32nd European Modeling & Simulation Symposium
17th International Multidisciplinary Modeling & Simulation Multiconference

ISSN 2724-0029 ISBN 978-88-85741-44-7 © 2020 The Authors.
DOI: 10.46354/i3m.2020.emss.053

Modeling of discrete controllers for Smart Manufacturing Systems

Armand Toguyéni[1,*]

[1]Centrale Lille Institut, CRIStAL, UMR 9189 F-59650 Villeneuve-d'Ascq, France

[*]Corresponding author. Email address: armand.toguyeni@centralelille.fr

Abstract

Smart Manufacturing Systems inherit the work done for three decades on Flexible Manufacturing Systems (FMS) and Reconfigurable Manufacturing Systems (RMS). Their control is characterized by many residual indeterminism that must be resolved in real time by smart functions that take into account the state of the system and the production objectives. Inadequate decisions can lead to system blockages or even dangerous situations. This study focuses on the design of control part which is seen as a hierarchy of communicating controller layers. The study focuses in particular on the piloting layers, the implementation of the extended operating sequences and the operation of the transport system. The different layers are modeled using Colored Petri Nets (CPN). They are structured in communicating CPN processes.

Keywords: Smart Manufacturing System, Discrete Event Systems, Colored Petri Nets, controllers, modelling

1. Introduction

Smart Manufacturing Systems (SMSs) are production systems that are agile enough to adapt their production to the needs of each customer (Mittal et al., 2019). In fact, they are designed for individualized mass production. This is completely contradictory in that for decades mass production and individualization of production were opposite concepts in terms of production. In the past, individualized production was only possible in Flexible Manufacturing Systems (FMSs). Smart Manufacturing Systems therefore inherited FMS.

Another aspect of SMSs is their ability to adapt to changing production environments (Hozdić, 2015). For example, they must adapt to failures. Indeed, they incorporate intelligent control functions that allow decisions to be made based on the state of the system and production objectives.

This study concerns the implementation of SMS control which corresponds to level 2 of the ISA-95 reference model ((www.isa-95.com): the Manufacturing Control System (Sprock and McGinnis, 2015).

In this paper, we propose a modular and hierarchical structuring of the control that allows to implement an intelligent control while limiting combinatorial explosion problems. In particular, this work proposes an operating sequence model that allows to manufacture on the system any type of parts whose manufacturing range uses existing machine manufacturing operations. It also offers solutions for dynamic routing of the parts in the system according to their manufacturing operations and the state of the system.

The paper is structured as it follows. In the second section, we will present in a synthetic way our method for designing the sequential control of SMS. In the following sections we will present the different models of the control. Thus the third section will be devoted to the presentation of the Transport System Coordination

Graph (TSCG). In the fourth section we will present a generic model, of a non-combinatorial Extended Operating Sequence. The fifth section will propose a production machine allocator model. We will end with a conclusion and the perspectives of this work.

2. State-of-the-art

At present, there is no consensus definition of SMSs. According to (Qu et al., 2019), SMSs can be defined from 3 points of view: engineering, communication technologies, predictive analysis capabilities and decision making. This study focuses more precisely on the engineering aspects.

From an engineering point of view, an SMS "is an intensified application of advanced intelligence systems which enable the rapid manufacturing of new products, dynamic response to product demand, and real-time optimization of manufacturing production and supply chain networks ..." (Qu et al., 2019).

For that, SMSs must be agile and reconfigurable. To meet these challenges, two key concepts must be considered in their design: flexibility and dynamic re-configuration (Radziwon et al., 2014). Several types of flexibility can be distinguished, such as product flexibility, operating sequence flexibility, transport flexibility, etc. (Beach et al. 2000). The reconfiguration of a production system is necessary when some of the resources of the system are changed. This modification can be permanent (in the case of the addition of a new resource) or temporary (in the case of a resource failure). A resource failure is an unforeseen event that must be dealt with online, both for maintaining products that are already in the system and for products that are to be loaded into the system. The concept of reconfiguration was formalized a few years ago to define a new class of production systems called Reconfigurable Production System or RMS (Koren 2014; Koren and Shpitalni 2010).

This study concerns the design of the operational control of SMSs. According to (Sprock and McGinnis, 2015), this control corresponds to level 2 of the control architecture which is implemented by PLCs and SCADA software. From the point of view of this layer, the system is a Discrete Event System (DES). The objective of this study is therefore to develop discrete controllers that allow to realize an intelligent control of the system. This requires to model all the potential evolutions of the system due to its flexibility. But the difficulty is that in this case one is confronted with the combinatorial explosion induced by all the flexibilities.

3. Control design method

3.1. Synthetic presentation of the design process

Our design process is based on a decoupling between product specification constraints and resource specification constraints. Therefore, it comprises two design flows: a product-centric flow and a resource-centric flow (Toguyeni 2018).

This study focuses on the construction of the final models built by the product flow and the transport resources. Given the distribution of these models on separate computers communicating through industrial computer networks and industrial messaging, a client/server approach is used to take into account the asynchronous constraint induced by this environment. Since our approach is based on Petri Nets (PN hereafter, in our models, this will result in the use of pairs of semaphore places (Request/Acknowledge) to synchronize distributed PN processes (Figure 1).

The main final models are the Extended Operating Sequence (EOS), the Transport System Coordination Graph (TSCG) and the resource allocators. An EOS models the different operations applied to a product so that it goes from its raw state to its finished state. The TSCG manages transport resources in order to transfer the products between different workstations. Resource allocators make it possible to assign resources to products according to requirements.

