JANUARY 1988 LIDS-P-1739

EXECUTION STRATEGIES FOR PETRI NET SIMULATIONS*

Jean-Louis M. Grevet **
Louise Jandura **
John Brode ***
Alexander H. Levis **

ABSTRACT

Petri Nets can provide the means for modeling and analyzing asynchronous and concurrent operations. In some applications, such as the modeling and analysis of information processing and decision making organizations where different decisionmakers may use different protocols to perform their tasks, it is often necessary to use simulation to study the dynamic behavior of the system. However, when implementing a simulation system on a digital computer capable of handling large scale nets with complex protocols, the automation of the firing process poses problems because a computer executes instructions sequentially. A simulation system based on Predicate Transition nets has been designed which has imbedded in it as choices a number of rules for handling concurrency, confusion, and token colors. These rules may represent either the actual protocols, or ways of handling some model implementation problems. Several examples illustrate the effect of different rules on the execution of the net and on the final markings.

* This work was carried out at the MIT Laboratory for Information and Decision Systems with support provided in part by the Office of Naval Research under contracts nos. N00014-84-K-0519 (NR 649-003) and N00014-85-K-0782 and in part by the National Science Foundation under grant no. SES84-19885.

** Laboratory for Information and Decision Systems, MIT, Cambridge, MA 02139, USA

*** Meta Software Corp., 150 CambridgePark Drive, Cambridge, MA 02140, USA

INTRODUCTION

Petri Nets, which are bipartite directed graphs, can provide the means for modeling and analyzing asynchronous and concurrent operations. They prove particularly useful for the analysis of information processing and decision-making organizations.[1], [2], [3]. Up to now, most of the theoretical developments in which a Petri Net representation has been used have addressed static characteristics of the organizations, e.g., the organizational structure. However, there is a need to investigate the dynamics of decision-making processes and, in order to do so, it is practical to use simulation.

When implementing a simulation system of Petri Nets on a digital computer capable of handling large scale nets with complex protocols, the automation of the firing process presents problems because the computer executes instructions sequentially. It is therefore necessary to define strategies for deciding in which order concurrent events must take place. In some cases, the execution of the net may be very sensitive to this ordering.

This paper introduces a simulation system of Petri Nets, the MIT/SIM system, which has been implemented on an Apple Macintosh Plus using the Design Open Architecture System (OADS) from Meta Software Corporation [4] and which can be applied to organizational design and evaluation problems.

The MIT/SIM system is a discrete-event simulation system and several execution strategies have been incorporated in it in order to automate the execution of the nets of interest. This paper presents these different execution strategies and shows, through the application of the system to specific examples, how they can lead to different final states.

The paper is organized as follows: in the second section, a description on the Petri Net formalism and its application to the modeling and analysis of decision-making organizations is given. The different execution strategies of the simulation system are presented in the third section. In the fourth section, several examples are investigated. Finally, conclusion are presented in the last section.

PETRI NET FORMALISM AND ORGANIZATION THEORY

Ordinary Petri Nets consist of places, transitions, and arcs which connect places to

transitions and vice versa. The state of the system is denoted by its marking, i.e., the tokens present in each place. A transition may fire, if it is enabled - if it has tokens in all its input places. When a transition fires, it removes tokens from its input places and creates tokens in its output places, thus changing the marking of the net. The governing rule for the execution of a Petri Net is that a transition may fire when it is enabled.

An example of Petri Net is shown in Figure 1. Places are represented by circles and transitions by bars. The marking can be illustrated graphically by writing the number of tokens of each place in the circle that represents the place as shown in Fig. 1; empty places are left blank. In this example, transitions t_1, t_2 and t_3 are all enabled.

