

## An agent-based Flexible Manufacturing System controller with Petri-net enabled algebraic deadlock avoidance

Sotirios C. Messinis<sup>1</sup>, George-C. Vosniakos<sup>2</sup>

<sup>1</sup>National Technical University of Athens, School of Mechanical Engineering, Manufacturing Technology Section, Athens, Greece, e-mail: sotirismn@gmail.com

<sup>2</sup>National Technical University of Athens, School of Mechanical Engineering, Manufacturing Technology Section, Athens, Greece, e-mail: vosniak@central.ntua.gr

---

### Article Info

#### Article history:

Received August 30, 2020  
Revised September 28, 2020  
Accepted October 3, 2020

---

#### Keywords:

Flexible Manufacturing  
Petri Net  
Resource Allocation System  
Deadlock Avoidance  
Agents

---

### ABSTRACT

This work focuses on the efficient design of a controller for a Flexible Manufacturing System (FMS) using Agents. The necessary agents were selected and defined according to the Design of Agent-based Production Control Systems (DACs) methodology. The Contract Net Protocol (CNP) was applied for agent communication and interaction. A particular Algebraic Deadlock Avoidance Policy (DAP) is efficiently embedded into CNP. As a result the multi agent system is live and deadlock-free. Feasibility analysis of the controller was performed by exploiting Resource Allocation Systems techniques being defined in the framework of Petri Net theory. The controller is demonstrated in simulation mode in the framework of the Java Agent Development Framework (JADE) system.

*Copyright © 2020 Regional Association for Security and crisis management and European centre for operational research.  
All rights reserved.*

---

### Corresponding Author:

George-C. Vosniakos,  
National Technical University of Athens, School of Mechanical Engineering  
Email: vosniak@central.ntua.gr

---

## 1. Introduction

Flexible Manufacturing Systems (FMS) are soft-programmable Manufacturing Systems that are capable of producing a variety of parts, categorized into families, irrespective of the mix and quantity of each part family at each point in time (Pérez-Pérez et al., 2018). They have been introduced several decades ago as a means to serve the mass customization paradigm in manufacturing (Suzić et al., 2018) having undergone several revamps over time, which have been associated with new terms, such as Agile, Digital, Virtual, Intelligent, Smart Manufacturing etc. (Esmailian et al., 2016), the latest one being Cyber-physical in the framework of Industry 4.0 hype (S. Wang et al., 2016).

The most crucial part of FMS design is its control system design, for which both theoretical (mostly Petri-net (Z. Li & Zhou, 2009) and Agent-based (Bussmann et al., 2004) approaches as well as simulation approaches (Leitão & Karnouskos, 2015) have been suggested aiming at the management of the discrete events involved in operating such a system, part of which constitutes the scheduling problem (Cardin et al., 2017). However, FMS control in its generality refers to the coordination of manufacturing equipment in order to achieve the required part processing schedule (Monostori et al., 2016).

Among the methods employed in designing FMS controllers, Multi Agent Systems are widely accepted. The suitable design methodology for agent selection and their subsequent programming, the right

communication protocol and the necessary avoidance of emerging undesirable states, such as deadlocks, during FMS operation are of crucial importance (Leitão & Karnouskos, 2015).

The classical approach of FMS control is mainly characterized as hierarchical and timetable-based, which is inadequate in case of system disturbances. Furthermore, every possible divergence from the program directly affects the neighboring units causing cascading disturbance phenomena. The lack of system-wide reconfiguration makes impossible the limitation of disturbances. Such problems can be overcome by the necessary freedom for a unit to choose its actions with respect to its current situation. In this way, the production program has to be divided into sub-goals and distributed to the existing system units (Rocha et al., 2017). In system control distributed to the physical components of system the engaged local controllers have to intelligently make decisions about the sub-goals and cooperate in an efficient manner. Hybrid hierarchical – distributed approaches have also been explored (Jimenez et al., 2017).

Currently, the trends in production control focus on two main streams of approaches, namely heterarchical and holonic control. Heterarchical control was developed by reaction to the limitations of centralized and hierarchical control architectures (Esmaeilian et al., 2016). In these cases, the control logic becomes extremely difficult to implement because of the large amount of excessive information with respect to system size (Monostori et al., 2016). On the other hand, the main result of heterarchical control is autonomy as implemented by the connection of local controllers through a communication network. The other paradigm, namely holonic manufacturing systems focus on the whole manufacturing process instead of the local control elements. Such a system consists of autonomous units (holons) which cooperate flexibly. In agent-based control framework, both the above cases of autonomy and cooperation concepts are efficiently implemented highlighting its power (Bussmann et al., 2004) and more recently (L. Wang & Haghghi, 2016).

Multi-agent based manufacturing system controllers have been extensively studied in literature (Azizi et al., 2018), (Hsieh, 2008), (Monostori et al., 2016). From the onset, two main pertinent problems were identified, namely how to select them in order to effectively cover the control domain and how to establish their efficient cooperation (Leitão & Karnouskos, 2015). Several answers have been proposed to these issues: as regards identification of agents and their total design DACS methodology has been proposed (Bussmann et al., 2004), whilst, as regards their programming in JAVA, JADE (Java Agent Development Framework), the unique compliant platform with FIPA (<http://www.fipa.org>), has been developed (F. L. Bellifemine et al., 2007). Furthermore, Contract Nets have been used for the efficient communication and cooperation among the parts of the system (F. Bellifemine et al., 2003).

Petri nets have also been widely used for controlling an FMS (Z. W. Li & Zhou, 2004) especially focusing on the detection and avoidance of undesirable states, which cannot be dealt with formally by Agents alone (Z. Li & Zhou, 2009). Among many pertinent approaches, an analytical method for creating reversibility-enforcing supervisors through bounded Petri Nets has been proposed (Giua & Seatzu, 2015), exploiting parts of the Theory of Regions and the enforcement of Generalized Mutual Exclusive Constraints in the behavior of the system with monitor places. In another approach, a multi-step look-ahead deadlock prediction method was proposed to obtain a deadlock avoidance policy for a class Petri nets, without calculating a complete reachability graph through structural and functional simplification techniques (Lin et al., 2020). Next, a Deadlock Avoidance Policy (DAP) was developed exploiting Petri Net structural analysis (Luo et al., 2019). Similarly, various types of illegal markings that can be simply prevented by polynomial algorithms are structurally identified in Resource Oriented Petri Nets (Chen et al., 2019). Furthermore, a generalized class of Polynomial–Kernel DAPs focuses on net behaviors with non – convex state space representation which cannot be dealt with by the Theory of Regions (Reveliotis, 2017). Petri Nets have been embedded in an Agent-based system for dealing with the potential emergence of system deadlocks (Hsieh, 2008).

The gap that has been identified in the –otherwise- rich literature concerns the efficient combination of Algebraic DAPs – represented by Petri Nets – for achieving feasibility analysis and subsequent coordination and cooperation of the developed agents constituting the complete controller. More specifically, a particular Algebraic DAP is efficiently embedded into the communication and interaction protocol of the developed agents, namely Contract Net Protocol, as a result imposing on the multi agent system to be live and deadlock–free. This is highlighted as the innovation and contribution of the current work. The developed methodology is demonstrated for a particular FMS layout which is being developed in an academic environment for research purposes.

In Section 2 the necessary theoretical background is succinctly outlined. Section 3 briefly describes the FMS that is used as testbed. Section 4 presents the agent-based controller design. Section 5 outlines simulation results of controller execution. Conclusions and further research potential are discussed in Section 6.

## 2. Basic theoretical foundations

### 2.1 Multi Agent Systems

Software agents offer an innovative approach to the design and operation of complex distributed systems. They exploit decision and interaction enabling dynamic reaction of systems to unpredictable events by embedding a range of behaviors and being capable of adapting to environmental changes (Bussmann et al., 2004; S. Wang et al., 2016). Furthermore, multi-agent systems are ideally suited to controlling manufacturing systems as they are characterized by a distributed nature. Their interactions are based on coordination and negotiation.