3.2. Basic Principles of Modelling

The guiding idea of modeling is to limit the combinatorial explosion while having models that reflect the structure of the system. To limit the combinatorial explosion, we have chosen Petri Nets (PN) as modeling tools. The interest of PNs is that they are adapted to the modelling of DES characterized by a strong parallelism. Since Smart Manufacturing Systems are made up of different transformation or transport resources, each with its own computer, our modeling approach is adapted to the asynchronous of the system. Thus, each resource is modeled as a PN process synchronizing with other processes by using pairs of semaphores (Figure 1).

The second main idea of this work is to decouple the problems. This decoupling leads controllers to be functionally specialized for specific tasks. When they need the services rendered by other control functions, they use a request/acknowledge mechanism to request a service as a client and for the controller acting as server to respond with an acknowledgement and, if necessary, data transmission (Figure 1).

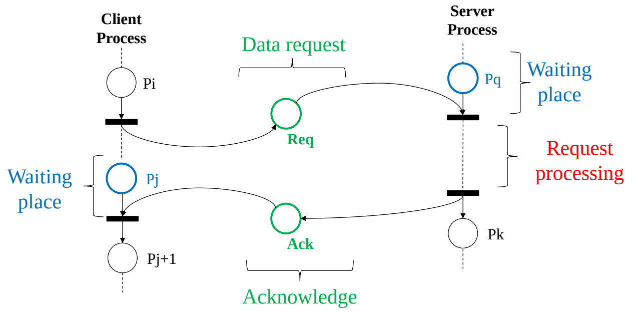

Figure 1. Pair of semaphores for synchronization between client and server processes.

3.3. Illustrative case study

As an example, Figure 2 describes an SMS that will serve as an illustrative case study throughout this study. Each machine implements different types of machining operations denoted "f_i". Operation "f_3" is thus performed by all machines. Operation "f_4", on the other hand, can only be performed by the M_4 machine.

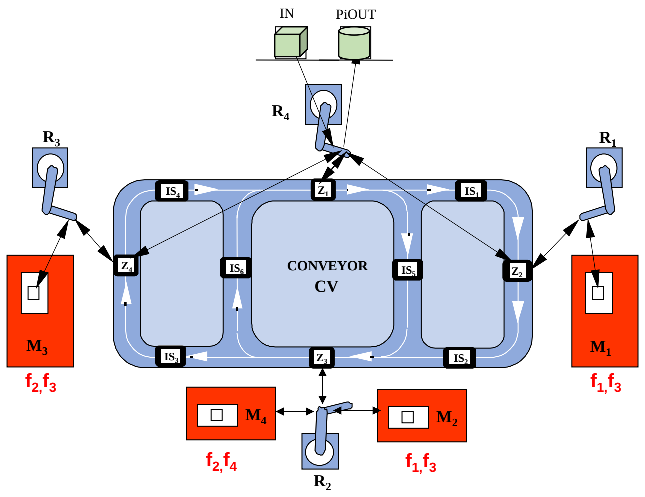

Figure 2. Example of Smart Manufacturing System

The machines in this system are served by a conveyor belt. On this conveyor are positioned work stations (Z_i) allowing the robots to palletize or de-palletize the parts loaded on pallets in order to be transported by the conveyor to a machine. The reachability relationships of the conveyor system are illustrated in Figure 2 by oriented arcs showing the direction of the movement of parts and pallets. To move from Z_1 to Z_3, a part has two possibilities: either be routed through the IS_1-Z_2-IS_2 route, or use directly the IS_5 route. These alternatives illustrate the flexibility of routing. Similarly, robot R_4 can transfer a part directly from the IN stock to the Z_2 loading station of machine M_1. In normal operation, from IN, the infeed stock, it loads the parts onto the pallets blocked at the Z1 station. Likewise, it can unload the finished parts to the outgoing stock PiOUT. IS_i ($i \in \{1,2,3,4,5,6\}$) and Z_i ($i \in \{1,2,3,4\}$) are temporary storage areas of the conveyor. This study assumes that the capacity of each Z_i is one pallet and that the capacity of each IS_i zones is 5 pallets.

3.4. Presentation of the modelling tool

For the modeling of our discrete controllers, we chose Jensen's Colored Petri Nets (CPN) (Jensen and Kristensen, 2015). They are more than just Colored PNs. They effectively incorporate features of Predicate/Transition PNs. In CPN, each place must be typed to define the type of tokens allowed. One can associate to arcs expressions or even functions written in the ML language. One can also associate to the transitions, Boolean expressions called guards, which must be true for a validated transition to be passed. On the other hand, CPN syntax gives the possibility to link different models by merging places (use of merge tags). Figure 3 gives a synthesis of the syntax of a CPN model. This syntax and semantics will be precise in the following sections during the presentation of the different models.

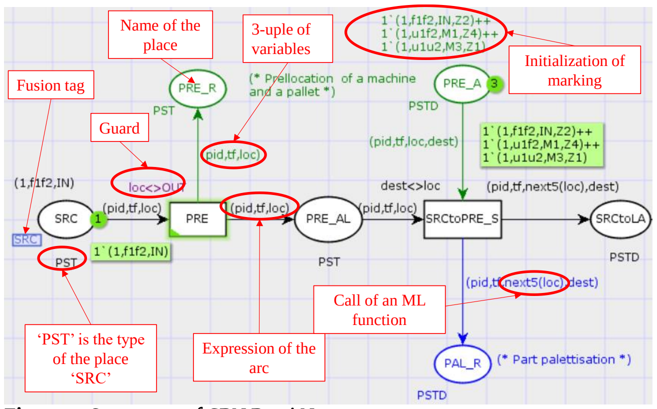

Figure 3. Summary of CPN Petri Net syntax

In Figure 3, all red annotations have been added to present the syntax of a CPN model.

