Engineer
Vol. - XXXVII, No. - 01, (Oct. 2003/Jan. 2004), Section-I, pp. 20-33, 2004

Design and Implementation of a Maximally Permissive (Optimal) Supervisory Controller for a CIM System – Conveyor Module

Thushara Ekanayake and Devapriya Dewasurendra

Department of Production Engineering, Faculty of Engineering,
University of Peradeniya, Peradeniya,

Abstract: *The control of Integrated Manufacturing Systems poses a number of challenging problems. Most of the systems developed do not attempt to achieve an optimal control in a broad sense. For instance, the individual schedules generated could be "optimal", however the real-time control of the system using this schedule need real-time (scheduling) decisions which are not necessarily based on optimality principles. This on-line control has, at best, been supported by limited look-ahead simulations. The notion of optimality in the sense of this control is itself quite new. The optimality we search for in this context is maximally permissive control introduced in the seminal papers by P. J. Ramadge and W. M. Wonham. This thesis deals with the Design and implementation of a maximally permissive (optimal) supervisory controller for an automated conveyor module of a CIM system. Transition synchronized Petri nets are used to model the dynamic behaviour of the conveyor. RW theory – that introduced by Ramadge and Wonham[-] is used to obtain the optimal supervisor. A generated supervisor is implemented on a conveyor module of a laboratory scale CIM system using an OMRON C200H programmable logic controller.*

Keywords: discrete event systems, Petri nets, supervisor.

1. Introduction

Due to the rapid development of technologies related to material transformation, transportation and communication, manufacturing has become a global activity: if one country can produce a product at a lesser cost than another with comparable or better quality, sourcing the product from the former seems to be the only logical option. Therefore for the product to be competitive in the global market industrialists will certainly have to look for better technologies and management policies. So, FMSs, Computer Integrated Manufacturing (CIM) systems, Agile Manufacturing Systems, Holonic Manufacturing Systems, Biological Manufacturing Systems and other automated manufacturing systems are gradually becoming feasible (and hence competitive) options in today's world.

As automated manufacturing becomes inevitable, people search for new techniques of controlling such systems and supporting theories. Many approaches for developing such controllers have been attempted in the last few decades: based on automata, Petri nets, min-max and other algebras and queuing networks. Among the theories developed in the context of automata, the RW theory, which was introduced in the control literature by P.J.Ramadge and W.M.Wonham in 1987 [1, 2], has some unique features which we

Manuscript received: 07.11.2002
Revised manuscript received: 01.02.2004
Manuscript accepted: 06.02.2004

intend to incorporate into our controller for CIM. It introduces the notion of an optimal supervisor as a minimally restrictive supervisor to control the plant or the system.

A Discrete Event System (DES) is a dynamic system that evolves in accordance with the occurrence, at possibly unknown irregular intervals, of physical events [3]. A few examples (of these events in systems that we deal with) are, starting and finishing the machining of a work piece, transmission of a packet in a communication system, arrival of a job to a manufacturing system, etc.. These systems require control and coordination to ensure the orderly flow of events. The most attractive point in Discrete Event Dynamic Systems (DEDS) theory is that, it separates the conventional concept of uncontrolled dynamics from the feed back controller, i.e. the plant and its controller or the supervisor separately. This permits the formation and solution of a variety of control synthesis problems.

In this paper we try to apply and adapt the above theory to control an existing CIM system. In the event of this implementation we first try to identify the existing system, its hardware and its communication capabilities. We first model the system with transition synchronized Petri-nets with inhibitor arcs and we obtain the automata for the system from the marking graph of the Petri-net and apply RW theory to obtain an optimal supervisor to control the system behavior. Then we implement it on the laboratory scale CIM system in the University of Peradeniya.

In section 2 we discuss underlying theoretical preliminaries. Next, in section 3 we show the generation of the optimal controller and in section 4 we discuss conclusions and future extensions of this work.

2. Preliminaries

2.1 Discrete Event Systems

The following results are particularly relevant to the current work and hence reproduced from [4].

Let Σ be a finite set of distinct symbols $\sigma_1, \sigma_2, \ldots\ldots$, called an *alphabet*. Let Σ^+ be the set of all finite string sequences of the form $\sigma_1\sigma_2\ldots\ldots\sigma_n$ where $n \geq 1$ is arbitrary and $\sigma_i \in \Sigma$. Let us denote the empty sequence as ε, where $\varepsilon \notin \Sigma$. Then,

$$\Sigma^* := \{\varepsilon\} \cup \Sigma^+$$

Elements of Σ^* are called *strings* or *words* over the alphabet Σ and ε, the *empty string*.

Then the *catenation* of strings is defined as

$$\text{cat}: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

such that

$\text{cat}(\varepsilon, s) = \text{cat}(s, \varepsilon) = s, \; s \in \Sigma^*$

$\text{cat}(s, t) = st, \qquad s, t \in \Sigma^+$

Thus ε is the empty string for catenation.