Following Contract Net Protocol (CNP), there are two kinds of agents: managers and bidders. Bidders can be supposed to be potential contractors. After initialization the protocol continues as follows: the manager announces the job to the contractors, the contractor answers with a bid, the manager makes comparisons among the received bids and chooses the best according to some criterion. The contractor, who sent the best bid, receives a message for creating a contract with the manager for the particular job. CNP simply stops after the manager received the first bid from each bidder instead of repeating the auction until no bidder changes its bid. Based on its communication structure, CNP can be viewed as an auction mechanism.

### 2.2 Petri nets

A generalized Petri net is a 4-tuple  $N = (P, T, F, W)$  where  $P$  and  $T$  are finite, non-empty, and disjoint sets of places and transitions, respectively,  $F$  is a *flow relation*, represented by arcs with arrows from places to transitions or from transitions to places.  $W$  is a mapping that assigns an integer weight to an arc. A marking vector represents the number of tokens at each place. If this does not exceed a bound then the net is bounded. It is *structurally bounded* iff it is bounded for any initial marking,  $M_0$ . A pure net  $N = (P, T, F, W)$  can be represented by its input and output matrix, indicating the arcs flowing into and out of transitions, respectively. The difference of these matrices constitutes the net's incidence matrix  $[N]$ . The flow relation of a pure Petri Net can be represented by the flow matrix  $\theta = \theta^+ - \theta^-$  where  $\theta^+[p, t] = W(t, p)$  and  $\theta^-[p, t] = W(p, t)$ . The definition of *liveness* refers to the disappearance of deadlocks. A net  $N$  with initial marking  $M_0$  ( $N, M_0$ ) is live iff all its transitions are enabled by corresponding firing sequences. A Petri Net  $N$  is reversible iff it is live and deadlock-free. A p-invariant (place invariant)  $X$  is a vector of places that is solution to  $X^T[N] = 0^T$ . A t-invariant (transition invariant)  $Y$  is a vector of transitions that is solution to  $[N]Y = 0$ . In a Petri net, *siphons and traps* are structural objects that involve marking invariants. A non empty set of places  $S \subseteq P$  is a siphon iff the set of input transitions of the siphon  $S \bullet$ , is a subset of the output transitions  $S \bullet$ , formally  $S \bullet \subseteq S \bullet$ . Similarly,  $S \subseteq P$  is a trap iff  $S \bullet \subseteq S \bullet$ . A siphon (trap) is minimal iff there is no siphon (trap) contained in it as a proper subset. A minimal siphon  $S$  is said to be strict if  $S \bullet \subseteq S \bullet$  and  $S \bullet \neq S \bullet$ . In the relevant bibliography, strict minimal siphons can be divided into elementary and dependent ones, the latter being expressed as a linear combination of the former. Feasibility analysis of the controller in this work is based on elementary and dependent siphons theory. Further details can be found in (Z. Li & Zhou, 2009).

### 2.3 Resource Allocation Systems (RAS)

A sequential RAS is defined by a quintuple  $F = \langle R, C, P, A, D \rangle$ , where:  $R$  is the set of the system resource types,  $C$  is the system capacity function, characterizing the number of identical units from each resource type available in the system.  $P$  denotes the set of supported process types  $\Pi_i$ , each of them being a composite element itself with  $\Pi_j = \langle \Delta_j, G_j \rangle$ , where:  $\Delta_j = \{ \varepsilon_{j1}, \varepsilon_{j2}, \dots, \varepsilon_{jn} \}$  denotes the set of processing stages involved in the definition of process type  $\Pi_j$ , and  $G_j$  is a data structure that encodes the sequential logic integrating the set of the processing stages  $\Delta_j$  into a set of potential process flows.  $A$  is the resource allocation function associating every processing stage with the resource allocation vector required for its execution.  $D$  is a function mapping each processing stage to processing time distribution. Further details can be found in (Reveliotis, 2017).

In this paper we focus on a certain class of RAS, namely Disjunctive –Conjunctive RAS (D-C RAS). The main reason for unsafe RAS conditions is partial deadlocks. For the efficient avoidance of such undesirable states, Polynomial Kernel Deadlock Avoidance Policies (PK – DAP) are proposed.

### 2.4 Polynomial Kernel Deadlock Avoidance Policies (PK-DAPs)

In modeling a RAS with Petri Nets, a deadlock formation is based on the lack of reversibility of the net. In the reachability space  $R(N, M_0)$  is represented with strongly connected components which are not co-

An agent-based Flexible Manufacturing System controller with Petri-net enabled algebraic deadlock avoidance (Sotirios C. Messinis)

reachable. Thus, a true deadlock avoidance policy must limit system operation to a strongly connected component of  $R$  which will include the initial marking  $M_0$ .

PK-DAPs' computational cost is polynomially related to the size of the underlying RAS. These policies will remain scalable for every given instance of the corresponding RAS class. In this study, PK-DAP is represented with the tuple  $(A, b)$ , where  $A$  is a  $K \times D$  real-valued matrix and  $b$  is a  $K$ -dimensional real-valued vector. It can be represented by a set of linear inequalities as:

$$A \times M_S \leq b \quad (1)$$

being imposed on the system behavior with the introduction of a monitor place  $p_c(k)$ . This place is connected with the rest of the network from the flow matrix

$$\theta_{p_c}(k) = -A(k, \cdot) \times \theta_S \quad (2)$$

where  $\theta_S$  denotes the flow sub-matrix of the uncontrolled network  $N = (P, T, W, M_0)$  corresponding to places  $p \in P_S$ . The initial marking of place  $p_c(k)$  is set to

$$M_0(p_c(k)) = b(k) \quad (3)$$

The resulting controller imposes Eq. (1) on the original system behavior by establishing the place invariant

$$A(k, \cdot) \times M_S + M_0(p_c(k)) = b(k) \quad (4)$$

Liveness-enforcing supervisor (LES) prevents the formation of potential deadlocks for a given RAS  $\Phi$  by imposing a separate constraint, in the form of a linear inequality, on the marking of the RAS-modeling net  $N(\Phi)$ . In this framework, we initially seek to control more than one strict minimal siphons of the net by a single marking inequality, by taking advantage of the marking dependencies. After dividing the strict minimal siphons into elementary (independent) and dependent we add monitors to elementary siphons only, an approach known as implicit siphon control (Z. W. Li & Zhou, 2004), (Reveliotis, 2017). If a dependent siphon cannot be implicitly controlled like this, a monitor is added for it.

Note that a siphon in a Petri Net is controlled if it never gets empty (Z. Li & Zhou, 2009), which can be expressed as

$$\forall m \in R(N, m_0), \quad m(S) = \sum_{p \in S} m(p) > 0 \quad (5)$$

The characteristic  $P$ - and  $T$ -vectors,  $\lambda_S$  and  $\eta_S$  respectively, of a siphon  $S$  can be defined as a  $|P|$ -dimensional binary vector  $\lambda_{S[p]} = I_{\{p \in S\}}$ ,  $p \in P$ , where  $I_{\{p \in S\}}$  indicates the corresponding P-vector of  $S$  and a  $|T|$ -dimensional integer vector, with  $\eta_S^T = \lambda_S^T \cdot \theta_S$ . We need to focus on the  $T$ -vectors which share linearly independent characteristics. Their corresponding siphons need to be elementary in order to be controlled. The remaining dependent vectors are necessarily computed by a linear expression of the independent ones that share dependent characteristics with them.

Based on the formation of  $T$ -vectors of the elementary siphons we can properly add monitors  $V_S$ , to ensure their controllability. Furthermore, we introduce a parameter  $k_S$ , namely the control depth variable of the siphon  $S$ . Thus, elementary siphon  $S_i$  is controlled under the following inequality:

$$M_0(S) > \sum_{i=1}^m (M_0(S_i) - k_{S_i}) \quad (6)$$

The number of monitors which are finally introduced may be greater than that of the elementary siphons if controllability condition of the latter does not hold. However, Eq. (6) is a sufficient but not necessary condition, as it will be shown in section 4.3. By employing linear programming, the necessary monitors are obtained and the obtained supervisor is verified, (Z. W. Li & Zhou, 2004) as follows:

$$\min \sum_{i=1}^m k_{S_i} \quad (7)$$

subject to:

$$M_0(S) > \sum_{i=1}^m a_i (M_0(S_i) - k_{S_i}), a_i \in \mathbb{R}, 1 \leq k_{S_i} \leq M_0(S_i) - 1, i = 1, \dots, m$$

If there is a feasible solution, then, the controlled net  $(N_{0V}, M_{0V})$  resulting from the addition of the monitors to primary net's elementary siphons  $S_i$ , is live. Otherwise, an additional monitor for a dependent siphon  $S$  can be introduced. Note that a control depth variable  $k_{S_i}$  can be properly adapted upwards, if necessary, but in that way it contributes to the degrading of the net's control performance. Further details can be found in (Reveliotis, 2017).