4. Transport System Coordination Graph

The transport system is one of the essential components of a Smart System. Indeed, it must be flexible enough to allow indirect reachability between all the workstations of the production system. The problem is therefore not to model explicitly the combinatorics relating to routing flexibility. For this purpose, all the controllers are designed based on the datagram technique for product routing. The datagram is a technique for packet routing in computer networks. In our approach, it consists in routing each product in the production system according to two parameters: its final destination and the state of the machining or transport resources.

The final destination of a part is defined by the resource allocators of the control function. To explain the principle, one considers Figure 4.

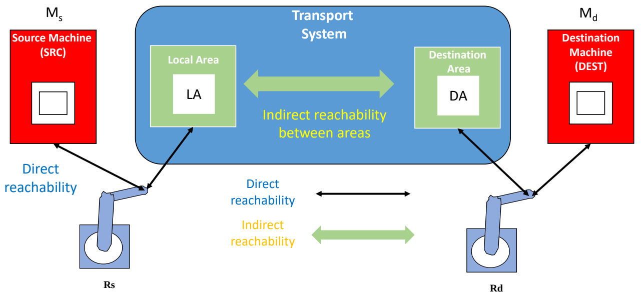

Figure 4. Generic diagram for the transport of a product

The transport of a product from a source location to a destination location is based on the reachability relationships between the different workstations of the production system. If one considers the system in

Figure 2, Z2 and M1 are in an external reachability relationship because it is necessary to use robot R1 to transfer parts between these two locations: Z2 belongs to the conveyor and M1 represents a machine. Conversely, there is an indirect internal reachability relationship between Z1 and Z3 because they both belong to the conveyor (Z1 and IS5 on the one hand and IS5 and Z3 on the other hand are in a direct reachability relationship).

Given a product located at a starting location such as the Ms machine in Figure 4, the control function must first define its final destination according to the next operation to be performed from the point of view of its machining range. Taking into account the state of the transformation resources, the control will allocate the Md machine to perform this transformation operation. The TSCG will then be in charge of managing the transport of the part to Md. The route is defined dynamically according to the state of the transport resources (breakdowns, saturation of intermediate stocks).

To build the TSCG of any system, we have defined generic primitives corresponding to the functions of each workstation. We also define different types of transfer between stations in direct accessibility.

4.1. Generic model of a workstation

Any workstation can be abstracted as a processing area and an internal stock managed as a FIFO (Figure 5). A system can be abstracted as composed of workstations operating according to the producer and consumer principle. This is because, from the point of view of products transportation, an upstream station can be seen as the producer of this product and the downstream station as the consumer of this product. In fact, in relation to the entire production process for the individual parts being manufactured, a station can be either a producer or a consumer.

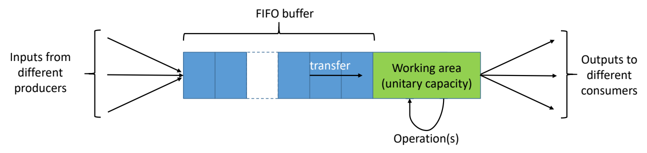
FIFO buffer
Inputs from different producers
transfer
Working area (unitary capacity)
Outputs to different consumers
Operation(s)

Figure 5. Abstraction of a workstation

This leads us to represent each workstation controller by a PN process that describes its different functionalities. This process has a pair of places named PROD/CONS which describes its storage capacities. Indeed, the PROD place models the parts present in the internal stock of the station while the CONS place models the residual capacity of the machine. When the machine is empty, the CONS slot contains as many tokens as the storage capacity of the station. This pair of slots will allow to condition the part transfers to the workstation. A workstation is also characterized by a second pair of places noted REQ/ACK. The REQ place models a request to evacuate parts from the station. The ACK place allows the process evacuating the part to indicate to the process managing the workstation that the part has been evacuated. Figure 6 shows the generic PN process modeling any workstation from the perspective of the transportation system. It is exactly the model of processes that manage workstation called Zi in the example of Figure 2. The optional part does not exist for processes that manage workstation ISi or machine Mi. The difference is because Zi manage two kinds of objects: parts and pallets. The pairs of places PROD/CONS and REQ/ACK manage the composite object product on a pallet. The optional part of Figure 6 serves to manage part palletization (or depalletization) on (from) a pallet.

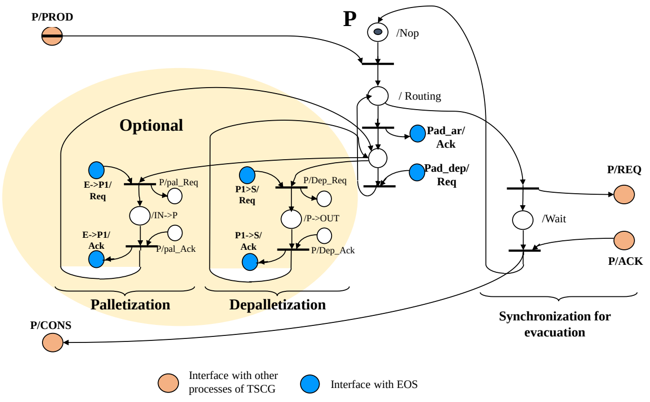

Figure 6. Generic model of a workstation of transport system

4.2. Generic model of a transfer

In a Smart System, a transfer of part can be done by different transport resources: Autonomous Guided Vehicles (AGV), robot, or pallets on a conveyor. In all cases, a transfer can be abstracted in the same way than the process depicted by Figure 7.