Any subset of Σ^* is called a *language* over Σ, i.e. an element of the power set $Pwr(\Sigma)$. It is clear that the empty language ϕ and Σ^* are also included in the language over Σ. The difference between the empty language ϕ and the empty string ε is that the language will not be empty if it contains the empty string ε. Then a generator is defined as a 5-tuple,
$G = (Q, \Sigma, \delta, q_o, Q_m)$
Where Σ is the alphabet of events, Q is the set of states, q_o is the initial state, Q_m is the set of final or marker states, is a *partial function* (pfn) and $\delta: Q \times \Sigma \rightarrow Q$ (pfn)

The partial function, δ, is extended to event strings as

$$\delta(q, \varepsilon) = q$$
$$\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$$

provided $q' := \delta(q, s)!$ and $\delta(q', \sigma)!$.

$\delta(q, s)!$ means that $\delta(q, s)$ is defined. Next, a state transition graph of a small generator G, which represents a conveyor, is illustrated. The conveyor at any time can become idle (state *I*) by a command to stop (event α), or be moving in the forward

direction (state F) by a command to move in the forward direction (event β) or be moving in the reverse direction (state R) by a command to move in the reverse direction (event γ). The marker state is I.

In the simple generator shown in Figure 1, I, F and R are states and α, β *and* γ are events. The elements of G are thus, $\Sigma = \{\alpha, \beta, \gamma\}$, $Q = \{I, F, R\}$, $q_0 = I$, $Q_m = \{I\}$, and δ is the partial function, which changes a system state to another state within the system as a result of an occurrence of an event.

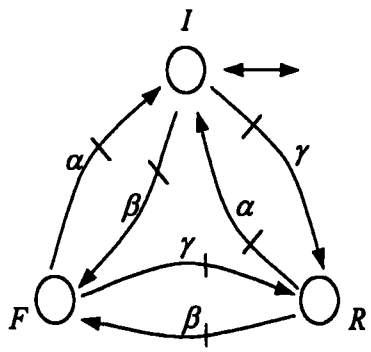


Figure 1: A Simple Generator

The set of words $s \in \Sigma^*$ such that $\delta(q_0, s)!$ is the L(G) which is called the *closed behaviour* of G or the *language generated* by G. Then the subset of strings $s \in L$ (G) such that $\delta(q_0, s) \in Q_m$ is the *marked behaviour* of G or the *language marked* by G, which is denoted by L_m (G).

In the *transition graph* of the generator *states* are represented by circles (o) while *arrows represent events*. The *initial state* is represented by a circle with an entering arrow (→o) and *marker states* by a circle with an exiting arrow (o→). If the initial state is also a marking state then we use a circle with a double arrow (o↔). A transition or an *event* of G is a triple of the form (q, σ, q') where $\delta(q, \sigma) = q'$. Where $q, q' \in Q$ are respectively the *exit state* and the *entrance state* while $\sigma \in \Sigma$ is the event label. The *event label set* of G is Σ.

Some of the events in the alphabet Σ can be controllable (denoted by Σ_c) while the other events are uncontrollable (denoted by Σ_u):

$$\Sigma = \Sigma_c \cup \Sigma_u$$

In the transition graph controllable events are represented by a tick on the arrow (o⟶+⟶) and uncontrollable events without the tick (o⟶).

The *reachable subset* of G is

$$Q_r = \{q \in Q \mid (\exists s \in \Sigma^*)\delta(q_0, s) = q\};$$

G is *reachable* if $Q_r = Q$. The *coreachable subset* is

$$Q_{cr} = \{q \in Q \mid (\exists s \in \Sigma^*)\delta(q_0, s) \in Q_m\};$$

G is *coreachable* if $Q_{cr} = Q$. G is said to be *nonblocking* if every reachable state is coreachable or equivalently L (G) = L_m (G), which means that any string that can be generated by G is a prefix of (i.e. can always be completed to) a marked string of G.

Finally, G is *trim* if it is both reachable and coreachable. Therefore, G trim implies that G is nonblocking, but the converse may not be true. In other wards a nonblocking generator might have nonreachable states (that might or might not be coreachable).

Each is then; subset of enabled events is a *control pattern*; and the set of all control patterns

$$\Gamma = \{\gamma \in Pwr(\Sigma) \mid \gamma \supseteq \Sigma_u\}$$

A *supervisory control* for G is any map V: L (G) → Γ. The pair (G,V) will be written as V/G to suggest 'G under the supervision of V'. The *closed behaviour* of V/G is defined to be the language L (V/G) describe as

i. $\varepsilon \in L$ (V/G)

ii. If $s \in L$ (V/G), $\sigma \in V$ (s) and $s\sigma \in L$ (G) then $s\sigma \in L$ (V/G)

iii. No other strings belong to $L(V/G)$.

$L(V/G)$ is always nonempty and closed because the fact $\{\varepsilon\} \subseteq L(V/G) \subseteq L(G)$ is always true.

The *marked behaviour* of V/G is

$$L_m(V/G) = L(V/G) \cap L_m(G)$$

Hence, the *marked behaviour* of V/G contains the strings that remain after the supervision of V. Then always $\phi \subseteq L_m(V/G) \subseteq L_m(G)$.

V is *nonblocking* over G if

$$\overline{L}_m(V/G) = L(V/G).$$

A language $K \subseteq \Sigma^*$ is *controllable* with respect to G if

$$(\forall s, \sigma) s \in \overline{K}, \sigma \in \Sigma_u \text{ and } s\sigma \in L(G) \Rightarrow s\sigma \in \overline{K}$$

which says that K is controllable if no string in $L(G)$ that is already a prefix of K when followed by an uncontrollable event in G, leaves the prefix set of K: the prefix closure $\overline{K}$ is invariant under the occurrence in G of uncontrollable events.

