order to reach a given performance. Thus, the results presented in this chapter, which aim at reaching a given mean cycle time in a steady state while minimizing a linear combination of the place markings, are particularly important at the preliminary design level of manufacturing systems working on a ratio-driven basis. We also consider a few techniques for scheduling and verifying the

throughput of holonic cyclic manufacturing systems (HCMSs). A holonic cyclic production system manufactures a set of products at a constant frequency. We notice that any holonic cyclic production system can be modeled as an event graph, and therefore it is possible to drive the optimal throughput using the properties of the event graphs [12 - 14] by scheduling the tasks of the HCMSs in order to minimize the work in process. One may notice that nowadays often these production systems display a distributed configuration, e.g. they can be seen as distributed systems. Section 2 frames the optimum behaviour of the Petri net model of a holonic cyclic manufacturing system. In section 3 we give a sufficient condition for optimality, e.g. an algorithm that maximizes the throughput of scheduling production lines and we propose an algorithm to evaluate the bounds used to calculate the average cycle time which is exemplified on a kanban system in section 4. Section 5 introduces the new concept of transductional holonic formalisms, section 6 establishes the illustrative scenario, section 7 applies some transductional models to the given scenario, and section 8 concludes the chapter.

2 Framing the Mean Cycle Time

In this chapter we examine the holonic cyclic manufacturing systems (HCMSs), e.g. production tasks including assembly/disassembly or fork/join ones, with unreliable machines displayed in Jakson (infinite-capacity) networks and we consider the problem of maximizing the throughput by achieving a balance among the processing and repair of all machines under certain economical criteria. It has been proven [14] that in an event graph a marking belonging to the optimal solution under a periodic operational mode (POM) is an optimal solution under the earliest operational mode (EOM). So, we consider the earliest operational mode of the event graph, and we assume only non preemptive transition firings. We further assume that, when the transition fires, the related tokens remain in the input places until the firing process ends. They then disappear, and one new token appears in each output place of the transition. We use the following notations: Mi = the marking of the elementary circuits, $i \in N$; $X_t^k \in R^+$ = random variable generating the time required for the k^{th} firing of the transition t, $k \in N$; $I_t(n)$ = instant of the n^{th} firing initiation of the transition t; E = set of elementary circuits; s(e) = sum of the random variables generating the firing; $\sum_{t\in e} X_t^l$ = sum of times of the transitions belonging to e; E_t = set of elementary circuits containing the transition t. We assume that the sequences of transition firing times are independent sequences of integrable random variables. It was proven in [15] that there is a positive constant $s(M_0)$ so that:

$$lim_{n\to\infty}\frac{I_t(n)}{n} = C_m \tag{2}$$

Where: Cm = the average cycle time of the event graph.
Furthermore, we denote by m_t the mean value of X_t^k and by q_t the standard deviation of X_t^k , i.e., $m_t = F[X_t^k]$ and $q_t^2 = F[\left(X_t^k - m_t\right)^2]$.

2.1 The Lower Bound of the Mean Cycle Time

The cycle time of the deterministic problem obtained by replacing the random variables which generate the firing times by their mean values, is a lower bound of the mean cycle time [16]. The following relation proven in [17] provides a better lower bound for the value of the mean cycle time than the previous one:

$$C^{**} \geq max_{e \in E} \;\; F\left[max\left\{\frac{s\left\lfloor e \;\setminus\; \{t^*(e)\} + m_{t^*(e)}\right\rfloor}{M_0(e)}, m_{t^*(e)}\right\}\right] \tag{3}$$

Where $t^*(e)$ is a transition, belonging to event e, with the greatest average firing time, i.e., $m_{t^*(e)} = max_{t \in e} m_t$, and $s[e \setminus \{t^*(e)\}]$ is the sum of the firing times of the transitions belonging to e except $t^*(e)$.

2.2 The Upper Bound of the Mean Cycle Time

With M_0 being the initial marking, we derive a marking M_1 from M_0 by leaving the places which are empty in M_0, empty in M_1, and by reducing to one the number of tokens in the places containing more than one token in M_0. Thus, $M_1(p) \leq M_0(p)$ for any set of places of the strongly connected event graph. An earliest operation mode running with the initial marking M_1 leads to a greater mean cycle time than the one obtained when starting from M_0. Then, starting from M_1, we apply to the event graph the earliest operation mode, but we block the tokens as soon as they reach a place already marked in M_1. This operation mode is referred to as the constrained mode [18]. We denote by C* the mean cycle time obtained by using the constrained operation mode when M_1 is the initial marking. We know [8, 9] that C* is greater than the mean cycle time obtained by using the earliest operation mode starting from M_1 which, in turn, is greater than the mean cycle time obtained with the earliest operation mode when the initial marking is M_0. Thus, C* is an upper bound of the solution to our problem (i.e. the mean cycle time obtained starting from M_0 when using the earliest operation mode). The following relation defines this upper bound:

$$C^* = F\left[max_{z \in Z} \;\; s(z)\right] \tag{4}$$

Where, Z is the set of directed paths verifying the following properties:
- the origin and the extremity of any path is a marked place;
- there is no marked place between the origin and the extremity of the path.

