Blocking Phenomena Analysis for Discrete Event Systems with Failures or Preventive Maintenance Schedules*

Jose Mireles Jr[1], *Member IEEE*, and Frank L. Lewis[2], *Fellow IEEE*.

[1]Instituto de Ingeniería y Tecnología de la Universidad Autónoma de Ciudad Juárez,
Ave. del Charro 450 Nte., Cd. Juárez, Chih. MÉXICO.
[2]Automation & Robotics Research Institute, of The University of Texas at Arlington (UTA)
7300 Jack Newell Blvd. S., Fort Worth, TX 76118-7115, USA.
E-mails: jmireles@arri.uta.edu, flewis@controls.uta.edu

Abstract. ***We present an analysis of possible blocking phenomena, deadlock, in Discrete Event Systems (DES) having corrective and/or Preventive Maintenance Schedules (PMS). Although deadlock avoidance analysis for several classes of DES systems has been widely published, and although different approaches for PMS exist, it is not obvious how to mix deadlock avoidance and maintenance theories to improve throughput. In this paper we show that for some DES structures having reentrant flow lines, it is not necessary to stop activities in the DES, for the case one or more machines in production lines are in PMS. However, PMS may cause deadlock to occur if activities continue in some machines. We propose deadlock-free dispatching rules derived by performing circular wait analysis for possible deadlock situations in systems with PMS. This is accomplished by integrating the PMS structure and failure dynamics into a separate DES system that acts as a disturbance in the primary Reentrant Flow-line DES system. We propose a matrix formulation and a Finite State Machine to synchronize both subsystems.***

Keywords: Deadlock Avoidance, Petri nets, Discrete Event Systems, Reentrant flow lines, Maintenance.

1 Introduction

In this paper we address the problem of avoiding possible deadlock situations on Flexible Manufacturing Systems or Discrete Event Systems (DES) having shared resources in Reentrant Flow-lines [*Kumar 93*], with scheduled maintenance jobs. It is no doubt Preventive Maintenance (PM) is a vital activity for improving machines availability in DES. This improving of availability is due to the decreased number of corrective maintenance jobs in machines, which lead to a much more costly production times. PM methods, like the Reliability-Centered Maintenance method has been used for years, and is still a recommended approach [*Smith 1992*]. Recent studies have proven advantages of using PM techniques. For example, [*Hicks 1990*] has shown improvements in cost-reduction in different Army sites in the state of Texas. In Hicks' work, recommendations are given to keep improving PM schedules. One important recommendation is the search for automated expert systems for optimal use of machines in systems with PM schedules. In this paper, we present one expert system with PM schedules based on matrices that avoids blocking phenomena in reentrant flow-lines. If DES contain Multipart Reentrant flow-lines (MRF), i.e. shared resources perform more than one job for same product, in a system producing several products, and if it is possible not to stop processes, even if one or more machines are in PM, then blocking phenomena can occur if jobs are not correctly sequenced in the remaining non-in-maintenance resources. This blocking phenomena is known as system ***deadlock*** [*Banaszak et al. 90, Hsieh et al. 94, Ezpeleta et al. 95, Fanti et al. 97, Lewis et al. 98*]. Therefore, it is very important that the Discrete Event (DE) controller, after knowing which resources are in PM or corrective maintenance, properly sequences jobs and assigns available resources.

In this paper we restrict our analysis to systems lacking ***key resources*** [*Gurel et al. 00*]. These key resources are critical structured resources that might lead to possible *Second Level Deadlock* (SLD) [*Fanti et al. 00*]. Systems lacking SLD are called regular. In [*Mireles et al. 02*], we provide a matrix tests for system regularity. Based on the decision-making matrix formulation introduced in [*Lewis 92-93*], this paper presents the development of a deadlock-free ***augmented*** discrete event controller for regular MRF systems with failures and PMS. This augmented controller contains a framework capable of handling failures and maintenance-capabilities in the DES structure. We describe the DE controller (DEC) formulation, and show how to analyze and compute in matrix notation the structures needed for deadlock-free dispatching algorithms. Based on these matrix constructions, we integrate PM systems' information for deadlock-free dispatching rules in our augmented DEC matrix formulation by limiting the work-in-progress (WIP) in some critical subsystems, which we define later. This is accomplished by integrating a Finite State Automata system composed of the primary Reentrant Flow-line DES system, and the disturbance-acting PMS structure containing failure dynamics.

* Research supported by ARO Grants DAAD19-00-1-0037 and NSF-CONACyT DMI-0219195.