The same transport resource can make different transfers. Therefore, to trigger a transfer, it is necessary to check several preconditions. The object to be transferred must be on the start workstation of the transfer (example O2 on Ai in Figure 7). There must also be a free space at the destination location so that the transport resource is not blocked by the current operation, since it may be requested for other transfers. The first precondition is defined by marking the REQ place of the PN process that manage the upstream workstation (Ai in Figure 8). The second precondition is obtained by marking of the place CONS modeling the residual capacity in free spaces of the downstream workstation. If both conditions are met (stage 1 of Figure 7) then the evacuation is carried out (stage 2 of Figure 7). These two first stages are modelled in the transfer process through the START place. The transfer process can then issue an acknowledgement to the management process of the upstream station. The transfer then continues with the transfer itself and the deposit of the product on the downstream station (stage 3 of Figure 7). It is modeled by the END place in Figure 8.

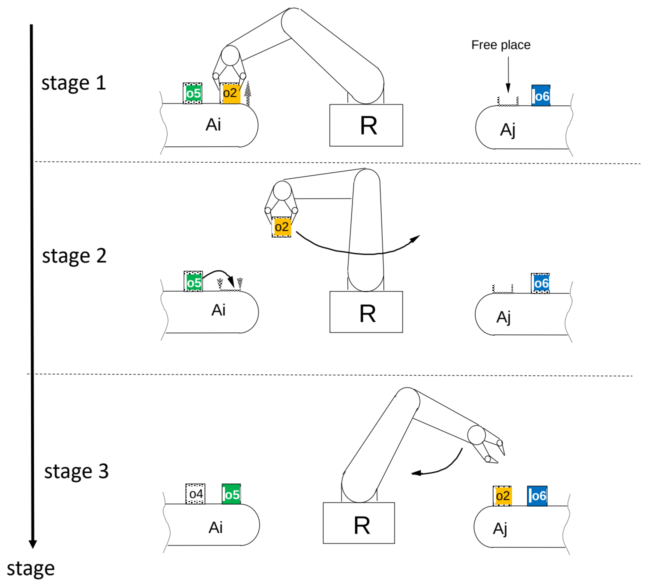

Figure 7. The different stages of a basic transfer.

Figure 8 is a CPN model of a generic transfer. To interpret this figure, the following definitions must be associated with it:

- *val KNPT = 4; (* NPT=Number of Part Types*)*
- *colset TPARTTYPE=index pi with 0 .. NPT; (* Definition of part types pi(0), pi(1), pi(2), .., pi(KNPT) ; pi(0) means no part *)*
- *colset TPI=record pit:TPARTTYPE * piid:INT;*
- *colset TAREA=with null|M1|M2|M3|M4|Z1|Z2|Z3|Z4|IS1|IS2|IS3|IS4|IS5|IS6 ;*
- *colset TPIDEST=record pit:TPARTTYPE*piid:INT*dest:TAREA ;*
- *colset TLISTTPIDEST= list TPIDEST (* Define a list of part *);*
- *var part:TPIDEST; (*Declaration of a variable of type TPIDEST to be bind with token in places of type TPIDEST *)*

As an example, the place PROD can contain a list with two part as *[{pit =PI(1), piid=12,dest=Z3},{pit =PI(2), piid=5,dest=Z1}].{pit =PI(1), piid=12,dest=Z3}* define a part of type PI(1) that have the identifier piid=12 and this part destination is Z_3.

Note here that there are other types of transfer corresponding to palletization or depalletization operations (their preconditions are defined in the optional part of Figure 6).

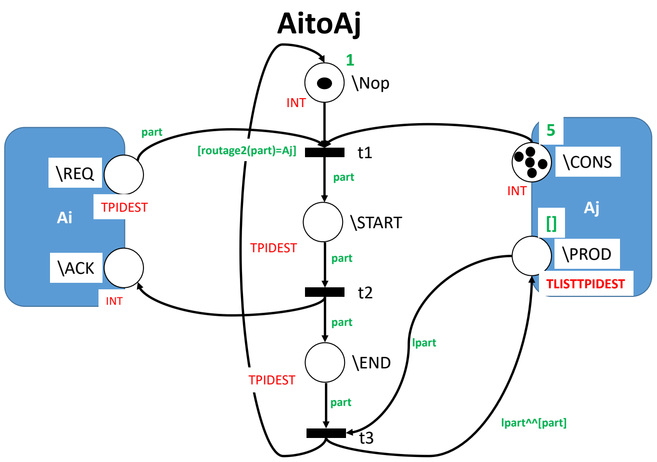

Figure 8. Generic process for transferring an object between two workstation in direct accessibility.

One can note that to model the PROD place as a list, one can use the CPN list type constructor. We have thus defined the type *TLISTTPIDEST* which is used as the type of the PROD place.

The routing of the parts is done from workstation to workstation based on the final destination of the part (for example Z3 for part 12). At the exit of each workstation, a routing function (such as the route2 function used as a guard for the $A_i$toA_j/t1 transition in Figure 8) is used to select a transfer to one of the next workstations downstream of the current workstation. Thus, from one station to another, the pallet is directed to the destination station. When it arrives at this station, the guard of the arrival transition is true and so the station communicates with the EOS to indicate that the part has arrived at its destination (See the Pad_ar/Ack interface position in Figure 6.

5. Extended Operating Sequences

An Extended Operating Sequence (EOS) models all the operations to be applied to transform a raw part into a finished part. It is extended to take into account its interactions in order to ask the pilot to allocate a machine to carry out the next transformation operation in its production line, and then to ask the TSCG to coordinate the transport resources in order to transport the product to this machine.

The EOS of a product involves dozens of processing operations and each operation can be performed on dozens of different machines. It is therefore necessary to propose a new model of EOS that allows to reduce this combinatorial explosion.