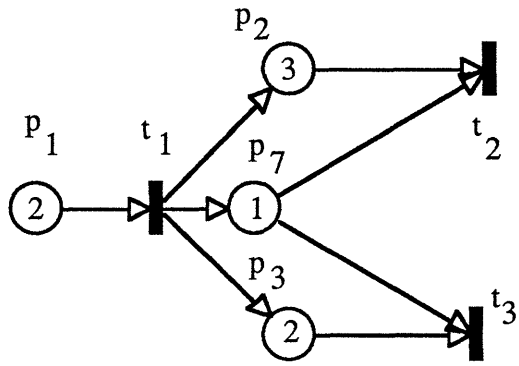

Fig. 1 Marked Petri Net

A Petri Net is a formal model of information flow. The tokens can be considered as symbolic information carriers; the places are the nodes where those tokens can stay without being processed; the transitions are the events that perform some transformation on the information.

When a token enables concurrently several transitions, there is a conflict. In Figure 1, t_2 and t_3 are in conflict because the token in p_7 enables both transitions. A switch is a transition which resolves conflict situations. It is thus a transition with multiple output places and some decision rule according to which each token is routed toward one and only one of the output places. The introduction of a switch s_1 in the Petri Net of Fig. 1 is illustrated in Figure 2.

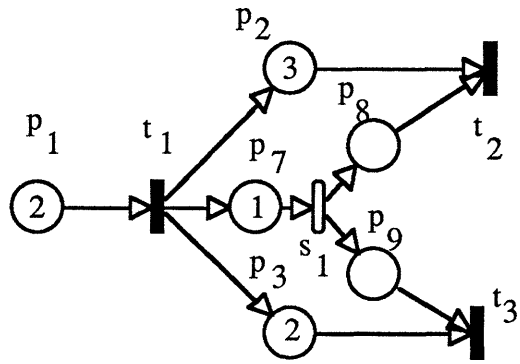

Fig. 2 Petri Net with switch

A Timed Petri Net (TPN) is a Petri Net in which a firing time is associated with each transition of the net. The firing time takes discrete values when the execution is simulated on a computer. A TPN allows the discretization of the process in units of time so that the state of the system can be observed at each instant of time. When a transition initiates its firing, it removes immediately the tokens from the input places, but inserts tokens in the output places only when an amount of time equal to its firing time has elapsed. During the firing of the transition, if the transition is enabled again by the presence of tokens in all its input places, the transition may fire again. One can put a constraint of the firing rules of a Timed Petri Net by allowing transitions to fire only when they are not already executing.

A Predicate Transition Net is a Petri Net where tokens are distinguishable with respect to certain attributes.[5] The rules of execution of the net can take explicitly into consideration the characteristics of the tokens to determine whether or not transitions can be fired.

An information processing and decisionmaking organization consists of a team of decisionmakers that receive and execute tasks, i.e., the organization receives an input x from the external environment and produces a response y. The Petri Net model of the organizations being considered has a unique source place p_{so} where the activation of a token corresponds to the appearance of an event in the environment that must be handled by the organization. The place p_{rs} is a resource place, used to model the limited number of resources that the organization can use to perform the tasks (Fig. 3).

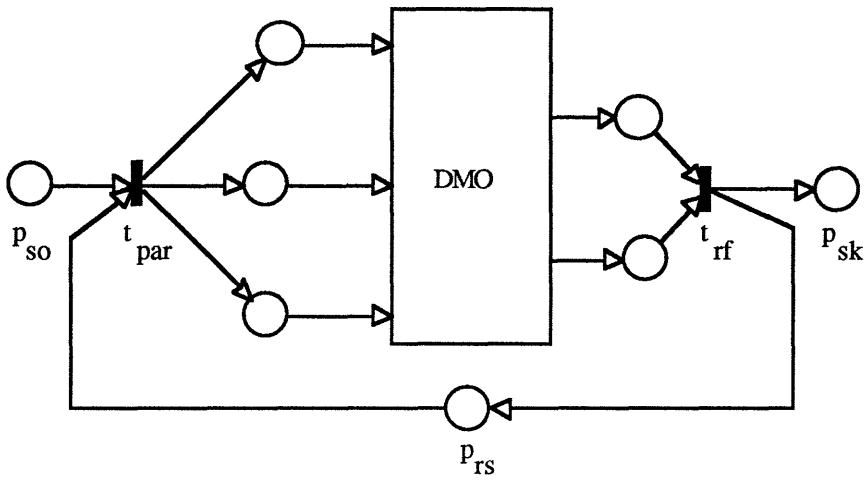

Fig. 3 Interactions DMO - Environment

A basic assumption of the model of coordination in decision-making organizations [6] is that, at any internal stage of the decision-making process, a decisionmaker can discriminate between different items of information on the basis of three characteristics:

- the time T_n at which the inputs that these items of information represent entered the organization.
- the time T_d at which the item of information entered the internal stage where it is currently.
- the class C associated with any item of information by the previous processing stage.

Therefore, each token is assigned a color which corresponds to the triplet (T_n, T_d, C). In this context, *the rule of enablement of transitions is that all its input places have tokens which have the same attribute T_n*. This rule requires that decisionmakers, when interacting, refer to the same event in the environment. The different resources that the organization has are not distinguishable because it is assumed that any organizational resource can be used to process any input. Thus, tokens in resource places have the color ¢ and they are not distinguishable.

SIMULATION OF PETRI NETS

The governing rule for the execution of a Petri Net is that a transition *may* fire when it is enabled. This means that no execution rule requires that a transition enabled by a certain marking must actually fire. Therefore, if one wants to automate the execution of a Petri Net,

rules must be implemented in the system to resolve all the situations where choices have to be made. For instance, if two transitions may be fired concurrently according to the net formalism, it is necessary to order their firing to simulate the same step on the computer. In the same way, when tokens are distinguishable, if a place contains two tokens that enable a certain transition, there must exist some rule to decide which one will be actually removed at the next stage. In some cases, the execution of the net may be very sensitive to these orderings.

In this section, different rules that have been implemented in the MIT/SIM system and that address issues of *conflict, concurrency, and token selection* are introduced.

(i) **conflict issues** : In this case, several transitions are enabled and the firing of one of them will disable the others. A rule must decide which transition will actually fire. For conflict situations modeled with a switch, as shown in Figure 4, five rules have been implemented:

- user selection: the user decides what transition will fire by selecting the corresponding input place.
- random selection: the selection is done randomly.
- probability: the selection is done according to a probability distribution defined in the arcs connecting the switch to its output places.
- priority order: all tokens are routed to the output place of the switch which has not reached its maximum capacity and according to a priority order defined in the arcs connecting the switch to its output places.
- token priority: a token is routed through the arc that has the same priority number as the class of the token. This can be used for simulating Predicate Transition Nets where tokens are assigned a class number.

(ii) concurrency issues : Concurrency occurs when several transitions can fire at the same instant. Since the computer executes the sequence of events linearly, it is necessary to implement rules to decide in which order the firings will be executed. Four rules have been implemented to resolve concurrency issues:

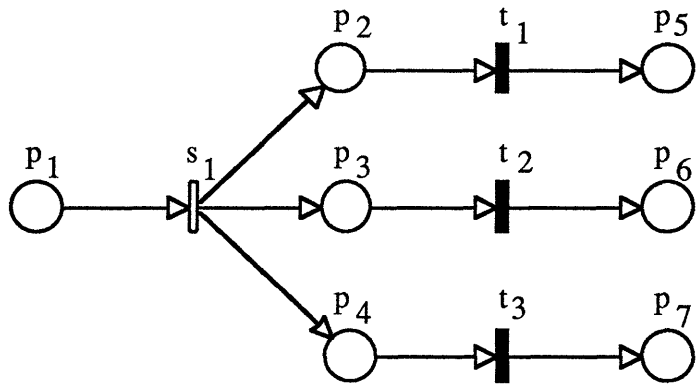

Fig. 4 Petri Net with Switch

- user selection: the user decides what transition will fire next.
- random selection: the selection is done randomly.
- Depth-first: the execution is accomplished by considering first for the next firing the set S of transitions enabled by the tokens produced by the last firing. However, if the firing of these transitions requires the resolution of conflicts involving transitions that do not belong to S, these conflicts must be resolved.
- Breadth-first: the execution requires the firing of all enabled transitions at a given marking M before considering new enablements. If there exists a conflict for these transitions, it must be resolved; otherwise, any conflict involving a transition not enabled for the marking M must not be resolved and the transition enabled for the marking M will fire.