3 The Evaluation of the Optimum Schedule of a HCMS

The problem is to determine the time at which each task should begin during the cycle. We assume that the set of tasks does not change, the tasks are not preemptive, and the task processing is deterministic. We determine two bounds for the mean cycle time of a Petri net model of a HCMS. The next step is to give an algorithm for optimizing the Petri net model in order to determine a feasible scheduling for the production cycle and

to minimize the work in process, e.g. the number of cycles necessary to complete one item of each product, given that schedule [19-22]. Algorithm:

1. The tasks of the HCMS that require resources are scheduled in any order. We notice by si and by ei the start, respectively the end time of task i. All tasks start in the interval [0, C*], and the resources can perform maximum one task at a time, where C* is given by relation (4).
2. For each product p = 1, 2, ... , n go to step 3 for the first task and go to step 4 for the rest of the tasks.
3. Let i be the first task in the sequence of tasks Tk , k = 1, 2, ..., m necessary to produce one unit of k. Assign the table Tk(i) = 0 for this task.
4. For each product label task $T_k(j)$ as follows: if $s_j > e_i$ then set $T_k(j) = T_k(j) + 1$, otherwise set $T_k(j) = T_k(i)$.
5. For each product calculate the production time as follows: $C_p = [T_p(l)T_p(f)] \cdot Cmed + s_1 - s_f$, where f is the first task, and l is the last one, in the sequences of tasks T_k, and $Cmed = (C* + C**)/2$, with C** and C* given by relation (3) , respectively relation (4).
6. For the entire production the total average work in process is $\sum_{p=1}^{n} \frac{C_p}{C_{med}}$, where p=1, 2, ... , n is the number of products in a complete production cycle.

In the reminder of the previous section, we compare the previous bounds with the existing ones. Under the assumption of non-preemptive transition firing, it was proven in [16] that:

$$C^{**} \geq max_{e \in E} \;\; F \left[max \left\{ \frac{s \lfloor e \;\setminus\; \{t^*(e)\} + m_{t^*(e)} \rfloor}{M_0(e)}, m_{t^*(e)} \right\} \right] = old \;\; lower \;\; bound \quad (5)$$

$$C^* \leq \sum_t m_t = old \;\; upper \;\; bound \quad (6)$$

The following relations show that the new bounds are better than the old ones. But how close are they to the optimal solution? In order to answer this question we give the next algorithm, inspired from the operational research area, for verifying the systems performance:

a) Express the token loading in $a(pxp)$ matrix P, where p is the number of places in the Petri net model of the system. Entry (A, B) in the matrix equals x if there are x tokens in place A and place A is connected directly to place B by a transition; otherwise (A, B) equals 0.

b) Express the transition time in a (p x p) matrix Q. The entry (A, B) in the matrix equals to the mean values of the random variables which generate the firing times (i.e., X_t^k) if A is an input place of the transition “i” and B is its output place. Entry (A, B) contains the symbol “w” if A and B are not connected directly as described above.

c) Compute matrix $Cmed \cdot P - Q(with\ p - w = \infty$, and $C_{med} = (C^* + C^{**})/2$, for $p \in N)$, than use Floyds algorithm to compute the shortest distance between every pair of nodes using matrix CP-Q as the distance matrix.

The result is stored in matrix S. There are three cases.

1) All diagonal entries of matrix S are positive $(i.e., C_{med} \cdot N_k T_k > 0$ for all circuits - see relation (1)) the system performance is higher than the given requirement;
2) Some diagonal entries of matrix S are zeros and the rest are positive $(i.e., C_{med} \cdot N_k T_k = 0$for some circuits, and $C_{med} \cdot N_k T_k > 0$ for the other circuits) - the system performance has just reached the given requirement;
3) Some diagonal entries of matrix S are negative $(i.e., C_{med} \cdot N_k T_k < 0$ for some circuits) - the system performance is lower than the given requirement.

In addition we may say that when a decision-free system runs at its highest speed, $C_{med} \cdot N_k$ equals to T_k for the bottleneck circuit. This implies that the places in the bottleneck circuit will have zero diagonal entries in matrix S. The system performance can be improved by reducing the execution times of some transitions in the circuit, or introducing more concurrency in the circuit (by modifying the initial marking), or increasing the mean cycle time (by choosing another average value for it).

4 The Evaluation of Kanban Systems

As we well know, an event graph can be used to model a kanban system. An example of a simple production line will be used to exemplify the above discussed problems. The production line consists of two machining tools (M1 and M2), two robot arms and two conveyors. Each machining tool is serviced by a dedicated robot arm, which performs load and unload tasks. One conveyor is used to transport work-pieces, a maximum of two at a time. The other conveyor is used to transport empty pallets. There are three pallets available in the system. Each work-piece is processed on M1 and M2, in this order. The stochastic timed Petri net model of this system is shown in Fig.1. The initial marking of the net is $(300012111)^T$. When the time delays are modeled as random variables, it has become a convention to associate the time delays with the transition only. The involved transition has the associated time delays expressed in time units. The random variables X_1, X_2, X_3, X_4 are assigned to the transitions t_1, t_2, t_3, t_4, respectively. X_1 is uniformly distributed on [0.2], X_2 and X_3 are random variables with $F[X_2] = 11$t.u. and $F[X_3] = 1$ t.u. X_4 is a constant and equals to 17 t.u. The Petri net model contains four loops.