For this, let us take into account the generic transfer scheme given in Figure 2 as well as the routing principle presented in section 364. On the basis of these data, we specify by means of a sequence diagram, the operations to be implemented from the EOS point of view (Figure 9). It shows that each part transfer requires systematically five operations:

- A pre-allocation at the current location (M_s) to define the destination of the part (DA for Destination Area).
- A palletization of the part from the source machine M_s on the palletization area (called here LA for Local Area), in direct accessibility relation with the source machine.
- A transfer from LA to DA. LA and DA are a priori in indirect reachability relationship. In order not to model at the EOS level all the routing possibilities between two characteristic zones in indirect accessibility, we simply model the fact that the part is in transfer between LA and DA. The flexibility of the transfer and thus the combinatorics generated by this flexibility are taken into account by the CPN model of the TSCG (section 4).
- A request to assign the destination machine Md when the palletized part arrives in DA.
- A depalletization of the part in DA and its loading on the destination machine Md.

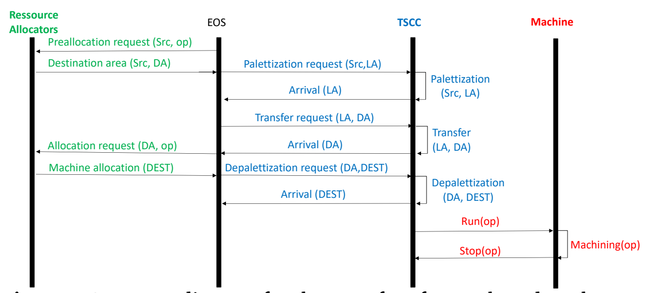
Ressource Allocators
EOS
TSCC
Machine
Preallocation request (Src, op)
Destination area (Src, DA)
Palettization request (Src,LA)
Palettization (Src, LA)
Arrival (LA)
Transfer request (LA, DA)
Transfer (LA, DA)
Allocation request (DA, op)
Arrival (DA)
Machine allocation (DEST)
Depalettization request (DA,DEST)
Depalettization (DA, DEST)
Arrival (DEST)
Run(op)
Machining(op)
Stop(op)

Figure 9. Sequence diagram for the transfer of a part based on the diagram in Figure 2

Once the part arrives in M_d, the EOS must start the manufacturing program corresponding to the requested machining operation. When it is finished, the status of the part machined on M_d is exactly the same as its initial status considered on the machine Ms. Thus, it is obvious to consider that M_d plays now the role of the previous M_s and that it is then possible to restart the whole procedure described here.

To be generic, it is necessary to deal with specific cases such as the departure of the part from "IN" stock and its return to "PiOUT" stock when it is finished. Indeed, when there is no longer a production operation, it is necessary for the pre-allocation/allocation function to direct the part towards the exit of the system (case 1). Another specific case is when pre-allocation designates the current machine as the one that is to carry out the next production operation. This means that at this point, the pre-allocation acts as an allocation and therefore it is not necessary to transfer the part since it is in the right location.

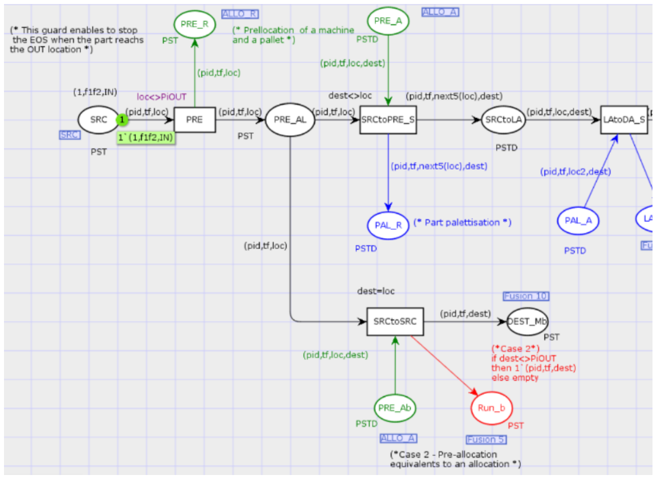

Figure 10. Generic EOS - Part 1

Figure 10, 11 and 12 give the generic CPN model of an EOS. It is decomposed into 3 figures because of its size and to allow a better reading. Some transition are repeated in several figures to show the link between the different parts of the model. For example, the transition "LAtoDA_s" (which models a request of a transfer from the location "LA" to the location "DA") is repeated in Figure 10 and Figure 11. The black places and transitions on these figures model the sequence of operating states for performing a transformation operation. The EOS is designed as a function that interprets the manufacturing sequence of each part as a list of processing operations to be performed. The green parts represent the asynchronous communications (interfaces) with the machines allocators. The blue places represent the interfaces with TSCG. The red places represent the interfaces with machine models (out of the scope of this study).

The CPN models are based on the following ML data structures:

(Definition of the location of a product *)*

colset OSA=with IN|INvZ1|Z1vZ2|Z2vZ3|Z3vZ4|Z1vZ3|Z3vZ1|Z4vZ1|Z1vOUT|Z2vM1|M1vZ2|Z3vM2|Z3vM4|M2vZ3|M4vZ3|Z4vZ3|M3vZ4|M1|M2|M3|M4|OUT|

IS1|IS2|IS3|IS4|IS5|IS6|Z1|Z2|Z3|Z4;

(Definition of manufacturing states of raw, semi-finished and finished products *)*

colset TRSF=with f1|u1|f2|u2|f3|u3|f4|u4|f5|u5|f1f2|u1f2|u1u2|f3f1f4|u3f1f4|u3f1u4|u3u1u4;

(Definition of the state of a product *)*

*colset PST=product PID*TRSF*OSA;*

(Definition of the state of a part and its final destination*)*

*colset PSTD=product PID*TRSF*OSA;*

In a CPN model, it is necessary to assign a type to each place in the model. The places of the EOS are 2 types: PST or PSTD. The type PST defines the state of a product from the point of view of its machining sequence (type TRSF) and from the point of view of its location in the production system (type OSA). For example, the place "SRC" (Figure 10) is the first place in the model. It defines the status of each product in the form of a 3-uple(part_id,manufacturing_sequence,location. Thus, the token (1,f1f2,IN), specifies that one has a part of type f1f2 present on the IN parts inbound stock and that it is the part of identifier 1. The identifier of each part is defined in relation to its type and makes it possible to know that it is processed in relation to the flow of parts in progress in the system.