2 Matrix-Based Discrete Event Controller

A novel Discrete Event Controller (DEC) for manufacturing workcells was described in [*Lewis et al. 93, Mireles et al. 01a-b*]. This DEC is based on matrices, and it was shown to have important advantages in design, flexibility and computer simulation. The definition of the variables of the Discrete Event Controller is as follows. Let *v* be the set of tasks or jobs used in the system, *r* the set of resources that implement/perform the tasks, *u* the set of inputs or parts entering the DES. The DEC Model State Equation is described as

$$\overline{x} = F_v \otimes \overline{v} \oplus F_r \otimes \overline{r} \oplus F_u \otimes \overline{u} \oplus F_{uc} \otimes \overline{u}_C \quad (1)$$

where: $\overline{x}$ is the task or state logical vector, F_v is the job sequencing matrix, F_r is the resource requirements matrix, F_u is the input matrix, F_{uc} is the conflict resolution matrix, and u_c is a conflict resolution vector.

This DEC equation is performed in the AND/OR algebra. That is, multiplication $\otimes$ represents logical "AND," addition $\oplus$ represents logical "OR," and the over-bar means logical negation. From the model state equation, the following four interpretations are obtained. The job sequencing matrix $\boldsymbol{F_v}$ reflects the states to be launched based on the current finished jobs. It is the matrix used by [*Steward 81*] and others and can be written down from the manufacturing Bill of Materials. The resource requirement matrix $\boldsymbol{F_r}$ represents the set of resources needed to fire possible job states this is the matrix used by [*Kusiak et al. 92*]. The input matrix $\boldsymbol{F_u}$ determines initial states fired from the input parts. The conflict resolution matrix $\boldsymbol{F_{uc}}$ prioritizes states launched from the external dispatching input u_C, which has to be derived via some decision making algorithm [*Graves 81*]. The importance of this equation is that it incorporates matrices F_v and F_r, previously used in heuristic manufacturing systems analysis, into a rigorous mathematical framework for DE system computation.

For a complete DEC formulation, one must introduce additional matrices, S_r and S_v, as described next. The state logic obtained from the state equation is used to calculate the jobs to be fired (or task commands), to release resources, and to inform about the final products produced by the system. These three important features are obtained by using the three equations:

Start Equation (task commands) $v_S = S_V \otimes x$ (2)

Resource Release Equation $r_S = S_r \otimes x$ (3)

Product Output Equation $y = S_y \otimes x$ (4)

3 Matrix Analysis of MRF systems

In these sections we present a technique for deadlock-free dispatching for MRF systems with maintenance schedules, and show how to implement some notions from other papers using matrices. First, we integrate PM systems in MRF structures using our matrix approach, then, we determine the deadlock constructions needed for free dispatching. This yields computationally efficient algorithms for analyzing the structure of MRF and deadlock-free dispatching.

Consider the following definition of Multiple Reentrant Flow-lines, which basically define the sort of discrete-part manufacturing systems that can be described by a Petri net. The characteristics of MRF systems are:

- No preemption. A resource cannot be removed from a job until this job is completed.
- Mutual exclusion. A single resource can be used for only one job at a time.
- Hold while waiting. A process holds the resources already allocated to it until it has all resources required to perform a job.

For the DE systems we consider in our analysis, the following are their particularities:

- Each job uses only one resource.
- After each resource executes one job, it is released immediately for its availability.
- In this paper we also consider handling scheduled preventive maintenance, as well as machine failures.

An example of a class of MRF system is given next. Consider the Multipart Reentrant Flow-line problem shown in Figure 1. This system uses two types of machining resources and three types of robotic resources, machine types A and B, and robots type 1, 2 and 3. Any of the (two) robotic resources type 1 move incoming parts P1 and P2 to conveyors C1 and C2 respectively. Any of the (two) robotic resources type 2 can accomplish two jobs, jobs R2a and R2b. Job type R2a moves part type P2 from conveyor C2 to buffer of (any of the two) machines type B. Job type R2b moves machined part type P2 from (any of the two) machines type B to conveyor C3. Any of the (two) robotic resources type 3 can accomplish three jobs, jobs R3a, R3b, and R3c. Job type R3a moves part type P1 from conveyor C1 to buffer of (any of the two) machines type A. Job type R3b moves machined part type P1 from (any of the two) machines type A to parts out P1. Job type R3c moves machined part type P2 from conveyor C3 to parts out P2.