More concisely, K is controllable if

$$\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$$

It is clear that ϕ, $L(G)$ and Σ^* are always controllable with respect to G. The controllability condition constrains only $\overline{K} \cap L(G)$, since if $s \notin L(G)$ then $s\sigma \notin L(G)$, i.e. the condition [$s \in \overline{K}$ - $L(G)$ and $s\sigma \in L(G)$] is always false.

Theorem 1: Let $K \subseteq L_m(G)$, $K \neq \phi$. There exists a nonblocking supervisory control V for G such that $L_m(V/G) = K$ if and only if

(i) K is controllable with respect to G, and
(ii) K is $L_m(G)$-closed.

The concept of *nonblocking supervisory control* (NSC) has been generalized for the marking behaviour known as the *marking nonblocking supervisory control* (MNSC). Let marking $M \subseteq L_m(G)$. Define a *marking nonblocking supervisory control* (MNSC) for the pair (M, G), as a map $V: L(G) \rightarrow \Gamma$ the same as before; and now the marked behaviour of V/G is defined as

$$L_m(V/G) = L(V/G) \cap M.$$

Theorem 2: Let $K \subseteq L_m(G)$, $K \neq \phi$. There exists an MNSC V for (K,G) such that

$$L_m(V/G) = K$$

if and only if K is controllable with respect to G.

Let $E \subseteq \Sigma^*$. Let us define the set of all sublanguages of E that are controllable with respect to G such that

$C(E) = \{K \subseteq E \mid K \text{ is controllable with respect to G}\}$

as a subset of E. Hence, $C(E)$ is a poset with respect to inclusion. It can be proven[4]; that $C(E)$ is non-empty and is closed under arbitrary unions. $C(E)$ always contains a unique supremal element, which we denote as $sup\ C(E)$.

Theorem 3: Let $E \subseteq \Sigma^*$ be $L_m(G)$-marked, and let $K = \sup C(E \cap L_m(G))$. If $K \neq \phi$, there exists a nonblocking supervisory control (NSC) V for G such that $L_m(V/G) = K$.

Theorem 4: Let $E \subseteq \Sigma^*$ and let $K = \sup C(E \cap L_m(G))$. If $K \neq \phi$ there exists a marking nonblocking supervisory control (MNSC) V for G such that $L_m(V/G) = K$.

Functioning of a Supervisor (Controller)

A controller achieves its objectives by enabling or disabling (activating or deactivating) the controllable events in the system at each state transition.

We say that the controller provides a control pattern at each state transition. A control pattern assigns enable or disable directions to each of the controllable events, $\sigma \in \Sigma_c$.

In order to do this, the controller must track the state changes in the system and at each new state calculate a set of enabled events (and a set of disabled events).

2.2 Petri nets

An ordinary Petri net is defined as a directed graph represented by a 5-tuple

PN = (P, T, *Pre*, *Post*, M_0)
where
- $P = (p_1, p_2, \ldots\ldots\ldots\ldots, p_n)$ is the set of places,
- $T = (t_1, t_2, \ldots\ldots\ldots\ldots, t_m)$ is the set of transitions,
- *Pre* is an input mapping $P \times T \rightarrow \{0, 1\}$, which is called a set of *directed arcs* from P to T.
- *Post* is an output mapping $P \times T \rightarrow \{0, 1\}$, which is called a set of *directed arcs* from T to P.
- M_0 is the initial marking.

Generally, places are used to express the states of the system, while transitions correspond to state evolutions. Petri nets can be represented graphically, which is helpful in both describing how they work and in understanding a particular model easily. A Petri net uses circles to represent places and bars for transitions. The input and output functions are represented by directed arcs between places and transitions. If an arc is directed from a place to a transition then that place is called an input place of that transition. Similarly, if an arc is directed from a transition to a place then that place is called an output place of that transition.

The *marking* M of a Petri net is a mapping of each place to a non-negative integer representing the number of tokens in that place. Tokens reside in places, and their flow through the net is controlled by transitions. Tokens are represented by dots marked on circles that we use for states.

Marking M is an n-dimensional vector whose i^{th} component $M(Pi)$ represents the number of tokens in the i^{th} place Pi; (i = 1,2, ..., n) and n is the number of places in the Petri net. M_0 denotes the initial marking. In the Petri net in Figure 2, places, transitions and tokens are shown.