The place "N_SRC" (Figure 12) represents the end of the EOS according to the sequence diagram in Figure 9 and is also of type PST. In fact the "SRC" and "N_SRC" places are merged by the blue 'src' tag in the figures. It means that they are in fact the same place, giving the EOS a recursive operating. Indeed, when the token arrives in the "N_SRC" place it means that all the operations necessary to perform a machining operation of the manufacturing sequence have been executed. For example, with respect to the product which was modelled by the token (1,f1f2,IN) in "SRC", a token (1,u1f2,M2) can be found in "N_SRC" . This means that the product is located on the machine M2 on which the transformation operation f1 was performed, leading to a semi-finished product. The merging of the two places means that the token (1,u1f2,M2) is now in the "SRC" square, indicating that the whole EOS process can now be started again in order to perform operation f2.

Note the guard defined by the expression "loc<> PiOUT" which is associated with the "PRE" transition. This condition avoid restarting the EOS when a product is exited to the stock "PiOUT". It is the stop condition of the recursive procedure implemented by the EOS.

Some of the places in the EOS and the interface places with the other models are of the PSTD type. This type adds the destination location (which is of type OSA) to the previous PST type. For example, this is the case of the "PRE_A" place (Figure 10) which is an interface place with the machine allocators. For example, the token (1, f1f2,IN,Z3), means that part 1 located on the IN stock must be transported to station Z3 of the conveyor. This is a pre-allocation that will be explained in the following section. The consequence is that all the interface places with the TSCG (blue colored places in the figures) are of type PSTD.

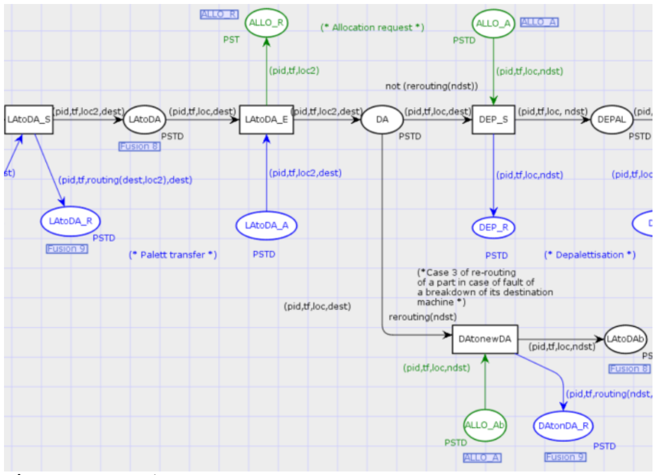

Figure 11. Generic EOS – part 2

Places "PRE_R", "PRE_A" and "PRE_Ab" (Figure 10) are used to manage a pre-allocation request. "PRE_R" is the request semaphore and "PRE_A" is the response semaphore. The transition "SRCtoSRC" models the situation of a pre-allocation equivalent to an allocation (note that the "PRE_A" and "PRE_Ab" places are merged by the tag "fusion 7"). This situation occurs when the allocated machine corresponds to the machine on which the part is located. In this case, the token modeling the product goes directly from the "SRC" place to the place modeling its presence on the destination machine. Therefore, the "DEST_Mb" place (respectively "Run_b") in Figure 10 is merged with the "DEST_M" place. (Respectively "Run") of Figure 12 by the tag "fusion10" (respectively "fusion 5").

A similar construction is performed in Figure 11 by the transition "DAtonewDA". It models the reorientation of a part to a new destination ("newDA") in case of failure of the destination machine "Md" during the transfer from "LA" to "DA". It is a case of dynamic reconfiguration.

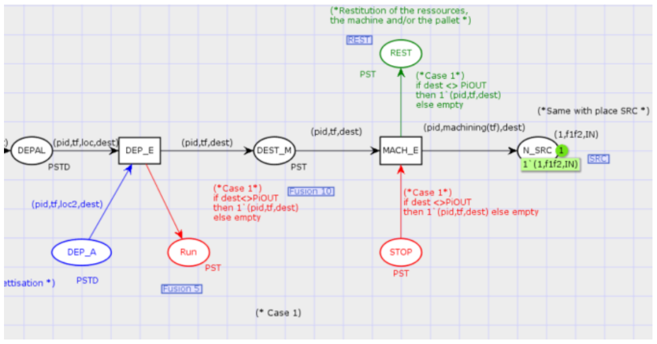
(*Restitution of the ressources, the machine and/or the pallet *)
DEST
REST
PST
(*Case 1*) if dest <> PiOUT then 1`(pid,tf,dest) else empty
(*Same with place SRC *)
DEPAL
(pid,tf,loc,dest)
DEP_E
(pid,tf,dest)
DEST_M
(pid,tf,dest)
MACH_E
(pid,machining(tf),dest)
(1,f1f2,IN)
N_SRC
1`(1,f1f2,IN)
SRC
(pid,tf,loc2,dest)
(*Case 1*) if dest<>PiOUT then 1`(pid,tf,dest) else empty
Fusion 10
(*Case 1*) if dest <>PiOUT then 1`(pid,tf,dest) else empty
DEP_A
Run
STOP
Fusion 5
(* Case 1)

Figure 12. Generic EOS – part 3

6. The piloting

In our approach, the role of piloting is to resolve the residual indeterminism of the control part. It is an intelligent function that can be implemented in different ways and with different formalisms. In particular, this function takes charge of resource allocation in order to arbitrate conflicting requests. The considered resources are pallets, machines, robots, switches and conveyor junctions. In this study, we have chosen to present only the allocation of machines. We have chosen here to use the formalism of CPN. This makes it possible to understand the role of the piloting and also to check correctness of the EOS in a verification stage (out of the scope of this study).