In this example, for simplicity, we are assuming buffer sizes on conveyors and machines equal to one. This assumption will help us emphasize possible deadlock situations when resources are been in failure or scheduled for maintenance. Also, if we consider larger buffers, we will reach a practical point where the buffer might be full and so our same deadlock situation will appear.

3.1 Failure/Maintenance DES structure.

In this section we present an extension of the matrix framework presented in section 2 to incorporate DES systems with Failure and/or PMS. When human operators proceed to fix failures in machines/resources or proceed to perform a preventive maintenance, their jobs can be seen as specific jobs holding such machines/resources. The problem is that holding such resources being in Failure or PMS can lead to system deadlock. Therefore, in order to be able to control a DES with failures and/or maintenance schedules, one has to consider that each of such machine/robotic resources is in one of three possible states: In-Service state, Failure state, or in PM state. Then, for each resource in a PN representation, has to illustrate the Failure and PM states, as in the PN addition system in Figure 2. We call this PN

system the Failure-Maintenance (*FM*) system. In this figure, the places and transitions highlighted as "Ins Service Status" belong to the FMR system, where t_x and t_y represent transitions $\bullet Job_{ij}$ and $Job_{ij}\bullet$, for the j number of jobs from resource R_i. Notice that transition t_{fij} fires when a failure occurs in resource R_i (for i=1,2…n=number of resources) while performing operation Job_{ij}, after finishing this repair job, t_{frij} should be fired (in the PN from figure 2, this can be easily ensured by adding a virtual place between each t_{fij} and t_{frij} transition pairs). Transition t_{mi} will fire when a preventive maintenance M_{pi} for resource R_i is requested. When a transition t_{fij} fires, a failure repair job, F_{repi}, is requested for execution. Maintenance times for jobs type M_{pi} are deterministic times. However, repair time jobs, type F_{repi}, are stochastic and not deterministic, and usually F_{repi} job times are larger than M_{pi} job times. Note that in order to improve throughput, transitions t_{fij} have preference over all others. However, transition t_{mi} is not always an 'urgent' transition to fire due to a scheduled PM, by presence of a new token in place M_{anti}. This is, the supervisor can decide whether it is more important to finish pending jobs, or proceed to maintenance of corresponding resource R_i.

The definition of the Failure-Maintenance system structure follows. Since the structure discussed in section 2 is now augmented by the addition of corrective and PMS, the *FM* structures, we need to re-define the formulation from section 2. For this, we need to include jobs type F_{repi} and M_{previ} (the repair and the maintenance jobs, respectively), and the control transitions that activate these jobs for every type of resources R_i. We include these sets in our now augmented matrix form. We integrate these FM structures by incorporating in matrices F and S the transitions and places shown in figure 3. This figure shows black and gray dots, representing ones and zeros in the rows & columns shown. To properly maintain FM structures, we supervise the maintenance integrated system, and keep track of job markings that belongs to this system. That is, the number of tokens in the *FM* addition system for each resource plus the number of tokens in the job set of same resource is always constant and equal to the initial marking in that resource (assuming no maintenance is in schedule at the time the initial marking is calculated.)

Resource places are the only places shared between PM structures and the original PN (with no PMs). Notice also that for any of these two options, the 'travel' of tokens between one system to the other is through resource places R. Unless, of course, if a failure happens at the moment a machine is performing a job, a token passes from that job to failure status job place F_{repi} (by firing corresponding t_{fij}). For this case, we consider the part was not finished, and stays in standby as a damaged part or for to be re-machined. Then, when failure happens, t_{fij} is fired with high priority and start maintenance failure job type F_{repi}.

This separation of systems MRF and FM is practical for the following reasons:

1) Since *FM* system does not have resource loops and does not generate extra resource loops if exist any in the general existing system, this facilitates deadlock analysis on the MRF system without worrying about dynamics on *FM* systems.
2) It is possible to maintain and control an independent *FM* subsystem with its appropriate PMS, and the existing general system by properly handling the marking vectors from both systems. It is clear that at any given time, the total number of tokens in a job set from a specific resource set, plus the available set of resources from that set is maintained equal to the initial marking of that resource set. This total number of tokens is diminished by one, for every job been in maintenance, i.e. been in its corresponding *FM* system's job set. Then, by maintaining for each resource this number of tokens equal always to the sum of tokens from both systems, it is possible to maintain control of the MRF and *FM* systems.