The time delays associated with these loops, as well as their token contents are:

1) loop: $t_1 p_2 t_2 p - 3 t_3 p_4 t_4 p_1 t_1$, loop delay: 30 u.t., token sum: 3, cycle time: 10 u.t.
2) loop: $t_1 p_2 t_2 p_5 (orp_8) t_1$, loop delay: 12 u.t., token sum: 1, cycle time: 12 u.t.
3) loop: $t_2 p_3 t_3 p_6 t_2$, loop delay: 2 u.t., token sum: 2, cycle time: 1 u.t.
4) l oop: $t_3 p_4 t_4 p_7 (orp_9) t_3$, loop delay: 18 u.t., token sum: 1, cycle time: 18 u.t.

Thus, the minimum cycle time is 18 time units. This means that it takes a minimum of 18 time units to transform a raw workpiece into a final product. Computing the lower-bound and the upper bound of the cycle time of the event graph from Fig.1. by using the relations (3) and (4) , we obtain the values: C** 12 u.t.; C* 18 u.t.

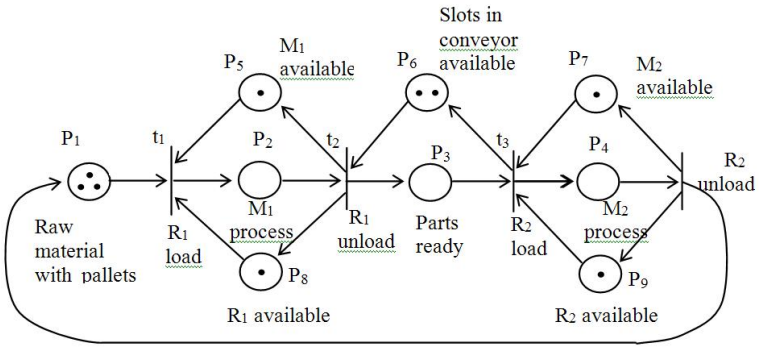

Fig. 1. Stochastic Petri net model for a manufacturing system

5 Transductional Holonic Formalisms

Holonic manufacturing systems (HMS) represent a novel paradigm based on concepts developed by Arthur Koestler to living organisms and social organizations [1]. The basic elements of a HMS are called *holons*, a term coined from the Greek words ”holos” (the whole) and ”on” (a particle). This combination reflects the fact that a holon may act both as an autonomous entity (the whole attribute) and as a part of a larger whole (the particle attribute) As it is shown in [2] at least three separate information processing levels can be distinguished in a factory entirely based on holonic principles:

1) *real-time control*, tightly connected with the physical level of manufacturing equipment;
2) *production planning and scheduling*;
3) *supply chain management*, integrating the particular plant with its external entities (suppliers, customers, partners, etc.).

Based on these levels, there are several architectures proposed for holons that combine both real-time control technologies (like the IEC 61499 standard [12]) and Multi-Agent Systems (MAS) technologies (for reasoning activities and high-level communication). Such an architecture is presented in Fig.2. In a holon equipped with both a lower level real-time component and a higher-level intelligent component, three communication channels should be considered [3]:

1) *intra-holon* communication between the function block part and the intelligent component;
2) *inter-holon* communication that is aimed at communication between the intelligent components of multiple holons;
3) a *direct* communication channel between real-time components of neighboring holons.

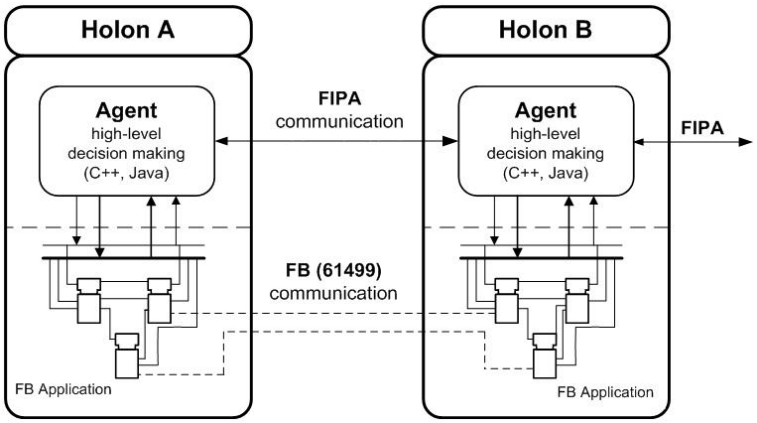

Fig. 2. Holonic architecture combining both real-time and MAS technologies (from [2])

Our chapter is focused on some aspects of formal modelling and verification of low-level interaction between holons, taking as a practical example the control and communication mechanism between three production facilities sharing a common resource. This work is inspired from [4], where the authors introduced a Petri net based approach for defining robust and reliable communication between the high and low-level sub-holons within a device holon. In this chapter we developed the formal modelling example by means of two kinds of Petri nets: Place/Transition Petri net, and Net Condition/Event Systems (NCESs), as a more effective modelling method.