The marking vector for the example Petri net shown in Figure 2 is

$$M = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

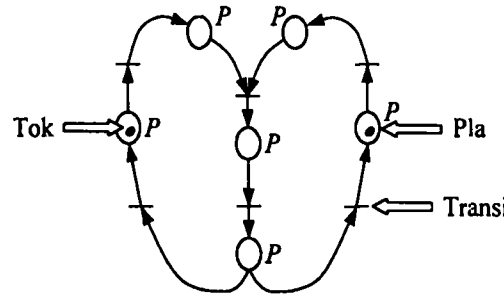


Figure 2 Components of a Petri Net

Inhibitor arc Petri nets add the facility of 'zero *testing*' i.e. the ability to test whether a place has no token. To do this we introduce an *inhibitor arc*. The inhibitor arc connects an input place to a transition and is represented by an arc whose end is marked by a small circle. If an inhibitor arc connects an input place to a transition that transition is only enabled if the input place does not have any tokens. When a transition associated with an inhibitor arc is fired, the marking of the place connecting the inhibitor arc is not changed. Figure 3.2.5 shows an inhibitor arc Petri net, where there are three places $P = \{p1, p2, p3\}$ and one transition $T = \{t1\}$. In the Petri net, the arc $p2 \rightarrow t1$ is an inhibitor arc, i.e. $In(p2, t1)$. The Petri net in Figure 3 (a) is not enabled, because $Pre(p1, t1) = 1$

and $M(p1) = 0$, although $In(p2, t1) = 1$ and $M(p2) = 0$. In Figure 3 (b), transition $t1$ is not enabled, because $In(p2, t1) = 1$ and $M(p2) = 1$, although $Pre(p1, t1) = 1$ and $M(p1) = 1$. The inhibitor arc Petri net in Figure 3 (c) is enabled, because $Pre(p1, t1) = 1$ and $M(p1) = 1$, and, $In(p2, t1) = 1$ and $M(p2) = 0$. When transition $t1$ fires, it removes one token from place $p1$ and deposits one token into place $p3$, as shown in Figure 3 (d). Note that after the firing of transition $t1$, the marking of the place $p2$ remains the same.

2.2.1 Synchronized Petri nets

A Synchronized Petri net is a 7-tuple

$(P, T, Pre, Post, M_0, E, Sync)$
where

- P is a finite set of places,
- T is a finite set of transitions such that $P \cap T = 0$, $P \cup T \neq 0$,
- *Pre* is the input mapping $P \times T \rightarrow N$. *Pre* (Pi, Tj) is the weight of the arc from P_i to T_j. It is non-zero if the arc exists, and 0 if not.
- *Post* is the output mapping $P \times T \rightarrow N$. *Post* (P_i, T_j) is the weight of the arc from T_j to P_i. A transition T_j is validated if there are at least *Pre* (P_i, T_j) tokens in each input place P_i.

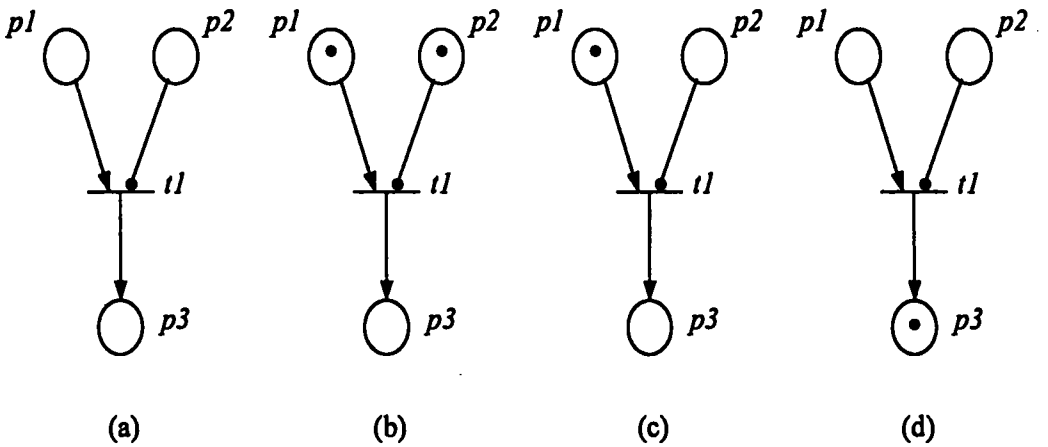


Figure 3: An inhibitor Arc Petri net: (a). Not enabled. (b). Not enabled. (c). (Enabled) before firing. (d). After firing.

- M_0: $P \rightarrow N$ is the initial net marking.
- E is a set of external events.
- *Sync* is a function from the set $E \cup \{e\}$ to transitions T_j in which e is the perpetually activated event (it is the neutral element of the monoid E^*).

The firing of a transition T_j consists of removing $Pre\ (P_i,T_j)$ tokens from each input place P_i and adding $Post\ (P_k,T_j)$ tokens to each output place P_k [5].

3 Optimal Real-Time Control for Collision Prevention in a CIM System.

3.1 Background

Optimal controller design and implementation in Discrete Event Systems has been attempted in the context of automated manufacturing systems over the past several years, particularly after the supervisor control theory proposed by Ramadge and Wonham [2]. The main attraction of this theory has been the introduction of concepts of controllability, observability, optimal control and control under partial observation, in a strict mathematical framework for the computationally unattractive domain of discrete event systems. However, large state spaces that characterize most of the real application domains have impeded rapid application development. In this regard several approaches have been proposed to offset the state space explosion: use of structured and potentially modular-modelling tools like Petri nets has shown some promise. In this attempt of the research we use an approach that uses transition synchronized Petri nets for the system model.