Figure 2 shows the FM Petri net system structure that has to be added for each resource in the FMR system to supervise preventive and corrective jobs. Figure 3 shows the matrix representation section representing only the FM system of resource R_1. For the class of MRF systems we are considering including *FM*, deadlock can occur only if there is a ***circular wait relation*** among resources [*Deitel 84, Gurel et.al 00*]. Circular wait relations are ubiquitous in reentrant flow-lines and in themselves do not present a problem. However, if a circular wait relation develops into circular blocking, then one has deadlock. But, as long as dispatching is carefully performed, the existence of circular wait relations presents no problem for ***regular systems*** [*Gurel et.al 00*]. In this paper we restrict our analysis to regular systems. This systems lack ***key resources***. These key resources are critical structured resources that might lead to possible *Second Level Deadlock* (SLD) [*Fanti et al. 00*] situations in MRF systems. In [*Mireles et al. 02a-b*], we provide a matrix tests for system regularity.

3.2 Circular Waits: Simple Circular Waits and their Unions.

In this section we present a matrix procedure to identify all circular waits (CW) in MRF systems. CWs are special wait relationships among resources described as follows. Given a set of resources R, for any two resources $r_i, r_j \subset R$, r_i is said to wait for r_j, denoted $r_i \rightarrow r_j$, if the availability of r_i is an immediate requirement to release r_j, or equivalently, if there exists at least one transition $x \in \bullet r_i \cap r_i \bullet$. Circular waits among resources are a set of resources r_a, r_b,…r_w, which wait relationships among them are $r_a \rightarrow r_b \rightarrow \ldots \rightarrow r_w$, and $r_w \rightarrow r_a$. The simple Circular Waits (sCW), are primitive CWs which do not contain other CWs. If sCW are present in the PN system structure, these are identified by constructing a ***digraph*** of resources. [*Hyenbo 95*] demonstrated a technique to identify such sCW. We used his approach to contruct digraphs in matrix form. The entire set of CWs are the sCW plus the circular waits composed of unions of non-disjoint sCW (unions through shared resources among sCW.)

In [*Mireles et al. 01*], we obtained two matrices, C_{out} and G, using digraph theory and string algegra. C_{out} provides

the set of resources which compose every CW (in rows), that is, an entry of 'one' on every (i,j) position means that resource j is included in the i^{th} CW. G provides the set of composed CWs (rows) from unions of sCW (columns), that is, an entry of 'one' on every (i,j) position means that j^{th} sCW is included in the i^{th} composed CW.

3.3 Deadlock Analysis: Identifying Critical Siphons and Critical Subsystems.

Three important sets associated with the CWs C are the ***siphon-job*** sets $J_s(C)$, the ***critical siphons***, $S_c(C)$, and ***critical subsystems***, $J_o(C)$. The critical siphon of a CW is the smallest siphon containing the CW. Note that if the critical siphon ever becomes empty, the CW can never again receive any tokens. This is, the CW has become a circular blocking. The siphon-job set, $J_s(C)$, is the set of jobs which, when added to the set of resources contained in CW C, yields the critical siphon. The critical siphons of that CW C are the conjunction of sets $J_s(C)$ and C. The critical subsystems of the CW C, are the ***job sets*** J(C) from that C not contained in the siphon-job set $J_s(C)$ of C. That is $J_o(C)= J(C)\setminus J_s(C)$. The job sets of CW C are defined by $J(C) = \cup_{r\in C} J(r)$, for $J(r)=r^{\bullet\bullet}\cap J$, where J is the set of all jobs.

In order to implement efficient real-time control of the DES, we need to compute these sets in matrix form. We need intermediate quantities ${}^{\bullet}C$ and $C^{\bullet}$, ***input*** and ***output*** transitions from C, and which in matrix form for each CW are denoted ${}_dC$ and C_d respectively, computed as,

$${}_dC = C_{out}\, S_r, \text{ and} \quad (5)$$

$$C_d = C_{out}\, F_r^{T}. \quad (6)$$

In terms of these constructions, matrix form sets are described next, indicating 'one' on every entry (i,j) for places that belong to that set existing in every i^{th} CW. The job sets described earlier for each CW C, J(C), in matrix form (for all CWs arranged in rows) are described by

$$J_C = {}_dC\, F_v = C_d\, S_v^{T}. \quad (7)$$

The ***siphon-job*** sets are defined for each i^{th} CW C_i as $J_s(C_i)$:= $J(C_i)\cap({}^{\bullet}C \setminus C^{\bullet})$. In matrix notation, we can obtain them for all CWs by