### 3. The FMS studied

The layout of the FMS which is studied as an example in this paper is depicted in Figure 1. The FMS consists of a cnc milling machine ( $M_1$ ), a cnc turning machine ( $M_2$ ), a robot ( $R$ ), an storage buffer for unprocessed workpieces (In), a storage buffer for finished workpieces (Out), two intermediate storage buffers for work-in-progress (B<sub>1</sub> for milling and B<sub>2</sub> for turning). Capacity of all buffers is assumed to be equal to 8. Ten workpiece types are processed ( $W_i, i = 1, \dots, 10$ ) as in Table 1, defining 10 jobs respectively.

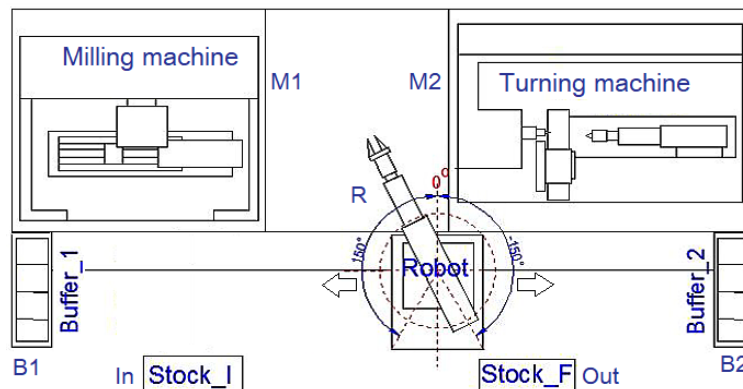
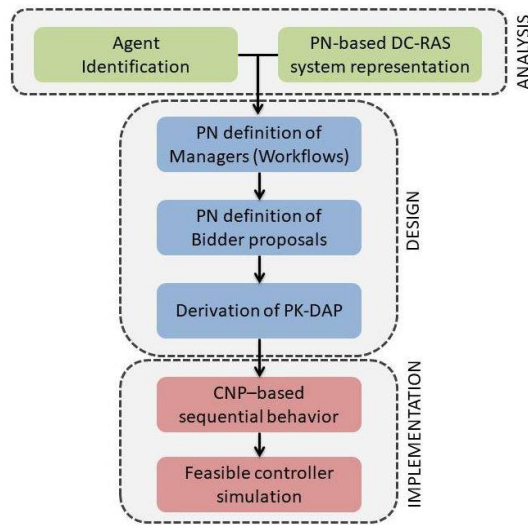


Figure 1. FMS layout

Table 1. Process requirements per workpiece type

Workpiece Type	Process plan	Workpiece Type	Process plan
W <sub>1</sub>	M <sub>1</sub>	W <sub>6</sub>	M <sub>2</sub> , M <sub>1</sub> , M <sub>2</sub>
W <sub>2</sub>	M <sub>2</sub>	W <sub>7</sub>	M <sub>1</sub> , M <sub>2</sub> , M <sub>1</sub> , M <sub>2</sub> , M <sub>2</sub>
W <sub>3</sub>	M <sub>2</sub> , M <sub>1</sub>	W <sub>8</sub>	M <sub>2</sub> , M <sub>1</sub> , M <sub>2</sub> , M <sub>1</sub> , M <sub>1</sub>
W <sub>4</sub>	M <sub>1</sub> , M <sub>2</sub>	W <sub>9</sub>	M <sub>1</sub> , M <sub>2</sub> , M <sub>2</sub> , M <sub>2</sub> , M <sub>1</sub> , M <sub>1</sub> , M <sub>2</sub>
W <sub>5</sub>	M <sub>1</sub> , M <sub>2</sub> , M <sub>1</sub>	W <sub>10</sub>	M <sub>2</sub> , M <sub>1</sub> , M <sub>1</sub> , M <sub>2</sub> , M <sub>2</sub> , M <sub>1</sub> , M <sub>2</sub>

The flow diagram of activities followed to construct the system's controller is shown in Figure 2. Loading and unloading is executed by the robot, under the following rules: (a) The system has two inputs: each workpiece can be transferred from the initial stock to either M<sub>1</sub> or M<sub>2</sub> (b) workpieces waiting for further processing at buffers B<sub>1</sub> and B<sub>2</sub> are of higher priority compared to the rest, thus contributing to the minimization of lead time (c) if a machine is not available, the workpiece can wait in the corresponding buffer for further processing. Note that the controller is not engaged in workpiece placement in buffers.



**Figure 2.** Flow of activities in constructing the FMS controller

## 4. FMS operations modelling

### 4.1 Design of Agent-based Control (DACS)

DACS methodology (Bussmann et al., 2004) was employed.

#### 4.1.1 Control Decision Analysis

At this stage, the decisions whose execution involves some physical action in the system are identified, as well as the dependencies among them. Assessing all the alternative decisions given in the framework of system resources leads to their characterization. Control interfaces triggering pertaining to process states definition and the underlying decision spaces are necessary details for decisions characterization.

Decisions involved in system operation were identified as in Table 2, whereas dependencies among decisions can be classified as in Table 3.

**Table 2.** Control Decisions (wkp: workpiece)

	<b>Decision</b>						
	$D_1$	$D_2$	$D_3$	$D_4$	$D_5$	$D_6$	$D_7$
<b>Title</b>	Choose a wkp in In	Move wkp to $M_i$	Move wkp to $B_i$	Processing on $M_i$	Waiting at $B_i$	Move wkp to Out	Choose next machine
<b>Parameters</b>	-	$W_1 - W_{10}$	$W_3 - W_{10}$	$W_1 - W_{10}$ and $M_i$	$W_1 - W_{10}$	-	-
<b>Control Interface</b>	Robot	$M_i$	$B_i$	$M_i$	$B_i$	Output stock	$M_i$
<b>Trigger</b>	$M_i$ is free	$W_3 - W_{10}$ arrives at $M_i$	$W_3 - W_{10}$ arrives at $B_i$	$W_1 - W_{10}$ arrives at $M_i$	-	-	-
<b>Decision Space</b>	$W_1 - W_{10}$	{Input on $M_i$ }	{Input on $B_i$ }	Total processing at $M_i$	-	Total number of processing wkps	{ $W_3 - W_{10}$ }
<b>Local Decision Space</b>	Choose a workpiece	{ $W_1 - W_{10}$ } on $M_i$	{ $W_1 - W_{10}$ } on $B_i$	{ $W_1 - W_{10}$ } on $M_i$	-	Choose finished wkp if ready.	-

**Table 3.** Decision Dependencies (wkp: workpiece)

	<b>DP<sub>1</sub></b>	<b>DP<sub>2</sub></b>	<b>DP<sub>3</sub></b>	<b>DP<sub>4</sub></b>
<b>Title</b>	Wkp routing on next machine $M_i$	Wkp routing on buffer $B_i$	Minimization of necessary routing per wkp	Limitation to necessary machine processing
<b>Decision Tasks</b>	$D_2$ & machine processing capabilities	$D_3$	$D_2, D_3$	$D_1, D_2, D_3, D_4, D_5$
<b>Constraints</b>	Wkp move to processing machine	Move a wkp to buffer	Choice of the right machine order	Input of wkp into the system
<b>Preferences</b>	-	Wkp move to corresponding buffer	Choice of the right machine and buffer order	-

4.1.2 Identification of the final agents

Based on the resources of the system, decisions can be further divided into sub-spaces.  $D_1, D_5$  and  $D_6$  refer only to the choice of workpieces, hence they cannot be divided further. On the contrary,  $D_2$  refers to Workpiece routing to either milling or turning and is included in dependency  $DP_1$ , so it can be divided into: (a)  $D_{2m}$  concerning machine (b)  $D_{2w}$  concerning workpiece (c)  $D_{2r}$  concerning robotic transfer. Similarly  $D_3$  is divided into:  $D_{3r}$  and  $D_{3w}$ ,  $D_4$  is divided into  $D_{4m}$  and  $D_{4w}$ , and  $D_7$  is divided into  $D_{7m}$  and  $D_{7w}$ . Identification of agents is presented in Table 4.