An originality of this work is to distinguish two stages in terms of machine allocation: pre-allocation and final allocation. The pre-allocation makes it possible to increase the performance of the production system.

To understand the idea, let us return to the generic scheme of transporting a given product in Figure 4. From the transport point of view, it is necessary to carry out the allocation of the destination machine (Md) before the part leaves. If its capacity is one unit, this means that this machine could no longer be assigned to other products. As a result, all other products that would have to be machined in Md would be blocked at their current location, instead of approaching Md in masked time. This would therefore limit the productivity of the system. For a given processing operation, pre-allocation consists of indicating the machine to which the part is to be transferred. This pre-allocation therefore does not require the machine to be free. It only requires that it is not faulty. The result of the pre-allocation is the workstation upstream of the destination machine. Let us call it, destination location or DA (see Figure 4). Thus, the final allocation is requested only when the product has arrived at DA, i.e. close to the machine Md. If the destination machine chosen by the pre-allocation is not faulty, the allocation consists of changing its status from free to busy (“occupied”) and then indicating to the EOS the final destination as corresponding to Md.

In the EOS, the pre-allocation request is modelled by the pair of places "PRE_R"/"PRE_A".

The allocation of a machine therefore corresponds to defining it as the final destination of a product. This is the reason why in our generic EOS model presented in the previous section, we also have the pair of places "ALLO_R"/"ALLO_A" which respectively model the allocation request and the response given by the resource allocator.

In fact, pre-allocation and allocation work very closely together. They both depend on the state of the machine and the location of the part at the time they are requested. For pre-allocation, the machine must be available (not broken down) and the request must be made at a source location such as Ms (Figure 4). For allocation, the machine must be free and operational. The request is made on DA the characteristic area before the final destination machine. In case of failure of this machine (occurring during the routing), the allocation request is implicitly interpreted by the control as a pre-allocation request. The part is then re-routed to another machine implementing the same function as the one that became defective during the transfer of the part to Md (see transition "DAtonewDA" in Figure 11).

On the contrary, in case of a pre-allocation request, if the pre-allocated machine is the machine on which the part is located, the request is interpreted by the control as an allocation request (cf. transition "SRCtoSRC" in Figure 10) These two cases show that it is necessary to implement the pre-allocation and the allocation in the same way. Therefore, the same controller implements both functions. Its behavior differs according to the location of the part at the time of the request and according to the state of the machine.

To implement the machine allocation controller, let us define the following types and variables in CPN ML.

(Define the status of a machine *)*

colset STAT=with free|occupied|faulty;

colset RES=subset OSA with [M1,M2,M3,M4];

colset RAWOP=subset TRSF with [f1,f2,f3,f4];

(Defines a list of transformation operations *)*

colset LOP=list RAWOP;

(Defines a machine with respect to its state and its list of transformation operations *)*

*colset MACH=product RES*STAT*LOP*OSA;*

(Defines the allocated machine type *)*

*colset ALMAC=product RES*PID*TRSF*LOP*OSA;*

Thus the type MACH, allows to specify a machine from the allocator point of view. For example, the token *(M_2,free,[f1,f3,f1f2],Z3)*, indicates that the machine M_2 is free, and that it can make parts of type f1, f3 or f1f2 (means operation f1 on a part of type f1f2). The value Z_3 indicates the location in the system from which an allocation request can be made. In all other locations, any allocation request received through the "*ALLO_R*" (allocation request) place is considered as a pre-allocation request. This place of the allocator model is merged with the "*ALLO_R*" place of the EOS (Figure 13).

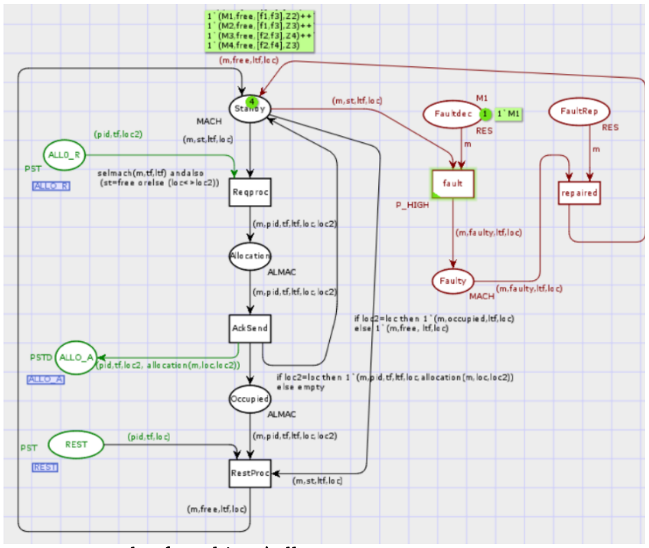

Figure 13. Example of machines' allocator

These types are used to define the type of places of the CPN model in Figure 13. Next, let us define the variables that are used to write the model expressions.

(Part identifiers *) var pid,pid2:PID;*

(Location variables *) var loc, loc2,dest,ndst: OSA;*

(Machining operation variables *) var tf,tf2: TRSF;*

(List of machining operations performed by a machine *) var ltf:LOP;*

(Status variable of a machine *) var st:STAT;*

(Variable defining a machine *) var m:RES;*

By using these variables, it is possible to write the expressions of the CPN arcs. For example, the expression of the arc between the "*Standby*" place and the "*Reqproc*" transition is *(m,st,ltf,loc)*. It means that the "*Reqproc*" transition is validated if there is at least one such token at the "*Stanby*" place and an expression token *(pid,tf,loc2)* at the "*ALLO_R*" place. The "*Standby*" place is the default place that models the state of the machine.