$$J_s = J_C \wedge (\overline{C_d\, F_v}). \quad (8)$$

The ***critical subsystems***, $J_o(C_i) = J(C_i)\setminus J_s(C_i)$, in matrix form for all CWs C_i are obtained by

$$J_o = J_C \wedge (C_d\, Fv). \quad (9)$$

4 Deadlock Avoidance

In terms of the constructions just given, we now present a minimally restrictive resource dispatching policy that guaranties absence of deadlock for multi-part reentrant flow lines. To efficiently implement in real time a DE controller with this dispatching policy we use matrices for all computations. We consider the case where the ***system is regular***, that is, it cannot contain the Critical Resources (CR) (so-called structured bottleneck resources or 'key resources' [*Gurel et al. 00*] existing in Second Level Deadlock (SLD) structures [*Fanti et al. 97, 00*].) For this case, we described in [*Mireles et al. 02*], a mathematical test to verify that MRF systems are regular. If that is not the case, we can still use this matrix formulation, but with a different dispatching policy designed for systems containing second level deadlock structures. We will present such dispatching policy for FMRF systems having CR in a forthcoming work.

4.1 Dispatching Policy

In this section we consider dispatching for regular systems. In [*Lewis et al. 98*] was given a minimally restrictive dispatching policy for regular systems that avoids deadlock for the class of MRF systems considered in this paper, but without the failures or PMS. To understand this policy, note that, for this class of systems, a deadlock is equivalent to a circular blocking (CB). There is a CB if and only if there is an empty circular wait (CW). However, CB is possible (for regular systems) iff the corresponding critical siphon from any CW is empty. This is, there is a deadlock iff all tokens of the CW are in the Critical Subsystem.

Therefore, the key to deadlock avoidance is to ensure that the WIP in the Critical Subsystems is limited to one less job than the total number of initial tokens in the CW (i.e. the total number of resources available in the CW). Preliminary off-line computations using matrices are used to compute the Critical Systems. A supervisor is assigned to each Critical Subsystem (CS) who is responsible for *dynamic dispatching* by counting the jobs in that CS and ensuring that they do not violate the following condition, for each CW C_i,

$$m(J_o(C_i)) < m_o(C_i). \quad (10)$$

That is, the number of enabled places contained in the CS for each C_i must not reach the total number of resources contained in that C_i. In (10), $m_o(C_i)$, is the initial marking of C_i,. However, having failures and PM jobs, the total number of available resources will be diminished. So that $m_o(C_i)$ does not represent anymore the actual available resources contained for that C_i. To be able to keep track of such available resources, we need to define the total number of job places from systems *FM* corresponding to resources contained in a CW C_i, by $J_{MF}(C_i)$. Then, if we diminish $m_o(C_i)$ by jobs currently in failure and/or PM in $J_{MF}(C_i)$, our CB supervision test (10), we will be able to ensure actual available resources which will ensure deadlock-free dispatching. This is, our new CB supervision test is

$$m(J_o(C_i)) < \{m_o(C_i) - J_{MF}(C_i)\} \quad (11)$$

A graphical example of using (11) is pictured in Figure 4. This system has two circular waits, C_1={M1, R_3}, and C2={M2, R_2}. This system contains five *FM* systems split as separate subsystems. Notice that initial $m_o(C_i)$=4 for i=1,2. The current status shown in Figure 4, is that CW C_1 has two jobs pending in $m(J_o(C_1))$=2, jobs R3a and m1. Then, since $J_{MF}(C_1)$=0 (no jobs in places Fm_1, Mm_{p1}, F_{r3}, and M_{p3}), and $m_o(C_1)$=4, we are able to fire transition t_3 to have a total of three tokens allowed by (11). However, since a new attempt to start a PM job at place M_{pm1} is in place, and if we fire transition t_{mm1}, $J_{MF}(C_1)$ will become one, then we should not fire t_3 since C1 would be in deadlock, due to (11). For CW C_2, the allowable number of resources should be $<\{m_o(C_2) - J_{MF}(C_2)\}$. This is, should be smaller that 3. Then, we can not fire transition t_9, since C_2 will get into CB

until failure maintenance $Frep_2$ is finished. Therefore, it is better not to get into blocking and wait till one of the jobs m2 is finished to diminish $m(J_o(C_2))$ by firing t_{11}.