3.2 System Description

In the CIM system under consideration, we have a two way conveyor to transport pallets towards the two five axis robot units, where the robot RV-M2 can load and unload the 3-axis CNC Machining Centre with work pieces and the robot RH-M2 can assemble the given part which is in the storage, onto a machined component. The robot RV-M2 picks only the work-piece to load into the CNC Machining Centre. Hence, the pallet will always be on the conveyor. Also the conveyor can transport pallets into the visual inspection unit where we can determine the colour and the geometry of a work-piece, machined component or the assembled component. Then the conveyor can transport pallets to a given work-piece loading and finished part-unloading stations. A colour sensor attached to the storage unit senses the colour of a part that is to be assembled on to a machined component. Then there are two other storage units where we can store parts to be assembled and on the other we can store work pieces or finished parts. Configuration of the CIM system used to implement the control is shown in Figure 4.

In this context our main target is the conveyor unit. In the conveyor we have altogether six optical sensors. The first is to sense the pallet at the work input position. The second is to sense whether the work exists. Then the third is to check the height of the work piece. Then there is another to sense pallets at the assembling position. The next one is to sense the pallet at the processing position and the last one is to sense the pallet at the visual inspection station. After that we have three proximity sensors called ID sensors to identify each pallet being used. Finally, we have two stoppers to stop pallets in front of the assembling station and the processing station. At any time we consider the 'IN' side of the conveyor as the side where we have the CNC Machining Centre and the other as the 'OUT' side.

A schematic diagram of the conveyor under consideration is shown in Figure 5. For the modelling of the conveyor behaviour, we divided the conveyor into four zones as in Figure 5.

3.3 System Model

Then we model the system with Petri nets as in Figure 6. Notation used in Figure 6 is as follows.

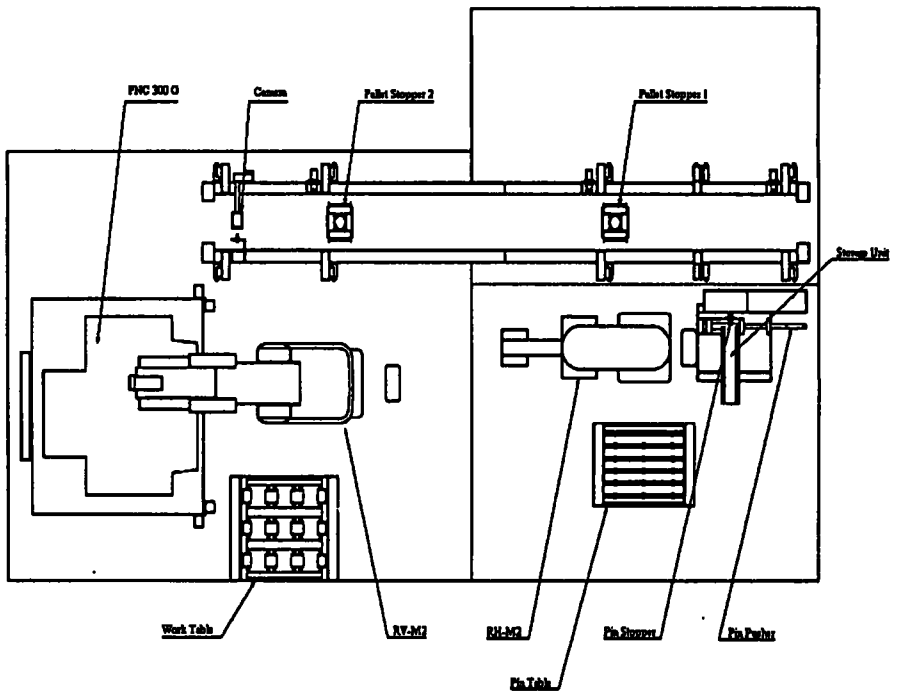
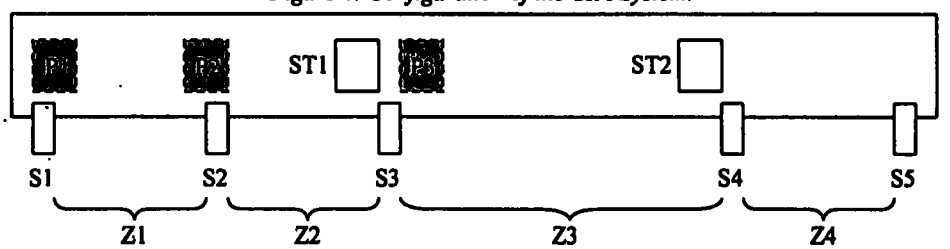


Figure 4: Configuration of the CIM System.



Abbreviation

S1 - Sensor for pallet in.
S2 - Sensor for height check.
S3 - Sensor for assembly.
S4 - Sensor for processing.
S5 - Sensor for visual inspection.

ST1 - Pallet stopper 1
ST2 - Pallet stopper 2

Z1 - Zone 1 – Between S1 & S2.
Z2 - Zone 2 – Between S2 & S3.
Z3 - Zone 3 – Between S3 & S4.
Z4 - Zone 4 – Between S4 & S5.