In order to select a machine that can be allocated, let us define the ML function noted "*selmach*". Its definition is as follows:

fun selmach(r:RES, ope:TRSF, lope:LOP)=

if lope=nil then false

else if ope=hd lope then true

else selmach(r,ope, tl lope) ;

This function is used in the guard of the "*Reqproc*" transition to check if the "*r*" machine implements the "*ope*" operation. This operation must be included in the list of operations of '*r*'. This ML function is defined recursively.

Likewise, let us define the allocation function in ML.

fun allocation(r:RES,l:OSA,l2:OSA)=

if l2=l then r else l;

This function is used in the arc expression between the transition "*AckSend*" and the place "*ALLO_A*". It defines the destination of a part. Although named allocation, it also implements pre-allocation. Indeed, if 'l_2' the location of request corresponds to 'l', the location of loading of the machine 'r', then the destination is the resource 'r', and so it is an allocation. Otherwise the destination is 'l', and it is thus a pre-allocation.

In Figure 13, the entire green part of the model represents the interface with the EOS. The part in purple models the management of machine failures and repairs. When it is validated, the "*fault*" transition has priority in order to be able to remove from the "*Standby*" place the tokens of a broken machine. The rest of the model in black color represents the pre-allocation/allocation process. Note the use of an alternative expression for the arcs between the "*AckSend*" transition and the "*Standby*" and "*Occupied*" places respectively. In case of pre-allocation (case where loc2<>loc), no token is added to the "*Occupied*" place and a machine token is put back to the "*Standby*" place with the free value for the machine status. In the case of an allocation, the machine token is put in both the "*Standby*" place and the "*Occupied*" place with the occupied value for its status. One may be surprised that the token is put back in the "*Standby*" place. This allows other parts to request pre-allocations even if the machine is machining a part.

The guard of transition '*RestProc*' is defined by the expression *selmach(m,tf,ltf) andalso (st=free orelse (loc<>loc2))*. This expression means that a pre-allocation is made if the position of the part (loc2) is different from the allocation request position of the machine (loc). If the part is in '*loc*', the selected machine must be free (*st=free*).

When a machine is returned after the unloaded part has been evacuated to DA (in case a token is placed in the "*REST*" place), it is necessary to reset the status field of the token of the machine in the "*Standby*" place to the "*free*" value. This is why the transition "RestProc" is validated simultaneously by the "*Standby*" and "*Occupied*" places. When this transition is fired, it removes the tokens from both places and returns a token whose status field value has been reset to "free" to the "*Standby*" place.

7. Conclusions

In this paper we have proposed the main bricks to model sequential controllers for the implementation of Smart Manufacturing Systems. All of these controllers are presented in the form of three layers with interlayer communications. The intermediate layer is the EOS. It communicates with the highest layer in which the machine allocators are found. In practice, the allocation results in the definition of an orientation of the part towards the next machine according to its manufacturing sequence. Once this direction is known, the EOS can ask the TSCG (low layer) to transport the product to this machine.

All the models are developed in the formalism of CPN, allowing in particular verification by simulation. Each model has been checked separately to ensure that, provided the other models are good properties, it is live and bounded. So the model of a system can be created simply by copying and pasting these modeling bricks.

The next step will be to develop a method for systematically translating these models into code that can be implemented on industrial computers. For example, the TSCG models are developed as safe PN processes (disregarding PROD/CONS places) and therefore could be easily translated into Sequential Functional Chart (IEC 61113-3) in order to be implemented on Programmable Logic Controller (PLC).

References

Beach, R., Muhlemann, A. P., Price, D. H., Paterson, A. and Sharp, J. A. (2000). A review of manufacturing flexibility. European journal of operational re-search, 122(1), 41-57.

IEC 61113-3 (2013). Programmable controllers –Part3: Programming languages https://webstore.iec.ch/publication/4552

Jensen, K. and Kristensen, L. M. (2015). Colored Petri nets: a graphical language for formal modeling and validation of concurrent systems. Communications of the ACM, 58(6):61-70.

Hozdić, E. (2015). Smart factory for industry 4.0: A review. International Journal of Modern Manufacturing Technologies, 7(1):28-35.

Koren, Y. (2014). Reconfigurable Manufacturing System. Preprint of the CIRP Encyclopedia of Production Engineering. Springer Berlin Heidelberg, 1035-1039.

Koren, Y. and Shpitalni, M. (2010). Design of reconfigurable manufacturing systems. Journal of manufacturing systems, 29(4):130-141.

Mittal, S., Khan, M. A., Romero, D. and Wuest, T. (2019). Smart manufacturing: characteristics, technologies and enabling factors. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 233(5):1342-1361.

Qu, Y. J., Ming, X. G., Liu, Z. W., Zhang, X. Y. and Hou, Z. T. (2019). Smart manufacturing systems: state of the art and future trends. The International Journal of Advanced Manufacturing Technology, 103(9-12), 3751-3768.

Radziwon, A., Bilberg, A., Bogers, M. and Madsen, E. S. (2014). The smart factory: exploring adaptive and flexible manufacturing solutions. Procedia Engineering, 69:1184-1190.

Sprock, T. and McGinnis, L. F. (2015). A conceptual model for operational control in smart manufacturing systems. IFAC-PapersOnLine, 48(3), 1865-1869.

Toguyeni A., 2018. Design and verification of discrete event controllers for Smart Factory. IMAACA'2018, Budapest (Hungry).