The appropriate way to keep the markings of resources equal in both systems is to use Finite State Automata techniques to supervise both subsystems alternatively. This is, run one (several) discrete event(s) in any one of these subsystems, then hold its markings and pass the new marking of resources *R, m*(*R*), before one run event(s) in the other subsystem. This Finite Element Machine interaction between subsystems is shown in Figure 5.

For implementation of the DEC, in every DE iteration, we can use any desired dispatching policy. For example, FBFS, which maximizes WIP and machine percent utilization. However, it is known that FBFS often results in deadlock. Therefore, we combine FBFS with our new deadlock avoidance test (11). Thus, before we dispatch the FBFS resolution, we must examine the marking outcome with our deadlock policy. If this resulting outcome does not satisfy (11), then the algorithm denies or ***pre-filters*** in real time the firing and we apply again the FBFS conflict resolution strategy for the next possible allowable firing sequence. Then, using FBFS while permitted, we will try to satisfy in most of the current status of the cell the case $m(J_o(Ci)) = \{m_o(C_i) - J_{MF}(C_i)\}$-1. The later condition is an extended policy from that called MAXWIP policy, defined in [*Huang et al. 96*].

6. Conclusions.

We show an analysis of blocking phenomena in Discrete Event Systems (DES) having corrective and/or Preventive Maintenance Schedules (PMS). We show that for some DES structures having reentrant flow-lines, it is not necessary to stop all activities in the DES, for the case one or more machines are in PMS. We proposed deadlock-free dispatching rules derived by performing circular wait analysis for possible deadlock situations. We analyze the so-called critical siphons, certain critical subsystems and resources to develop a DE controller that guarantees deadlock-free dispatching with PMS by limiting the work-in-progress in the critical subsystems associated with each CW. This is accomplished by integrating a Finite State Automata supervision between two subsystems. One system is the Reentrant Flow-line system structure controlled by the DES matrix formulation, and an extra DES system contains the failure and preventive maintenance dynamics, called *FM* system structure. Deadlock-free dispatching is possible by passing the markings of available resources between these two subsystems. The extra *FM* DES system acts as a disturbance in the primary Reentrant Flow-line DES system.

References

[1] Banaszak Z. A. and B. H. Krogh. "Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows." *IEEE Trans. Robotics and Automation*, RA-6, pp. 724-734 (1990).

[2] Ezpeleta S. D., J. M. Colom and J. Martinez. "A Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems." *IEEE Trans. Robotics and Automatio*n, RA-11, pp. 173-184 (1995).

[3] Fanti M.P., B. Maione, S. Mascolo, and B. Turchiano. "Event-Based Feedback Control for Deadlock Avoidance in Flexible Production Systems." IEEE Transactions on Robotics and Automation, Vol. 13, No. 3, June 1997.

[4] Graves S.C. "A Review of Production Scheduling." *Operations Research*, vol. 29, no. 4 (1981).

[5] Gurel A., S. Bogdan, and F.L. Lewis. "Matrix Approach to Deadlock-Free Dispatching in Multi-Class Finite Buffer Flowlines." IEEE Transactions on Automatic Control. Vol. 45, no. 11, Nov. 2000, pp. 2086-2090.

[6] Hicks D.K. "Preventive Maintenance Program: Evaluation and Recommendations for Improvements." U.S. Army Construction Engineering Research Laboratory (USACERL). Report OMB No. 0704-0188. June 1990.

[7] Hsieh F.-S. and S.-C. Chang. "Dispatching-Driven Deadlock avoidance controller Synthesis for Flexible Manufacturing Systems." *IEEE Trans. Robotics and Automation*, RA-11, pp. 196-209 (1994).

[8] Hyuenbo C., T. K. Kumaran, and R. A. Wysk. "Graph-Theoretic Deadlock Detection and Resolution for Flexible Manufacturing Systems." *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 413-421 (1995).

[9] Kumar, P.R. "Re-entrant lines." *Queueing Systems: Theory and Applications*. vol. 13, pp. 87-110, SW, (1993).

[10] Kusiak A. and J. Ahn. "Intelligent Scheduling of Automated Machining Systems." *Computer Integrated Manufacturing Systems*, vol.5, no.1, Feb. 1992, pp. 3-14. UK (1992).

[11] Lewis F. L.. "A Control System Design Philosophy for Discrete Event Manufacturing Systems." *Proc. Int. Symp. Implicit and Nonlinear Systems*, pp. 42-50, TX (1992).

[12] Lewis, F.L., H.-H. Huang and S. Jagannathan. "A systems approach to discrete event controller design for manufacturing systems control." *Proceedings of the 1993 American Control Conference* (IEEE Cat. No.93CH3225-0). American Autom. Control Council. pp.1525-31 vol.2. Evanston, IL, USA (1993).