P1 - Pallet 1
P2 - Pallet 2
P3 - Pallet 3

Figure 5: Different Regions of the Conveyor

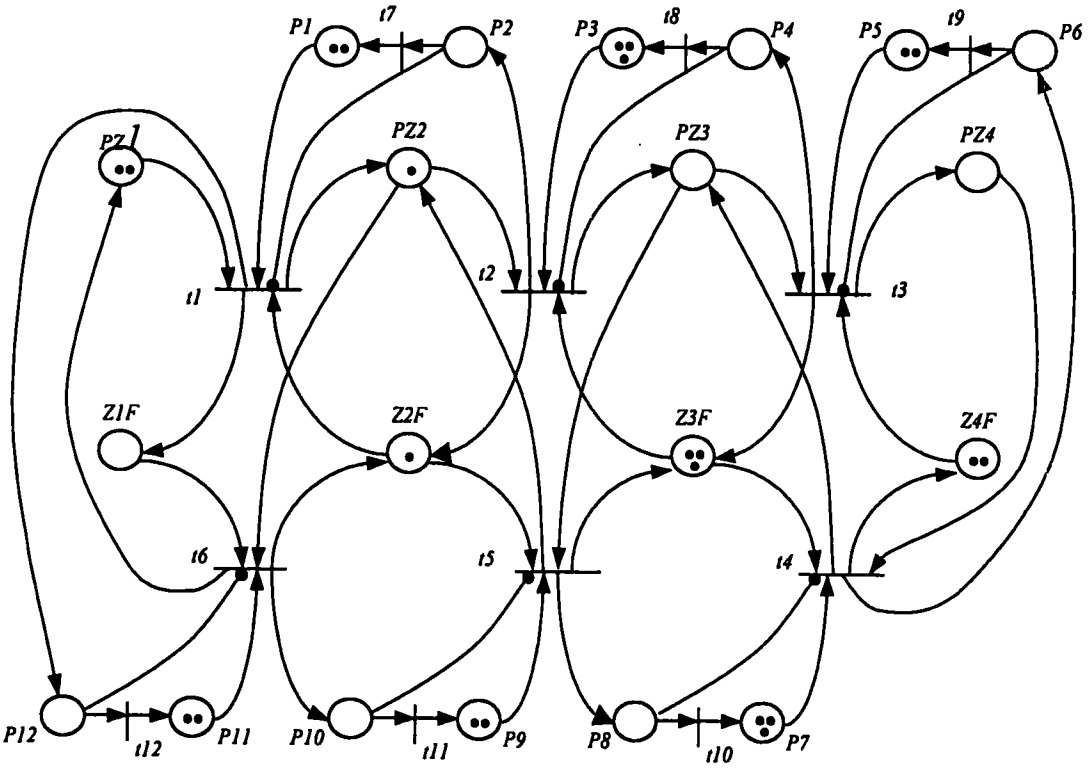


Figure 6: Petri net for the Conveyor

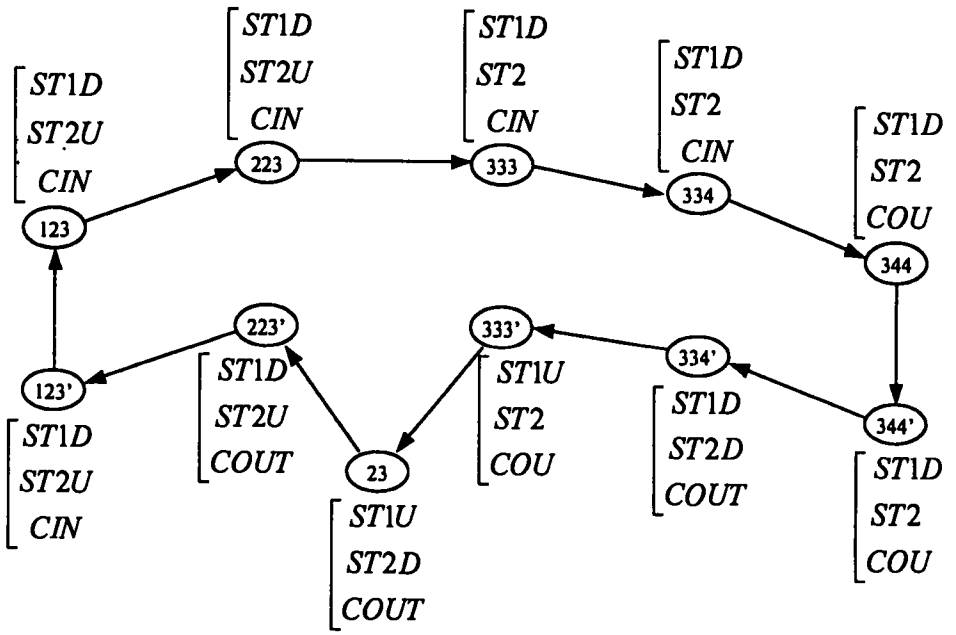


Figure 7: Feasible Controller

Places	*Transitions*
PZ1 - Pallets in Zone 1	t1 - A pallet enters from Zone 1 to Zone2.
PZ2 - Pallets in Zone 2	t2 - A pallet enters from Zone 2 to Zone3.
PZ3 - Pallets in Zone 3	t3 - A pallet enters from Zone 3 to Zone4.
PZ4 - Pallets in Zone 4	t4 - A pallet enters from Zone 4 to Zone3.
Z1F - Free space in Zone 1	t5 - A pallet enters from Zone 3 to Zone2.
Z2F - Free space in Zone 2	t6 - A pallet enters from Zone 2 to Zone1.
Z3F - Free space in Zone 3	*tj* - Transitions between dummy places. (*j* = 7 to 12)
Z4F - Free space in Zone 4	
pi - Dummy states (*i* = 1 to 12) act as pallet counters.	