5.1 Net Condition/Event Systems

For computer science *formal verification* provides the correctness of programs by using mathematical methods and models. Unlike testing via simulation the formal verification can explore complete set of systems state space and mathematically prove that no undesirable or dangerous behaviour occurs [6]. Formal verification is applied to a previously built model using an appropriate finite-state or hybrid formalism, e.g. finite-state machines, Petri nets, etc. Petri nets are widely used in formal modelling as a method for synthesizing and specifying control programs [4]. There are many different kinds of Petri nets, each one having its advantages and limitations and being more appropriate for a specific domain or application: Elementary Net systems, Place/Transition systems, Net Condition/Event systems (NCESs) [6, 7], etc. For our work we have chosen as main method the Net Condition/Event systems, given the similarities between these systems and the IEC 61499 standard. The formalism of Net Condition/Event systems was introduced by Rausch and Hanisch [10] as a modular extension of Signal-net systems (SNSs), informally described in the next section.

5.2 Signal-Net Systems

This section briefly gives some informal definition of Signal-net systems, according to [6]. More formal definition may be found in [11]. A Signal-net system is a place/transition model similar to Petri nets. Basic artifacts of the place/transition models are: *places*, which bear *tokens; (net) transitions*, and *arcs* connecting places with transitions and transitions with places, known as *token flow arcs*. SNS in addition have two types of arcs: *event arcs* from transitions to transitions and *condition arcs* from places to transitions.

Semantics The semantics of Signal-net systems is defined by the firing rules of net transitions. There are several conditions to be fulfilled to enable a net transition to fire. First, as in the ordinary Petri nets, an enabled transition has to have a token concession, i.e. all the flow arcs from its pre-places have to be enabled. A flow arc is enabled when the token number in its source place is not less than its weight. In addition to the flow arcs from places, transitions may have incoming condition arcs from places and event arcs from other transitions. A transition is enabled by condition signals if all source places of the condition signals are marked by at least one token. With respect to incoming event arcs a transition can have either OR or AND mode (event signal sensitivity mode). In the OR mode to fire a transition it is necessary that at least one event arc lead a signal-event whilst in the AND mode all event arcs must lead a signal-event. Several SNS transitions can fire simultaneously. A set of simultaneously firing net transitions is called *step*. A step is formed by first picking up a nonempty subset of enabled spontaneous transitions, and then by adding as many as possible enabled transitions which are forced to fire by event signals produced by the transitions already included in the step.

Discrete timing The concept of discrete timing is applied to the SNS as follows: to every pre-arc [*p, t*] of the transition t we attach an interval [*l, h*] of natural numbers with $0 < l < h < .$ The interval is also referred to as *permeability* interval. The interpretation is as follows: every place p bears a clock u(p) which is running if the place is marked $(m(p) > 0)$, and is switched off otherwise. The clocks run at the same speed measuring the time the token status of its place has not been changed. If a firing transition t removes a token from the place p or adds a token to p, the clock of p is turned back to 0. A (marking-enabled) transition t is *time-enabled* only if for any pre-place p of t, the clock at place p shows a time *u(p)* such that $l(p,t) < u(p) < h(p,t)$.

5.3 Net Condition/Event Systems NCES

The general idea of Net Condition/Event formalism is to design a system as a set of modules with a particular dynamic behaviour and their interconnections via signals [6]. Once designed, the modules can be re-used over and over again. Each module has inputs and outputs of two types: (1) condition inputs/outputs carrying information on marking of places in other modules, and (2) event inputs/outputs carrying information on firing transitions in other modules. Condition input signals as well as event

input signals are connected with transitions inside the module. The NCES concept provides a basis for a compositional approach to build larger models from smaller components. The ”composition” is performed by connecting the inputs of one module to the outputs of another module, as shown in Fig. 3. The result of the composition of two NCES,$N_1 and N_2, is an NCES N_{1+2}$ obtained as a union of the components. If such a module has no external inputs then it represents a Signal-net system (SNS). NCES can

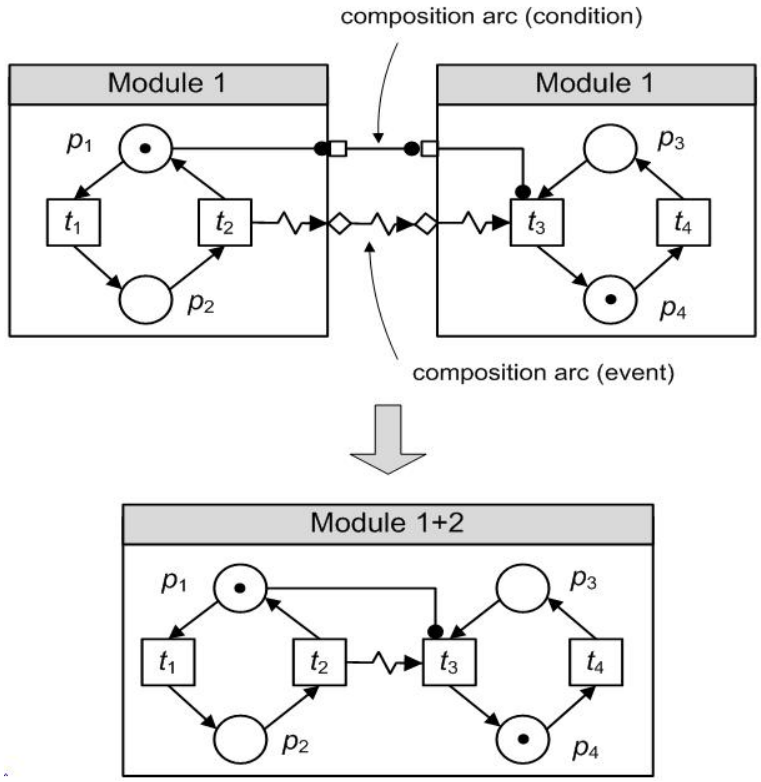

Fig. 3. Example of modular composition (from [9])

precisely follow the structure of popular block diagram modelling and implementation languages, such as state flow of MATLAB/Simulink and the function blocks of the IEC 61499 standard [9].