**Table 4.** Final Agent Identification

<b>Agent</b>	<b>Decision</b>	<b>Decision description</b>
Machine	$D_{4m}$	Process at $M_i$
	$D_{7m}$	Choose next machine
Robot	$D_1$	Load Workpiece
	$D_{2r}$	Move to $M_i$
	$D_{3r}$	Move to $B_i$
	$D_6$	Unload the product
Buffer	$D_5$	(Waiting for Process)
Workpiece	$D_{2w}$	Workpiece Moving at $M_i$
	$D_{3w}$	Workpiece Moving at $B_i$
	$D_{4w}$	Workpiece Processing at $M_i$
	$D_{7w}$	Choose next Machine

4.2 Controller Development with CNP

The Managers (or contractors) of the system correspond to the different workflows related to the workpieces. The bidders of the system are: Bidder 1:  $M_1$  &  $M_2$ , Bidder 2:  $B_1$  &  $B_2$ , Bidder 3:  $R$ . Thus, every bidder is responsible for a group of resources. Bidders can take part in one or more actions of a workflow creating contracts with managers. A manager negotiates with potential bidders and decides which of them can fulfill a job. In general, the total of bidders and managers create a collaborative network.

Managers conduct feasibility analysis of the collaborative network based on the corresponding workflow synthesis with the models of bidders, which are represented as PNs, see Figure 3. Note that, if a manager does not have the required resources to execute its job, it asks other managers for resources. Transitions indicate the operations in each job whilst their states are indicated by places. The first and final places of bidder's PN proposal point out the allocation (e.g.  $M_{1a}(i), M_{2a}(j)$ ) and de-allocation (e.g.  $M_{1d}(i), M_{2d}(j)$ ) of its resources, respectively, see Figure 3. This procedure is implemented each time with the right place and transition referring to the managers' PN models, see Figure 4 and Table 5.

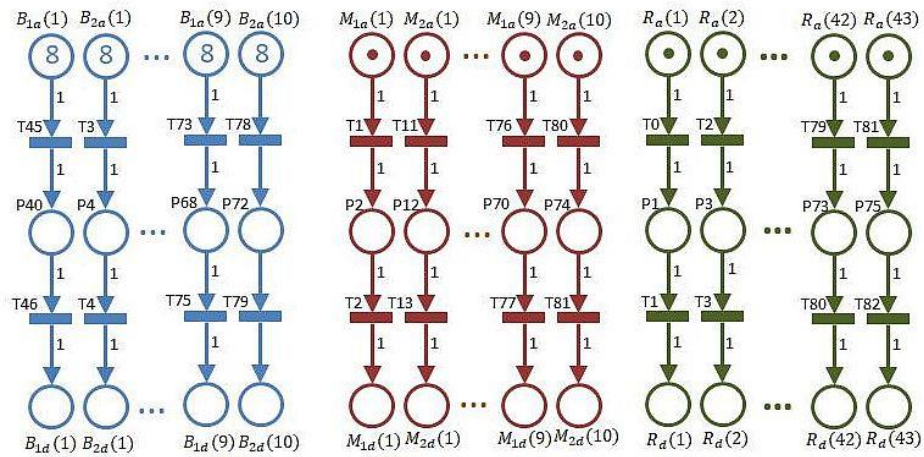


Figure 3: Sample PN-based proposals by Buffers (blue), Machines (red) and Robot (green)

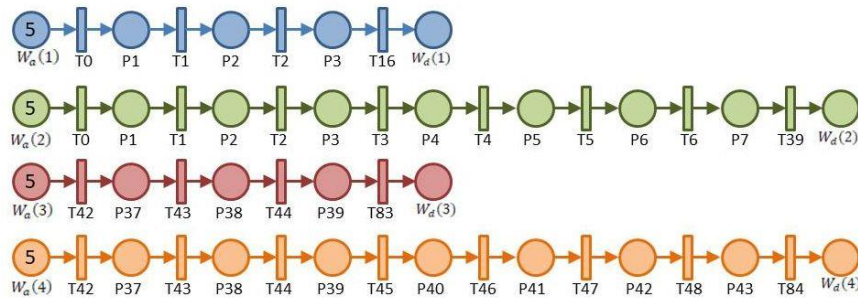


Figure 4: Sample PN – based workflows of Manager 1 ( $W_1$ ) to Manager 4 ( $W_4$ )

Table 5. Places and Transitions in Figure 4 (wkp: workpiece)

Places		Transitions	
P1: Robot Transition from Initial Stock to $M_1$	P37: Robot Transition from Initial Stock to $M_2$	T0: Robot Grasps wkp from the Initial Stock (P01)	T42: Robot Grasp wkp from the Initial Stock (P02)
P2: $M_1$ Processing	P38: $M_2$ Processing	T1: Robot leaves wkp on $M_1$ – Process starts	T43: Robot leaves wkp on $M_2$ – Process starts
P3: Robot Transition from $M_1$ to $B_2$ (Buffer 2)/Output	P39: Robot Transition from $M_2$ to $B_1$ (Buffer 1)/Output	T2: Process completed – Robot grasps wkp $W_i$ from $M_1$	T44: Process completed – Robot grasps wkp $W_i$ from $M_2$
P4: On $B_2$	P40: On $B_1$	T3: Robot loads wkp $W_i$ to $B_2$	T45: Robot loads wkp $W_i$ to $B_1$
P5: Robot Transition from $B_2$ to $M_2$	P41: Robot Transition from $B_1$ to $M_1$	T4: Robot grasps wkp $W_i$ on $B_2$	T46: Robot grasps wkp $W_i$ on $B_2$
P6: $M_2$ Processing	P42: $M_1$ Processing	T5: Robot leaves wkp on $M_2$ – Process starts	T47: Robot leaves wkp on $M_1$ – Process starts
P7: Robot Transition from $M_2$ to next station/Output	P43: Robot Transition from $M_1$ to next station/Output	T6: Process completed – Robot grasps wkp $W_i$ from $M_2$	T48: Process completed – Robot grasps wkp from $M_1$
		T16: Robot transfers wkp $W_3$ to Output Stock	T84: Robot transfers wkp $W_4$ to Output Stock



By merging the PN models of managers and bidders, including their allocation and de-allocation places, the following acyclic Marked Graphs are formally derived:

$$\text{Machine} = \{M_{1a}(i), M_{2a}(j)\} \cup \{M_{1d}(i), M_{2d}(j)\}$$

$$\text{Buffer} = \{B_{1a}(i), B_{2a}(j)\} \cup \{B_{1d}(i), B_{2d}(j)\}$$

$$\text{Robot} = \{R_a(k)\} \cup \{R_d(k)\}$$

$$\text{Workpieces} = \{W_a(l)\} \cup \{W_d(l)\}$$

where  $i = \{1 - 9\}$ ,  $j = \{1 - 10\}$ ,  $k = \{1 - 43\}$ ,  $l = \{1 - 10\}$  the corresponding proposals and  $M_{1a}, M_{2a}, B_{1a}, B_{2a}, M_{1d}, M_{2d}, B_{1d}, B_{2d}, R_a, R_d$  the respective allocation and de-allocation places

#### 4.2.2 Collaborative Networks and cooperation mechanism

Based on (Hsieh, 2008), a collaborative network can be put together by a manager obtaining the required resources for the completion of the jobs under its responsibility and taking into account redundant resources that may exist. As an example, consider the execution of the first Job (Workpiece 1) and second Job (Workpiece 2) which respectively correspond to the first and second workflow. In this case, four resources are required—robot for workpiece transfer, turning machine for the first machining operation, the buffer and finally the milling machine for the second operation. On the other hand, the third Job (Workpiece 3) and fourth Job (Workpiece 4) i.e. third and fourth workflows, respectively, require the same resources but with different order in the case of Workpiece 4 and milling instead of turning in the case of Workpiece 3. Therefore, in the case of Managers 3 and 4 there is a need for redundant resources lent by Managers 1 and 2.