Capacities of Places

Capacity	Place
1	P2, P4, P6, P8, P10,P12
2	PZ1, PZ2, PZ4, Z1F, Z2F, Z4F, P1, P5, P9, P11
3	PZ3, Z3F, P3, P7

Validation of a model

It is essential to clarify whether the model exactly represents the system considered. To do this depending on the tool used to model the system we can think of a validating procedure. For example if one models the system using Petri nets, as we did here, one can do a token play of the system. To do a token play you take an arbitrary behaviour of state changes in the uncontrolled system and simulate the Petri net manually to see whether the Petri net represents that behaviour in it. Similarly, you can take any behaviuor of the system and do a token play to see whether the Petri net exactly represents the system. Further it is better to see the opposite too, i.e. we can move some tokens in the Petri net and we can see the possibility of the conveyor behaving similar to the Petri net. A complete validation is normally done by developing the marking graph and/or calculating the marking/firing invariants. Another possible method of modelling is to model the system as an automaton. In this case also we can do an operation similar to a token play of a Petri net to see the validity of the model.

3.4 Feasible Controller

A feasible controller was then designed by observing the system, without any consideration to its optimality. This controller avoids collisions; however, it is more restrictive than is really necessary. This feasible controller is shown in Figure 7. To name the states of the automaton we used the following notation.

State XYZ means that the pallet 1 is in Zone X, pallet 2 is in Zone Y and pallet 3 is in Zone Z and the conveyor is moving in the 'IN' direction. Similarly, state XYZ' means that the pallets are at the same zones and moving in the 'OUT' direction. To illustrate this notation consider the state 234'. This means that pallet 1 is in zone 2, pallet 2 is in zone 3 and pallet 3 is in zone 4 and the conveyor is moving in the 'OUT' direction. The control vector for each state is indicated adjacent to the state.
Each state of the automaton in Figure 7 and Figure 8 is corresponds to the marking of the Petri net in Figure 6. For example, state 123 is equivalent to the marking of the Petri net in Figure 6 where we have one token each in places PZ1, PZ2 and PZ3 with conveyor moving to the 'IN' direction. So the pallet 1 is in zone 1, pallet2 is in zone 2 and pallet 3 is in zone 3. Similarly at state 333' we have all the pallets in zone 3 with conveyor moving 'OUT'. In the marking of the Petri net we have three tokens in place PZ3.

3.5 Optimal Real-Time Controller

This marking graph can be interpreted as a language generator. The optimal controller was developed using the methodology outlined in section 2.1 above (the software TCT: [65] was

used for computing the control patterns). The controller is optimal and provides maximally permissive behaviour of the controlled conveyor. The marking graph obtained for the Petri net in the Figure 6 is presented in Figure 8. When we look for the relation between the Petri net in Figure 6 and the marking graph in Figure 8, we see that in Figure 8 we have the marking graph for the Petri net in Figure 6 in a simplified form. Since we are interested only in the number of pallets present in each zone, we have neglected the markings of places P_1 to P_{12} in Figure 6. The notation of the states is similar to that in Figure 7. In this process we use the specification below to get the optimal supervisor.

Specification

- Capacity for each zone is 2 except that the zone 3 which has a capacity of 3. The corresponding DES is shown in Figure 9.

Abbreviation

Event	Corresponding Transition on Figure 8
1	t2
3	t3
5	t4
7	t5
9	CIN
11	COUT
0	t1
2	t6

A specification tells how the system should behave. Usually this is a requirement explained in words. For example this can be a requirement like deadlock avoidance, avoiding a hazardous situation or any thing that falls into the region of shop floor control explained in chapter 2. To convert this specification in words to a form that goes into the control of discrete-event systems, there are no rules or guidelines found up to now. What we can do is to translate it to an automaton or a Petri net, which exactly represents the said specification.

A methodology for the construction of an "optimal" supervisory controller and its design and test on an automated conveyor in the laboratory scale CIM system is presented in this paper. To generate the optimal supervisor from the conveyor model considering the specifications for the system, we used the TCT software [6]. There are 252 states in the optimal supervisor and the software gives us the events that have to be disabled at each supervisor state. Thushara [7] gives the listing obtained from TCT. The feasible controller and the optimal controller were implemented on an OMRON C200H PLC.

The proposed methodology allows progressive development of a system controller; or in other words it helps to add other modules in the system to the controller.

4. Conclusions and Future Extensions.

In this work we tried to apply well-developed theories in the context of Control of Discrete Event Systems to the factory floor. In the first instance we selected a single unit, in this case the automated conveyor, and developed an optimal real time controller to control the conveyor. To use RW theory for designing the controller we had to model it as a generator. Constructing the generator was facilitated by first modelling the system as a Petri net. The marking graph of the Petri net is the designed generator for the conveyor. Then we used RW theory to derive the optimal controller.