6 The Working Scenario

As we already have mentioned in the Introduction, our chapter is focused on an example of defining a formal model for a control and communication mechanism which has to

assure the exclusive access of three production facilities to a shared resource. The considered example is built around a Bosch Rexroth conveyor with three flows, presented in Fig. 4a. Along the conveyor belts three production facilities, called as *A, B* and *C* are considered to be placed, as is illustrated in Fig. 4b. Each production facility is placed

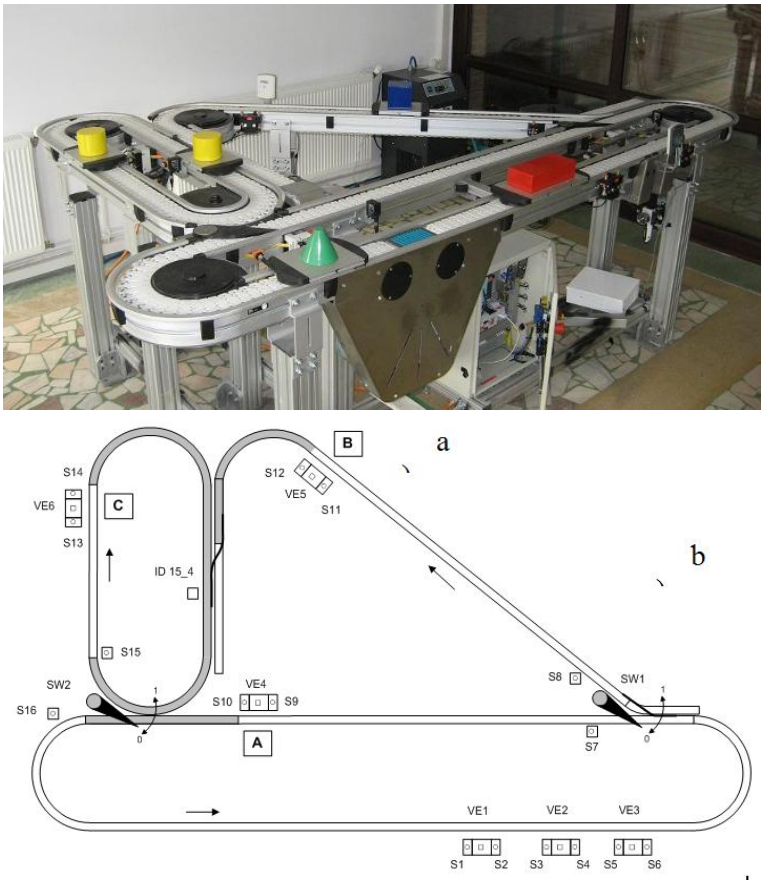

Fig. 4. a) Bosch Rexroth conveyor with three flows; b) Schematic of the system

alongside a *stop gate* with two proximity sensors. Considering the case of machine *A*, it is placed alongside the stop gate VE_4, which have attached the proximity sensors S_9 and S_{10}. A working scenario for the machine A may by as follows: when a pallet arrives, it is stopped by the gate VE_4 and its presence is detected by the sensor S_9. Now the production facility can execute some operations to the components transported on the pallet. After finishing its work, the production facility *A* will set free the pallet by opening the gate. The departure of the sensor is confirmed by the activation of the sensor S_{10}. Its activation will cause the gate to go back in the blocked state in order to stop the next ar-

riving pallet. When a pallet leaves a production facility it enters in a shared route area, marked in the Fig. 4b with the gray color. The critical route area is considered to be formed by the routes between each production facility (*A, B* and *C*) and the proximity sensors S_{15} and S_{16}. In order to avoid the collision of the pallets released by production facilities, their access in the shared area must be controlled such as one pallet enters at a time. Once this pallet leaves the critical area, another pallet can be released by a production facility. As a result, a mechanism for communi-cation and control must be developed, which has to assure the correct access of production facilities to the shared route area. The next sections will present some formal models for such a mechanism.

7 Transductional Formal Models

7.1 Formal Model Based on Place/Transi-tion Systems

Based on the concepts presented in [4] a model for the communication and control mechanism was firstly developed, using as formal method a classical form of Petri nets: Place/Transition systems. This model is depicted in Fig. 5 and the notations for the PNs are mentioned in Table 1. The control mechanism is consisting of four distributed software components: three of these are considered to be placed into the production facilities controllers, and the fourth, having the role of a supervisor, may be located into a different embedded device.