A mechanism for the efficient cooperation among the managers and the bidders of the system needs to be applied. The manager performs feasibility analysis of the collaborative network on the basis of a PN composing together bidder's proposal and task workflow expressed in PN markup language (PNML) (<http://www.pnml.org>). The mechanism will be applied through JADE, which is a JAVA programming-based platform compliant with FIPA ([www.fipa.org](http://www.fipa.org)) supporting the development of multi-agent systems. The messages exchanged among agents will have an ACL language-based structure according to FIPA standards. Their structure consists of i) the sender of message, ii) the list of receivers, iii) the communicative purpose and iv) the content of the messages. The platform includes a number of classes which offer considerable opportunities for the user to create a multi-threaded agent-based system. PNML is directly produced by software tools such as PIPE2 (<http://pipe2.sourceforge.net/index.html>) which is used in the current work for building PN models. We transform bidders' proposals in PNML format and then we fill them in the corresponding field. Similarly, managers decode the proposals of bidders in parallel with their internal information to finally connect them together. Communication is efficiently implemented in the framework of Contract Net protocol.

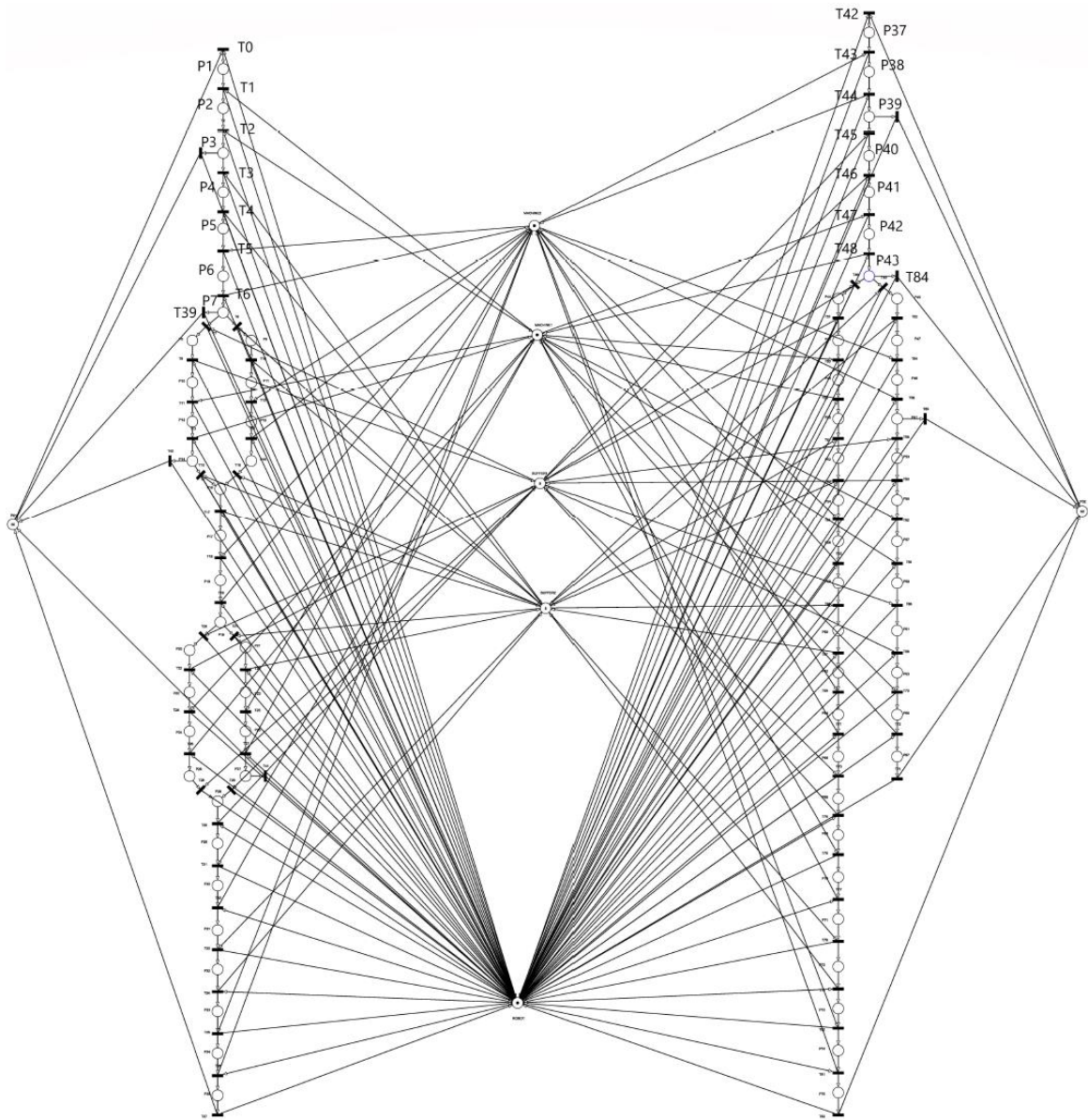
#### 4.2.3 Collaborative Networks as Resource Allocation Systems

A collaborative network can be feasible when its liveness is ensured. From an agent-based perspective bidders indicate the total of system resources and managers indicate the total of workflows. In RAS theory, see Section 2.3, jobs are represented in PNs where resources are included as places. The workflows developed with PNs in this section are in exact correspondence with jobs in RAS theory. In the case of resources, the only difference is that Bidder 1 (Machines) is divided into two separate places indicating the different machines. Based on D-C RAS, the FMS can be represented as in Figure 5, where  $\{P01, P02\}$  represent initial stocks and  $\{P1 - P75\}$  represent operations. Representing the FMS as a RAS facilitates the efficient assurance of its liveness by applying a DAP.

### 4.3 Development of Algebraic Deadlock Avoidance Policy

Development of an Algebraic PK–DAP requires initial determination of the minimal siphons and traps of the D-C RAS, their respective computation being obtained from PIPE2 software.

In our system we have 29 minimal siphons, 22 of which are strict minimal [Data Supplement section 1]. More specifically, siphons  $S_2, S_7, S_9 - S_{26}, S_{28}, S_{29}$  are strict minimal, their respective T- vectors being computed as in the Supplement of this paper [Data Supplement section 2]. From these T- vectors we can computationally discern that there are 11 elementary siphons, namely  $S_2, S_7, S_9, S_{10}, S_{11}, S_{21}, S_{22}, S_{23}, S_{24}, S_{28}, S_{29}$  and the rest of them (11) are dependent. Then, their monitors can be properly added through their complementary siphons [Data Supplement section 3].



**Figure 5.** FMS as D-C RAS

In order to check out emerging deadlocks regarding the operational context of the RAS-based Petri net under study, the HYPENS<sup>TM</sup> tool in MATLAB<sup>TM</sup> was used (Sessego et al., 2008) due to its speed of computation. In addition, it can graphically demonstrate the firing frequency of the transitions of the model. The input and output matrices of the net are extracted through PIPE2<sup>TM</sup> toolbox [Data Supplement section 4]. In order to add the 11 monitors which properly control the elementary siphons [Data Supplement section 4.2] defined above, and also to check the required liveness of the system, we add the T-vectors of the 11 complementary siphons [Data Supplement section 4.3] in the matrices of PIPE2<sup>TM</sup>.