[13] Lewis F.L., Gurel A, Bogdan S, Docanalp A, Pastravanu OC. "Analysis of Deadlock and Circular Waits using a Matrix Model for Flexible Manufacturing Systems*." Automatica,* vol.34, no.9, Sept. 1998, pp.1083-100. Publisher: Elsevier, UK (1998).

[14] Mireles, J. and F.L. Lewis, "On the Development and Implementation of a Matrix-Based Discrete Event Controller." *MED01, Proceedings of the 9th Mediterranean Conference on Control and Automation.* Pub. on CD, ref MED01-012. June 27-29 2001. Dubrovnik, Croatia (2001).

[15] Mireles, J. and F.L. Lewis. "Intelligent Material Handling: Development and Implementation of a Matrix-Based Discrete Event Controller." *IEEE Transactions on Industrial Electronics*. Vol. 48, No. 6, December 2001.

[16] Mireles, J. and F.L. Lewis, A. Gurel. "Deadlock Avoidance for Manufacturing Multipart Reentrant Flow Lines Using a Matrix-Based Discrete Event Controller." *Int. Journal of Production Research*. 2002.

[17] Smith A.M. "Preventive Impact on Plant Availability." *Proceedings 1992 Annual Reliability and Maintainability Symposium*. pp. 177-180. 1992.

[18] Steward, D. V. "The Design Structure System: A Method for Managing the Design of Complex Systems." *IEEE Trans. On Engineering Management*, vol. EM-28, no. 3, pp. 71-74 (1981).

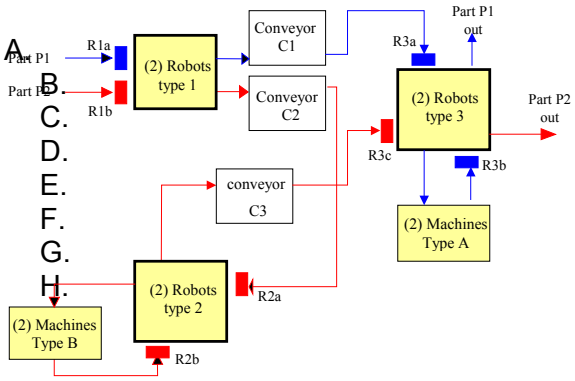

Figure 1. Multipart Reentrant Flow Line Problem.

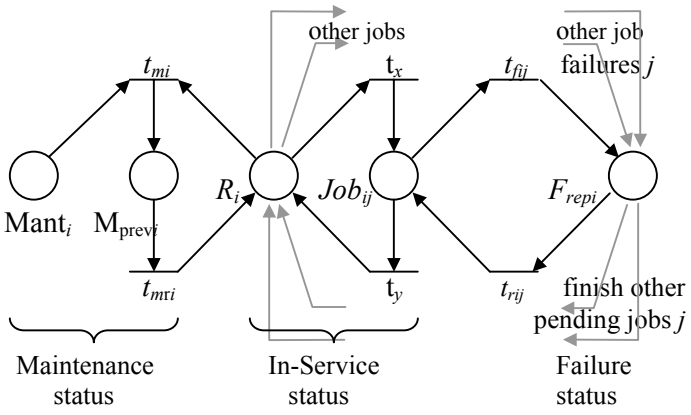

Figure 2. Corrective, In-Service and Preventive status of FM.