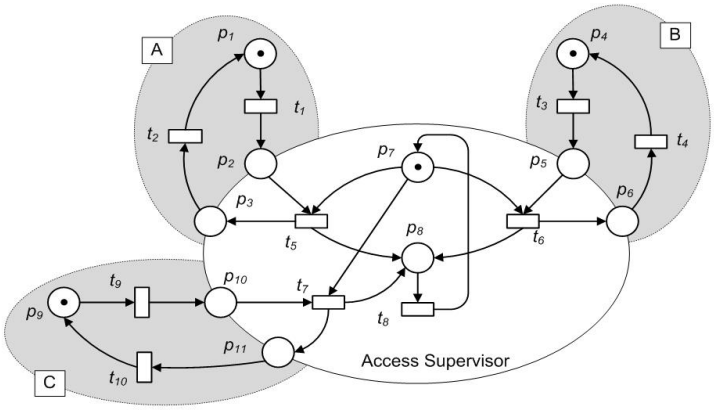

Fig. 5. Model of the access control mechanism based on P/T systems

Places/transitions	Description
p_2, p_5, p_{10}	Shared places having the meaning of messages passed from production facilities A, B and C to the supervisory component. Through these messages the production facilities may request access to the shared route.
p_3, p_6, p_{11}	Shared places having the meaning of messages passed from the supervisory module to the production facilities A, B and C respectively. Through these messages the machines are granted with access to the shared route in order to deliver one pallet.
t_1, t_3, t_9	Transitions meaning that a new pallet has became ready to be delivered. As a consequence, the corresponding production facility has to send a message to the supervisor component in order to get access to the shared route.
t_5, t_6, t_7	Transitions through that the supervisor component grants the route access to a certain production facility.
t_2, t_4, t_{10}	Transitions inside the models of production facilities, signifying the action of releasing a ready pallet.
p_8	After granting the route access to a production facility, the supervisor component will wait in this state for the shared route to become free. This will happen when the pallet occupying the route will pass one of the proximity sensors S_{15} and S_{16}.
t_8	Transition meaning that the shared route has become free.

Table 1. The notations for the PNs depicted in Fig. 5

7.2 Formal Model Based on NCES

A more effective and intuitive way for modelling the above presented mechanism is to use the NCES models. The condition and event signals, available in NCES allow a more clear specification of the messages passed between modules. Also, the modularity of NCES improves the models development process due to the possibility of reusing previously defined models. In Fig.6 is depicted the NCES version of the communication and control mechanism.

The components implied in the control mechanism are modelled as modules, called as *AccessCtrl A, AccessCtrl B, AccessCtrl C* and *Access Supervisor*. The first three modules, considered to be located in the production facilities controllers, are identical. Using the NCES tools one can build a single model of this type, save it into a library and then use its instance as often as is needed. The modules have interfaces for signal inputs and outputs, which allow them to be connected with each other. Also, these signals enhance the clarity in modelling of modules behaviour. For example, considering the case of *AccessCtrl A* module, the condition input *CmpFin* has the role to inform the module whether there is or there is not a pallet waiting to be released (i.e. to inform the module if the production facility A has finished its operation to the current pallet). This signal serves as incoming condition for the transition t_1. In other words the transition t_1 can be enabled only if there is a token load in the place p_1 (the modelled software component is not busy) and the *CmpFin* signal is true. In this case the transition fires (it is a spontaneous transition) and the token pass form place p_1 to place p_2. As a result, the condition output *AccessRQ* will become true, informing the supervisory module that the production facility A needs access to the shared route.

The *AccessRQ* output of the *AccessCtrl A* module serves as incoming condition for the

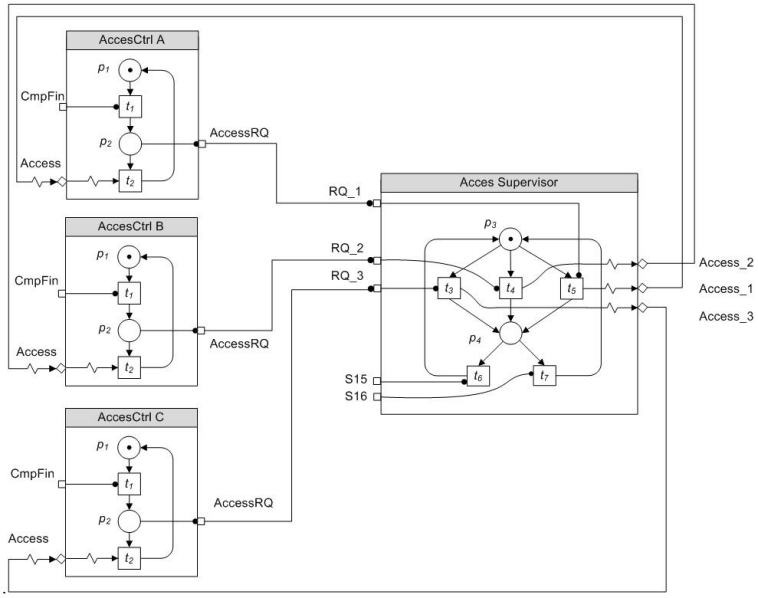

Fig. 6. Formal model of the control mechanism based on NCES

transition t_5 inside the *Access Supervisor* module. That means that the transition t_5 may become enabled only if the place p_3 contains a token (the software component is not busy) and the *AccessRQ* signal is true. Firing of the transition t_5 leads to the generation of the event *Access_1*, and the token from the place p_3 will pass to the place p_4. The event *Access_1* serves as incoming event for the transition t_2 of the module *AccessCtrl A* and may be interpreted as a message through which the supervisory module grants the route access to the production facility *A*. The event forces the firing of transition t_2, which may be seen as an action through that the production facility *A* release the waiting pallet. After the firing of transition t_2 the condition output *AccessRQ* becomes false (the token is no longer in p_2) and the module enters in a waiting state for a new pallet to be ready (now the token is in the place p_1). The conflict generated by the simultaneous activation of more than one transition from the set $\{t_5, t_6, t_7\}$ is assumed to be solved by an internal algorithm of the supervisory module. Finally, the place p_4 represents the state where the supervisory module waits for the shared route to become free. Outgoing from this state is possible by firing of transitions t_6 or t_7, i.e. through the activation of one of the proximity sensors S_{15} and S_{16}.