The initial marking of the monitors of the elementary siphons can be confirmed through the implementation of the mathematical formulation of Eq. (7) and the computation of their corresponding control depth variables  $k_{S_i}$ . So, the initial marking of monitors with respect to the initial marking of elementary siphons can be  $M_0(V_{S_i}) = M_0(S_i) - k_{S_i}$ . The LPP which is formulated by Eq. (7) is solved with the use of CPLEX solver on GAMS<sup>TM</sup> (Rosenthal, 2020) The linear dependencies among elementary and dependent siphons can be defined in MATLAB<sup>TM</sup>. The linear programming formulation is as follows:

$$\min Z = k_{S_2} + k_{S_7} + \sum_{i=9}^{11} k_{S_i} + \sum_{i=21}^{24} k_{S_i} + k_{S_{28}} + k_{S_{29}} \quad (8)$$

subject to

$$M(S_i) \geq M(S_2) - k_{S_2}, i = \{6, \dots, 12\}$$

$$M(S_i) \geq M(S_7) - k_{S_7}, i = \{13, 14, 19\}$$

$$M(S_{20}) \geq M(S_{21}) - k_{S_{21}}$$

where:  $1 \leq k_{S_i} \leq M(S_i) - 1, i = \{2, 7, 9 - 11, 21 - 24, 28, 29\}$

We take a feasible optimal solution  $z=11$  and  $k_{S_2} = k_{S_7} = k_{S_9} = k_{S_{10}} = k_{S_{11}} = k_{S_{21}} = k_{S_{22}} = k_{S_{23}} = k_{S_{24}} = k_{S_{28}} = k_{S_{29}} = 1$  from which the following markings of added monitors  $V_{S_i}$  result:

$$M(V_{S_2}) = 1, M(V_{S_7}) = 2, M(V_{S_9}) = M(V_{S_{11}}) = 10, M(V_{S_{10}}) = 18, M(V_{S_{21}}) = M(V_{S_{22}}) = M(V_{S_{23}}) = M(V_{S_{24}}) = M(V_{S_{28}}) = M(V_{S_{29}}) = 7$$

Applying the above solution to the augmented net, by entering the new input and output matrices in HYPENS toolbox, it is observed through firing transitions that partial deadlocks are still formed. Through trial and error method it was found that appropriate increase of control depth variables of the monitors with the highest initial markings  $V_{S_9}, V_{S_{10}}, V_{S_{11}}$  results in liveness of the controlled net without violating the controllability condition of Eq. (6). The final adapted solution involves:

$$M(V_{S_9}) = M(V_{S_{11}}) = 1, k_{S_9} = k_{S_{11}} = 9, M(V_{S_{10}}) = 1, k_{S_{10}} = 17$$

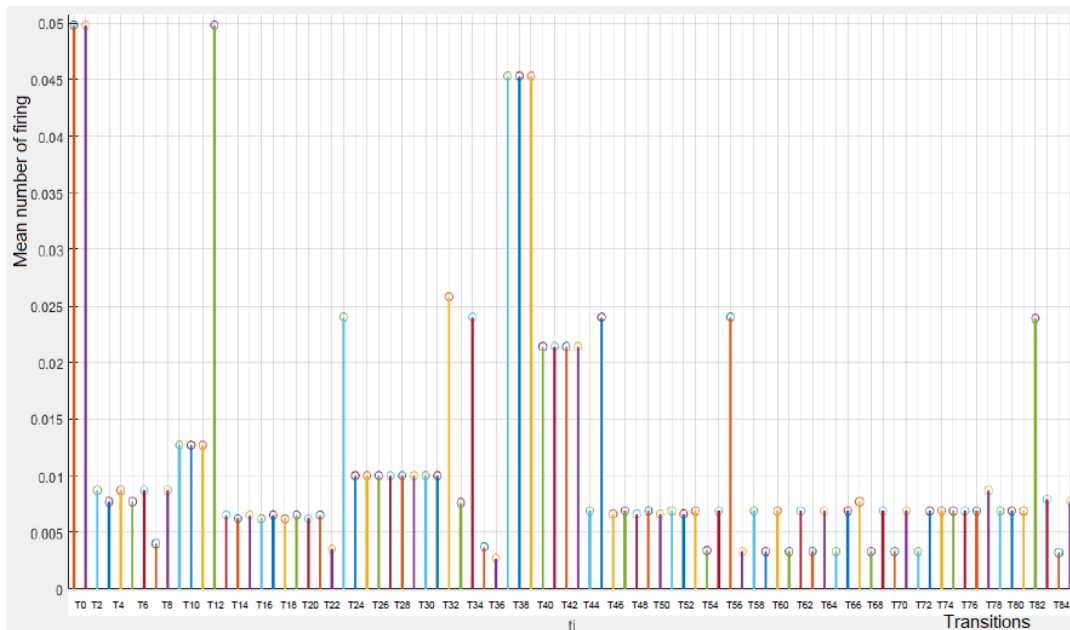
which gives a live and deadlock-free net with firing transitions frequency as shown in Figure 6.

The formed Algebraic PK-DAP can be finally defined as [Supplementary 4.4]:

$$\eta_{V_S} = [\eta_{V_{S_2}} \ \eta_{V_{S_7}} \ \eta_{V_{S_9}} \ \eta_{V_{S_{10}}} \ \eta_{V_{S_{11}}} \ \eta_{V_{S_{21}}} \ \eta_{V_{S_{21}}} \ \eta_{V_{S_{23}}} \ \eta_{V_{S_{24}}} \ \eta_{V_{S_{28}}} \ \eta_{V_{S_{29}}}]^T$$

$$\leq [1 \ 2 \ 1 \ 1 \ 1 \ 7 \ 7 \ 7 \ 7 \ 7 \ 7]^T$$

where  $\eta_{V_S}$  is the T – vector matrix of the monitors of the elementary siphons of size 11 x 86.

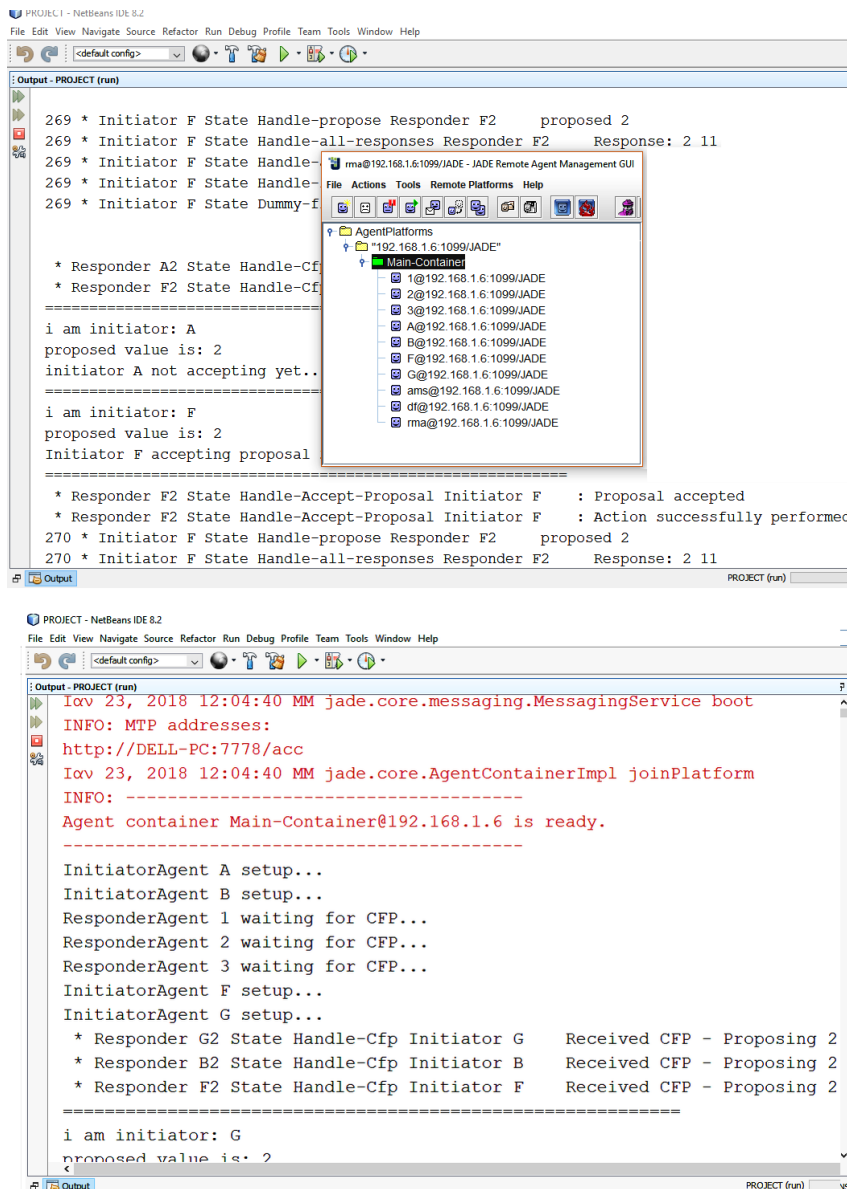


**Figure 6.** Firing Frequency of Transitions

## 5. Implementation, Results and Discussion

In pursuit of the right action sequence referring to the necessary jobs of the physical system, the developed Java classes of Contract Net protocol were embedded in a Sequential Behavior. More particularly, *SequentialBehaviour* class of JADE library was used in order to sequentially apply multiple Contract Net protocols as *subBehaviours* for obtaining the required interaction among managers and bidders.