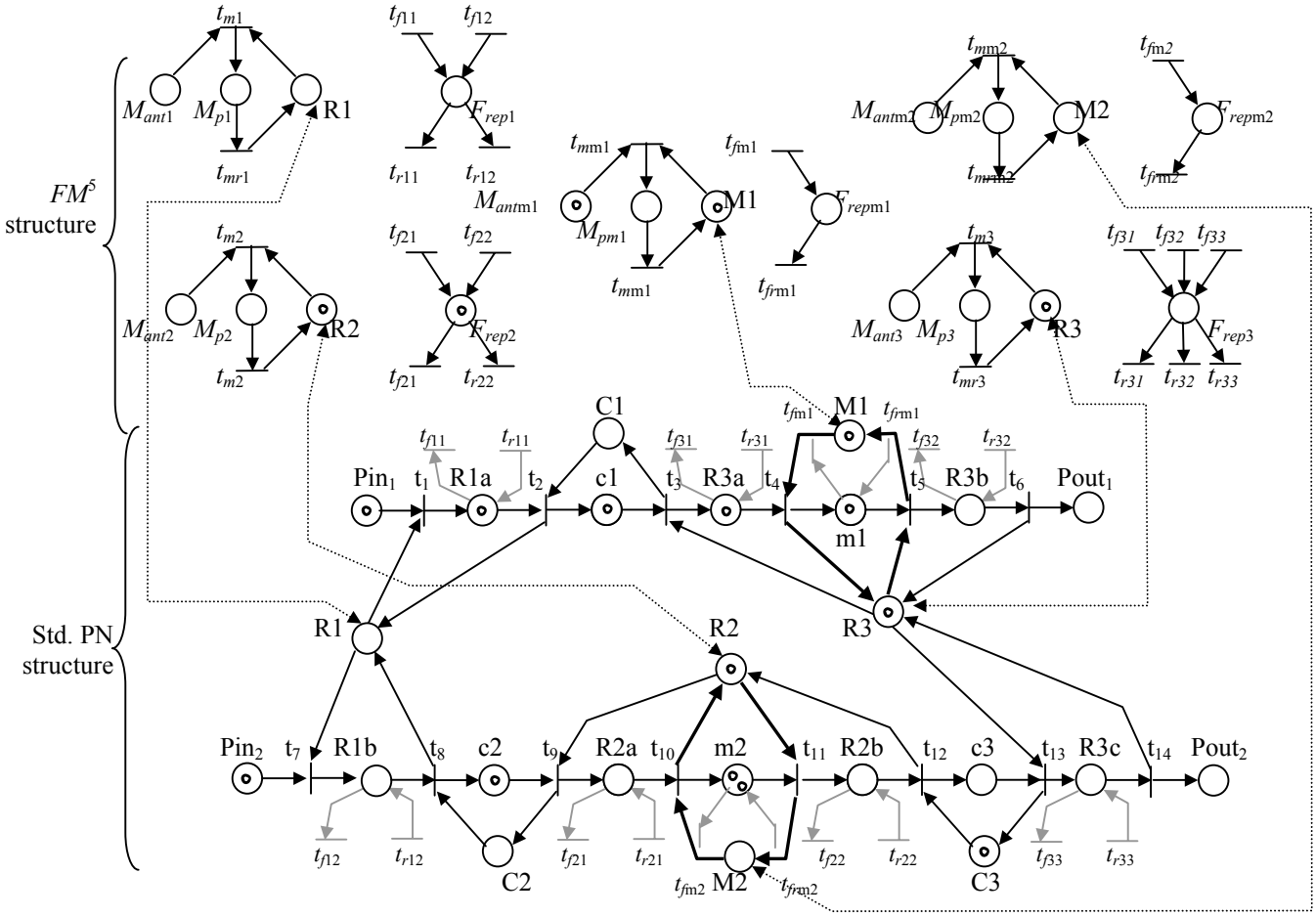

Figure 4. Complete Petri Net and FM^5 system structures.

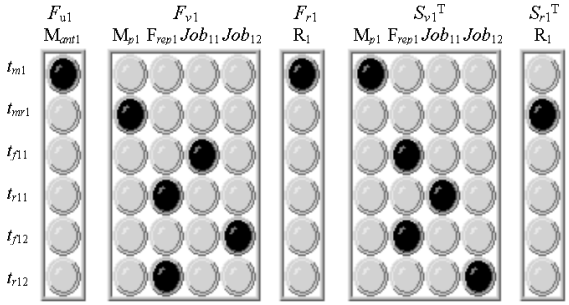

Figure 3. F_u, F_v, F_r, S_v^T, S_r^T matrices for resource R_1

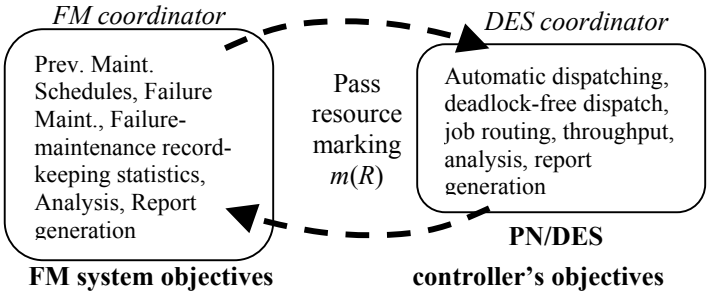

Figure 5. Finite State Automata interactions between the *FM* subsystem and DES controller structure.