7.3 The Autonomous NCES Model

In order to analyze an NCES module it has to be an autonomous one, i.e. to have no external inputs. Such a module represents a Signal-net system. It may be seen that the model depicted in Fig. 6 is not autonomous because it has five not assigned inputs, subject of external signals. Therefore we added to our model four modules (*Sim_A, Sim_B, Sim_C* and *CnvModel*), such as the resulted model do not need any external signal (see Fig. 7). The modules of type *Sim_X* simulate in a simple manner the activity

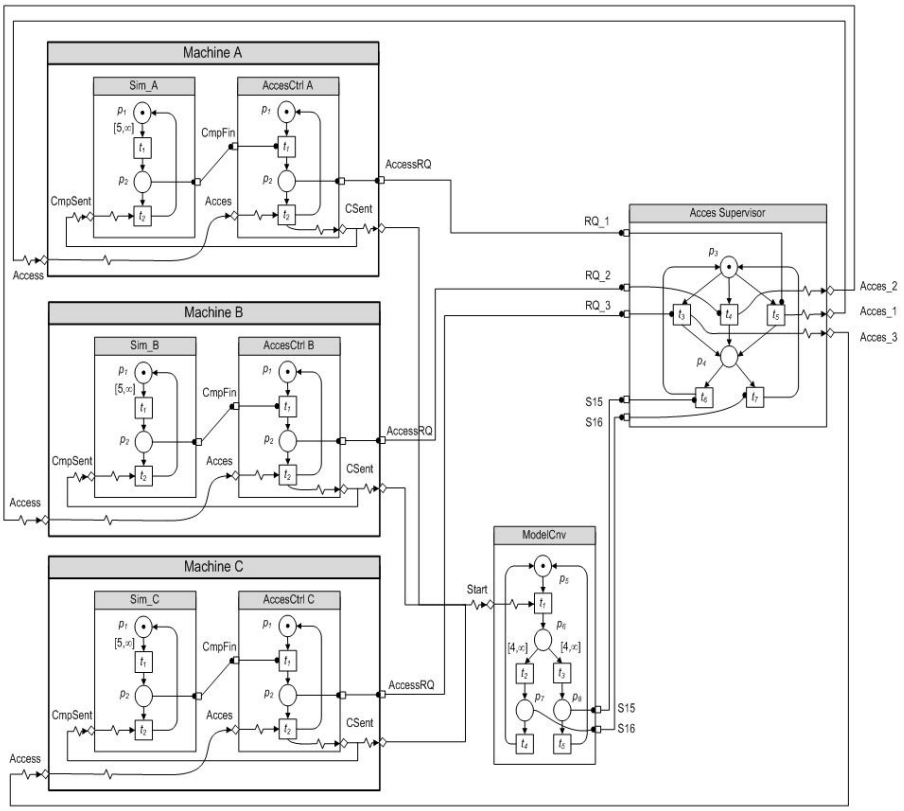

Fig. 7. The autonomous model of the control mechanism

of the production facilities *A, B* and *C*, considering the processing time corresponding to each machine being 5, 4 and 3 time units respectively. A pair of modules *Sim_X* and *AccesCtrl X* are grouped and encapsulated in a new module having the type *Machine X*. The module *CnvMode* is designed to model the conveyor behaviour since a production facility releases a pallet until it leaves the shared route (moment signalized through the activation of one of the condition output $S_{15}$and S_{16}). The time for transporting a pallet

from any production facility to the proximity sensors S_{15} orS_{16} is considered to be 4 time units.

8 Conclusion

This work establishes a necessary relation between diagnosis and performances of holonic distributed systems, such as discrete event manufacturing systems. We have proposed a new architecture for the diagnosis and performance evaluation of holonic distributed systems using Petri nets formalisms and their properties [19, 20]. The proposed approach can be applied to the modelling and analysis of manufacturing systems, supply chains, job scheduling in a chain management system, such as flexible manufacturing systems, etc. The novelty of the approach is that the construction of large Petri nets is not required. Formal methods are widely used in modelling of distributed and concurrent systems in order to assure the correct system behaviour by means of formal verification techniques. This chapter has exemplified the use of Petri net based formal methods for modelling the low-level interaction between holons (in our example represented by the production facilities and their associated intelligence). It was shown that NCESs represent an appropriate modelling solution for this problem due to their particular characteristics: modularity and synchronization of transitions by means of signals. An important result in this chapter is that it is always possible to reach a mean cycle time as close as possible to the greatest mean firing time using a finite marking, assuming that a transition cannot be fired by more than one token at each time. This result holds for any distribution of the transition firing time. An algorithm for verifying the distributed systems performance was introduced. An approach for computing the upper and lower bounds of the performance of a conservative general system is mentioned. However, the bounds produced may be loose. Further research will focus on the conditions under which a mean cycle time can be reached with a finite marking.