The involved manager agents are: Initiator A to Q (Manager 1 to 10), whilst bidder agents are: Responder 1 to 3 (Bidder 1 to 3). All agents are created in Main Container of JADE GUI, beginning with bidders and then creating managers by selecting every time the respective JAVA classes, see Figure 7.



**Figure 7.** JADE start-up and Agent Initialization

Creation of managers involves stating their arguments, e.g. Manager 1: Bidder 1, Bidder 2. Final interactions/communications among agents are also defined, e.g. Manager 1 – Bidder 2 – Manager 1 – Bidder 1 – Manager 1 – Bidder 2. The processing cycles of each workpiece are defined in terms of proposals acceptance, e.g.: Initiator A refers to Workpiece 1 corresponding to Acceptance of Proposals: 2 - 1 – 2. The respondents' orders are thus:

A: {2, 1, 2},

- B: {2, 1, 2, 3, 2, 1, 2},
- F: {2, 1, 2},
- G: {2, 1, 2, 3, 2, 1, 2},
- H: {2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2},
- J: {2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2},
- K: {2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2},
- M: {2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2},
- P: {2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2},
- Q: {2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2}

A typical cycle of message responses among managers and bidders is stated as: CFP (Call For Proposal), PROPOSE, ACCEPT PROPOSAL and INFORM (about acceptance) as JADE Sniffer Agent informs during execution of the engaged communications in Figure 8.

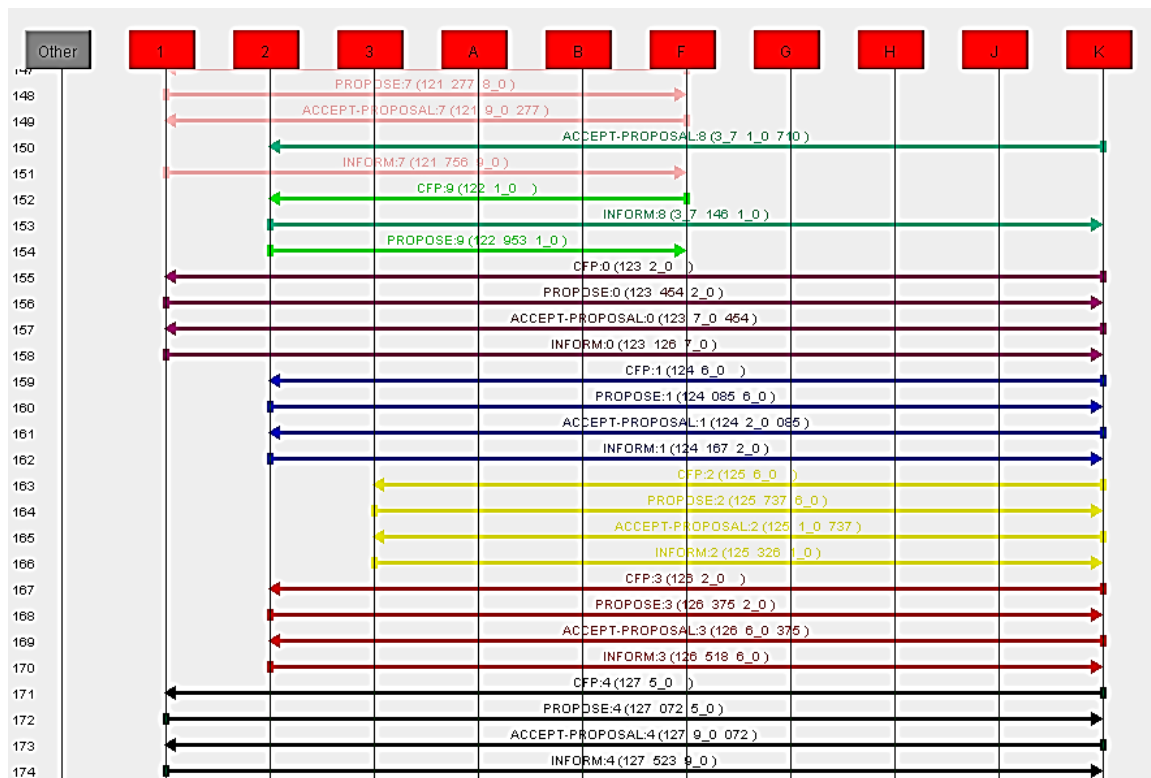
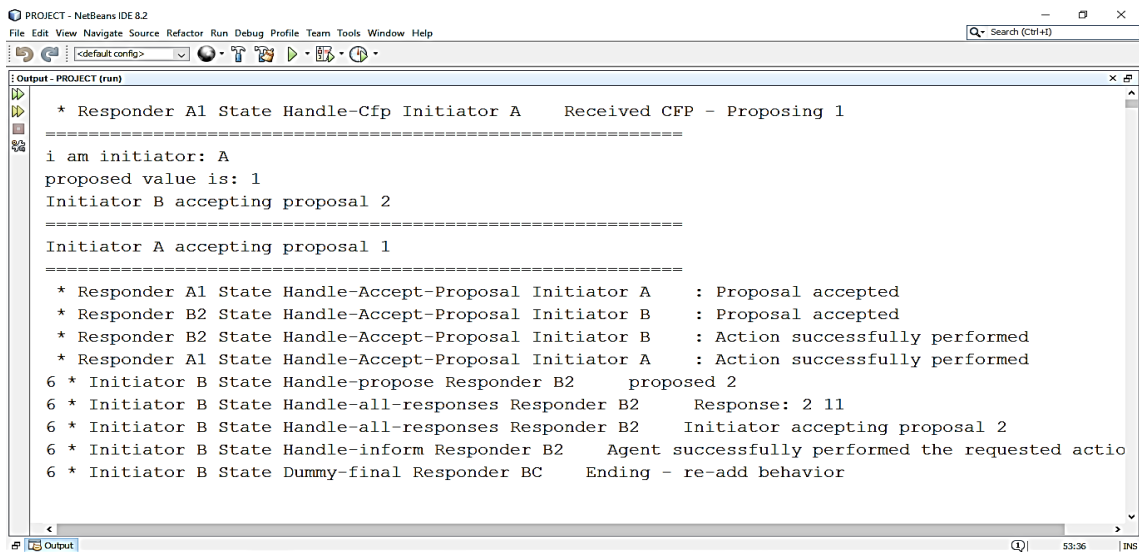


Figure 8. Agent Communication on JADE platform

The initial state of the system is supposed to consist of 5 units per different type of workpiece  $W_i, i = 1, \dots, 10$ . The exact position of the 10 different types of workpieces in the initial stock is not taken into consideration in this study.

Note that no specific priorities among the workpieces are imposed and that further jobs, resources and workpieces could be added taking into account that the proposed methodology strictly refers to interaction protocols among autonomous agents.