In the next step we try to develop a Decentralized Supervisory Controller for an assembly task. For this we add two robots in the system to the conveyor and we try to integrate visual inspection and colour sensors in the system as necessary. Using the same approach in the first part of the research we intend to model each distributed controller with Petri nets and try to optimize individual controllers using RW approach.

Figure 8. Matrix Graph of the Computer

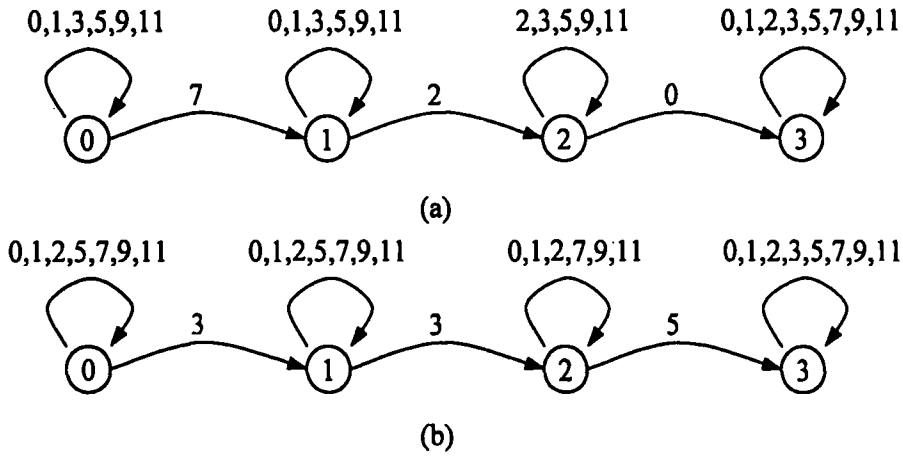


Figure 9: Generators for Specification

References

[2] [Ramadge 1987a] - P.J. Ramadge, and W.M. Wonham, 'Supervision of Discrete Event Processes', Proc. 21st IEEE Conference on Decision and Control, IEEE Control Systems Society, New York, pp. 1228-1229, 1987,

[1] - P.J. Ramadge, and W.M. Wonham, 'Supervisory Control of a Class of Discrete Event Processes', *SIAM Journal of Control and Optimization. Society for Industrial and Applied Mechanics*. Vol. 25, No. 1, 206-230, 1987,

[3] - P.J. Ramadge, and W.M. Wonham, 'The Control of Discrete Event Systems', *Proceedings of the IEEE*, Vol. 77, No. 1, 81-98, 1989.

[5] - Rene David and Hassane Alla, Petri Nets and Grafcet, Tools for modelling discrete event systems, Prentice Hall, 1992.

[6] - TCT. Systems Control Group, University of Toronto, 1995.

[7] - Thushara Ekanayake, 'Design and Implementation of a Maximally Permissive (Optimal) Supervisory Controller for Collision Prevention in a CIM System-Conveyor Module', Postgraduate Diploma Thesis, University of Peradeniya, Sri Lanka, 2002.

[4] - Wonham W.M., 'Notes on Control of Discrete-Event Systems', ECE 1636F/1637S, Department of Electrical and Computer Engineering, University of Toronto, 1999.

Biographical Sketches

THUSHARA EKANAYAKE received the B.Sc.Eng. (Hons) degree in Production Engineering from The University of Peradeniya, Sri Lanka in 1998, P.G.Dip. in Manufacturing Engineering from The University of Peradeniya (awaiting award) and M.Sc. in Computer Integrated Manufacturing from Nanyang Technological University, Singapore in 2004.
From 1998 to 1999 he was an Instructor and since 1999 he has been a Teaching Assistant, with an intervening one year period as full time research student, all in the Department of Production Engineering, University of Peradeniya.

S. DEVAPRIYA DEWASURENDRA received the B.Sc.Eng. (Hons) degree in Mechanical

Engineering from The University of Peradeniya, Sri Lanka in 1978, the M. Eng. Degree in Industrial Engineering & Management from Asian Institute of Technology, Thailand in 1984, D.E.A. and Ph.D. degrees in Automatic Control & signal Processing from E.N.S.I.E.G., Universite Joseph Fourier, Grenoble, France in 1988 and 1992 respectively.

From 1978 to 1983 he worked as Mechanical Engineer in N.W.S. & D.B., Samuel, Sons & Co. Ltd. and Sri Lanka Ports Authority. From 1984 to 1986 he was a Research Associate in the Regional Energy Policy Planning programme (ESCAP/REDP) at The Asian Institute of Technology, Thailand.

Since 1992 Dr. Dewasurendra has been with the Department of Production Engineering at the University of Peradeniya, Sri Lanka as a Senior Lecturer.

From 2002 to 2003 he was a Visiting Professor in the Department of Industrial Engineering & Management, Indian Institute of Technology (Kharagpur), India (on Sabbatical leave from University of Peradeniya) and

From September 2003 till date he has been a Visiting Fellow in The School of Mechanical, Manufacturing & Medical Engineering, Queensland University of Technology, Brisbane, Australia (also on Sabbatical leave from University of Peradeniya). Dr. Dewasurendra is a Chartered Mechanical Engineer, M.I.Mech.E. ('83) (UK) and C. Eng., MIE(SL).