References

1. Brusey, J., McFarlane, D. Designing Communication Protocols for Holonic Control Devices using Elementary Nets.*HoloMAS 2005*, LNAI 3593, pp.76-86, 2005, Springer Verlag Berlin Heidelberg, (2005)
2. Peters, R., Tobben, H. A Reference Model for Holonic Supply Chain Management. *HoloMAS 2005*, LNAI 3593, pp221-232, (2005)
3. Bussman, S., McFarlane, D.C. Rationales for Holonic Manufacturing Control, Proc. of the 2nd Int. Workshop on Intelligent Manufacturing Systems.*Leuven*, pp.177-184, (1999)
4. Koestler, A. Jenseits von Atomismus und Holismus der Begriff des Holons. *Wien*, (1970)
5. Delfmann, W., Albers, S. Supply Chain Management in the Global Context. *Koln*, (2000)
6. Lackmann, F., Nayabi, K., Hieber, R. Supply Chain Management Software, Planungssysteme im Uberblick, Germany, (2003)
7. Bacelli, F., Lin, Z. Compresion properties of stochastic decision free Petri nets.*IEEE Trans. Autom. Contr.*, vol 37, no. 12, pp. 1905-1920, (1992)
8. Laftit, S., Proth, J. M., Xie, X.L.: Optimization of invariant criteria for event graphs, IEEE Trans. Autom. Contr., vol 37, no. 12, pp. 547-555, (1992)

9. Proth, J. M., Xie, X.L.: Cycle time of stochastic event graphs: evaluation and marking optimization, IEEE Trans. Autom. Contr., vol 39, no. 7, pp. 1482-1486, (1994)
10. Campos, J., Chiola, G., Silva, M.: Properties and performance bounds for closed free choice synchronized monoclass quering networks, IEEE Trans. Autom. Contr., vol 36, no. 12, pp. 1368-1382, (1991)
11. Chauvert, F., Herrmann, J.W., Proth, J.M.: Optimization of Cyclic Production Sustems: A Heuristic Approach, IEEE Trans. On Robot. And Automat., vol.19, no.1, pp.15-154, (2003)
12. Proth, J.M., Xie, X.L.: Petri Nets. A Tool for Design and Management of Manufacturing Systems, New York: Wiley, (1996)
13. Minis, I., Proth, J.M.: Production management in a Petri net environment, Recherche Operationelle, vol.29, no.3, pp.321-352, (1995) Zurawski, R., Zhon, M. C.: Petri nets and industrial applications: a tutorial, IEEE Trans. Ind. Electr., vol 41, no. 6, pp. 567-583, (1994)
14. Zurawski, R., Zhon, M. C.: Petri nets and industrial applications: a tutorial, IEEE Trans. Ind. Electr., vol 41, no. 6, pp. 567-583, (1994)
15. Singh, J.: Operations Research, Penguin Books, Midlessex, (1979)
16. Ciufudean, C., Popescu., D.: Scheduling Diagnosis of Flexible Manufacturing Systems, In Proc. Int. Conf. on Autom, Contr. And Syst. Eng., Cairo, (2005)
17. Ciufudean, C., Filote, C., Graur, A., Turcu, C.: Diagnosis of Complex Systems Using Ant Decision Petri Nets, The 1st Int. Conf. on Avail., Reliab. and Security, Vienna, pp.473-482, (2006)
18. Ciufudean, C., Filote, C.: Performance Evaluation of Distributed Systems, International Conference on Control and Automation, Budapest, pp.21-25, (2005)
19. Yalcin, A., Boucher, T.O.: Deadlock Avoidance in Flexible Manufacturing Systems using Finite Automata, IEEE Trans. on Robot. And Automat., vol.16, no.4, pp.424-429, (2000)
20. Kouikoglou, V.: Optimal Rate Allocation in Unreliable Assembly/Disassembly Production Networks with Blocking, IEEE Trans. on Robot. And Automat., vol.16, no.4, pp.429-434, (2000)
21. Ciufudean, C., Satco, B., Filote, C.: Reliability Markov Chains for Security Data Transmitter Analysis, The 2nd Int. Conf. on Avail., Reliab. and Security, Vienna, pp.886-892, (2007)
22. Recalde, L., Teruel, E., Silva, M.: Modeling and analysis of sequential processes that cooperate through buffers, IEEE Trans. on Rob. and Autom., vol. 14, no. 2, pp. 267-277, (1998)
23. Zuo, M.J., Liu, B., Murthy, D.N.P.: Replacement-repair policy for multi-state deteriorating products under warranty, European Journal of Operational Research, no. 123, pp. 519-530, (2000)
24. Hopkins, M.: Strategies for determining causes of events, Technical Report R-306, UCLA Cognitive Systems Laboratory, (2002)
25. Hall, K.H., Staron, R.J., Vrba, P.: Experience with Holonic and Agent-Based Control Systems and Their Adoption by Industry. In Mak et al. pp. 1-10, (2005)