Executing the system on JADE platform yields a sequence of events corresponding to workpiece completion based on CNP communications evolution, as indicatively shown in Figure 9.



```

PROJECT - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - PROJECT (run)
* Responder A1 State Handle-Cfp Initiator A Received CFP - Proposing 1
=====
i am initiator: A
proposed value is: 1
Initiator B accepting proposal 2
=====
Initiator A accepting proposal 1
=====
* Responder A1 State Handle-Accept-Proposal Initiator A : Proposal accepted
* Responder B2 State Handle-Accept-Proposal Initiator B : Proposal accepted
* Responder B2 State Handle-Accept-Proposal Initiator B : Action successfully performed
* Responder A1 State Handle-Accept-Proposal Initiator A : Action successfully performed
6 * Initiator B State Handle-propose Responder B2 proposed 2
6 * Initiator B State Handle-all-responses Responder B2 Response: 2 11
6 * Initiator B State Handle-all-responses Responder B2 Initiator accepting proposal 2
6 * Initiator B State Handle-inform Responder B2 Agent successfully performed the requested actio
6 * Initiator B State Dummy-final Responder BC Ending - re-add behavior

```

Figure 9. Example of communication step in Java Output

## 6. Conclusions and further work

Multi Agent Control is an established approach which can be applied successfully to a range of distributed systems. In this paper, development of an agent-based controller for a particular Flexible Manufacturing System, serving as a representative example, is presented.

By applying DACS methodology the agents engaged in the controller were identified. Contract Net protocol was chosen for the development of communication among the agents. The main contribution of this paper focuses on the efficient incorporation of a class of Algebraic Deadlock Avoidance Policies (PK – DAPs) to Contract Net protocol for ensuring feasibility of the pertinent controller. The straightforward result was the achievement of high – level control for the operation of the system imposing the necessary avoidance of undesirable states, notably deadlocks.

Communication and interaction of agents were programmed in JAVA environment in the framework of JADE platform. All the agents were represented in Petri Nets and their message content in XML format.

The Flexible Manufacturing System of this work is based on 10 workflows referring to 10 different workpiece families. Extension of the developed control application to more workflows, different routes, more resources and further complexity considerations can be easily accommodated by the proposed design methodology.

Future research involves incorporation of maximally permissive DAPs to agent interaction protocols of more complex RAS. Moreover, further consideration of various mixtures of workpieces and processing priorities through Colored Petri Nets could contribute to advancing real-time intelligent control for FMS or even resource –bounded systems in general.

## References

- Azizi, A., Poorya-Ghafoorpoor, Y., Al Humairi, A., Alsalmi, M., Al Rashdi, B., Al Zakwani, Z., & ALSheikaili, S. (2018). Applications of control engineering in industry 4.0: utilizing internet of things to design an agent based control architecture for smart material handling system. *International Robotics & Automation Journal*, 4(4), 253–257. <https://doi.org/10.15406/iratj.2018.04.00132>
- Bellifemine, F., Caire, G., Trucco, T., & Rimassa, G. (2003). *JADE programmer's guide, ver. 3.8*.
- Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing multi-agent systems with JADE*. John Wiley.
- Bussmann, S., Jennings, N., & Wooldridge, M. J. (2004). *Multiagent systems for manufacturing control : a design methodology*. Springer.

- Cardin, O., Trentesaux, D., Thomas, A., Castagna, P., Berger, T., & Bril El-Haouzi, H. (2017). Coupling predictive scheduling and reactive control in manufacturing hybrid control architectures: state of the art and future challenges. *Journal of Intelligent Manufacturing*, 28(7), 1503–1517. <https://doi.org/10.1007/s10845-015-1139-0>
- Chen, H. F., Wu, N. Q., Li, Z. W., & Qu, T. (2019). On a maximally permissive deadlock prevention policy for automated manufacturing systems by using resource-oriented Petri nets. *ISA Transactions*, 89, 67–76. <https://doi.org/10.1016/j.isatra.2018.11.025>
- Esmailian, B., Behdad, S., & Wang, B. (2016). The evolution and future of manufacturing: A review. *Journal of Manufacturing Systems*, 39, 79–100. <https://doi.org/10.1016/J.JMSY.2016.03.001>
- Giua, A., & Seatzu, C. (2015). Petri nets for the control of discrete event systems. *Software & Systems Modeling*, 14(2), 693–701. <https://doi.org/10.1007/s10270-014-0425-1>
- Hsieh, F. S. (2008). Holarchy formation and optimization in holonic manufacturing systems with contract net. *Automatica*, 44(4), 959–970. <https://doi.org/10.1016/j.automatica.2007.09.006>
- Jimenez, J.-F., Bekrar, A., Zambrano-Rey, G., Trentesaux, D., & Leitão, P. (2017). Pollux: a dynamic hybrid control architecture for flexible job shop systems. *International Journal of Production Research*, 55(15), 4229–4247. <https://doi.org/10.1080/00207543.2016.1218087>
- Leitão, P., & Karnouskos, S. (2015). *Industrial Agents: Emerging Applications of Software Agents in Industry* (1st ed.). Elsevier.
- Li, Z. W., & Zhou, M. C. (2004). Elementary Siphons of Petri Nets and Their Application to Deadlock Prevention in Flexible Manufacturing Systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans.*, 34(1), 38–51. <https://doi.org/10.1109/TSMCA.2003.820576>
- Li, Z., & Zhou, M. (2009). *Deadlock resolution in automated manufacturing systems : a novel Petri Net approach*. Springer.
- Lin, R., Yu, Z., Shi, X., Dong, L., & Nasr, E. A. (2020). On Multi-step Look-ahead Deadlock Prediction for Automated Manufacturing Systems Based on Petri Nets. *IEEE Access*, 1–1. <https://doi.org/10.1109/access.2020.3022643>
- Luo, J., Liu, Z., & Zhou, M. (2019). A petri net based deadlock avoidance policy for flexible manufacturing systems with assembly operations and multiple resource acquisition. *IEEE Transactions on Industrial Informatics*, 15(6), 3379–3387. <https://doi.org/10.1109/TII.2018.2876343>
- Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W., & Ueda, K. (2016). Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2), 621–641. <https://doi.org/https://doi.org/10.1016/j.cirp.2016.06.005>
- Pérez-Pérez, M., Serrano Bedia, A.-M., López-Fernández, M.-C., & García-Piqueres, G. (2018). Research opportunities on manufacturing flexibility domain: A review and theory-based research agenda. *Journal of Manufacturing Systems*, 48, 9–20. <https://doi.org/10.1016/J.JMSY.2018.05.009>
- Reveliotis, S. (2017). Logical control of complex resource allocation systems. In *Foundations and Trends® in Systems and Control*, vol. 10. Now Publishers.
- Rocha, A. D., Barroca, P., Maso, G. D., & Oliveira, J. B. (2017). Environment to simulate distributed agent based manufacturing systems. In *Studies in Computational Intelligence* (Vol. 694, pp. 405–416). Springer Verlag. [https://doi.org/10.1007/978-3-319-51100-9\\_36](https://doi.org/10.1007/978-3-319-51100-9_36)
- Rosenthal, R. E. (2020). *A GAMS Tutorial*. [https://www.fer.unizg.hr/\\_download/repository/gams\\_prirucnik%5B1%5D.pdf](https://www.fer.unizg.hr/_download/repository/gams_prirucnik%5B1%5D.pdf)
- Sessego, F., Giua, A., & Seatzu, C. (2008). HYPENS: A matlab tool for timed discrete, continuous and hybrid petri nets. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5062 LNCS, 419–428. [https://doi.org/10.1007/978-3-540-68746-7\\_28](https://doi.org/10.1007/978-3-540-68746-7_28)

Suzić, N., Forza, C., Trentin, A., & Anišić, Z. (2018). Implementation guidelines for mass customization: current characteristics and suggestions for improvement. *Production Planning & Control*, 29(10), 856–871. <https://doi.org/10.1080/09537287.2018.1485983>

Wang, L., & Haghghi, A. (2016). Combined strength of holons, agents and function blocks in cyber-physical systems. *Journal of Manufacturing Systems*, 40, 25–34. <https://doi.org/10.1016/j.jmsy.2016.05.002>

Wang, S., Li, D., & Zhang, C. (2016). Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. *Computer Networks*, 101, 158–168. <https://doi.org/10.1016/J.COMNET.2015.12.017>