In the example of Figure 5, we assume that p_1 and p_2 both contain a token and that all transitions have zero firing times. Furthermore, we assume that *when concurrency occurs the place with the smallest index is considered first*.

Then, the sequences of transition firings will be the following:

- depth-first: $t_1, t_{11}, t'_{11}, t_{12}, t'_{12}, t_2, t_{21}, t'_{21}, t_{22}, t'_{22}$.
- breadth-first: $t_1, t_2, t_{11}, t_{12}, t_{21}, t_{22}, t'_{11}, t'_{12}, t'_{21}, t'_{22}$.

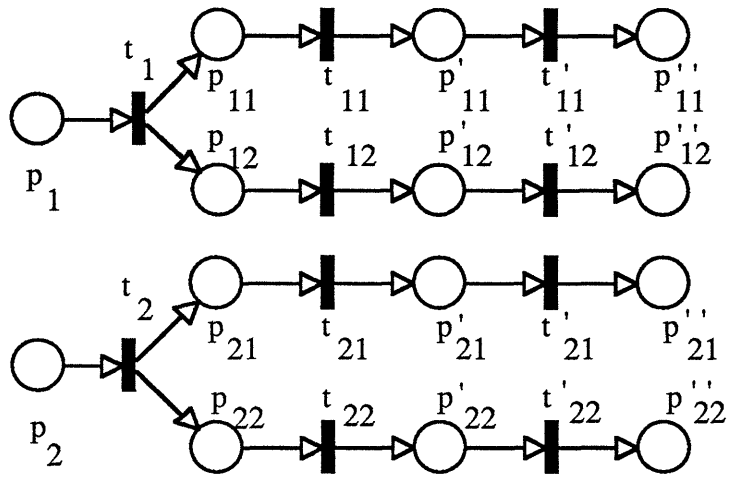

Fig. 5 Depth-first vs. Breadth-first

In this context, a step under the depth-first rule corresponds to the firing of all transitions for a branch, e.g., t_1, t_{11}, t'_{11} in the example above. A step under the breadth-first rule corresponds to the firing of all transitions at a particular instant, e.g., t_1, t_2.

If a non-zero firing time is assigned to these transitions, the depth-first and breadth-first rules are no longer necessary since the time index contributes to the ordering of the sequence of firings that takes place. In Timed Petri Nets, a step corresponds to the firing of all the transitions that must occur at the same time.

(iii) Token selection: Tokens can be marked with attributes on the basis of which one can define selection rules to decide which tokens should be fire when several tokens are in a place. Four types of rules of selection have been considered:

- discriminate with respect to the attribute T_n.
- discriminate with respect to the attribute T_d.
- discriminate with respect to the attribute C.
- combine different rules of the previous types.

The following rules have been implemented as menu options:

1/ FIFO: The token with the lowest T_n is selected.
2/ LIFO: The token with the highest T_n is selected.
3/ LOCAL FIFO (LFIFO): The token with the lowest T_d is selected.
4/ LOCAL LIFO (LLIFO): The token with the highest T_d is selected.
5/ PRIORITY: The selection is done according to priorities based on the attribute C.

Thus, when a place contains several tokens that can be fired by a transition, one of these rules must be implemented to determine which token will be actually removed by the next firing. When tokens are not distinguishable, as in Ordinary Petri Nets, such rules are no longer needed. They are therefore specific to the execution of Predicate Transition Nets.

The execution of a Petri Net with the MIT/SIM system requires that these different rules be defined in advance. The purpose of the next section is to illustrate with three examples the kind of results that are obtained when different rules are used.

APPLICATION

Breadth-first and Depth-first Rules

A Petri Net with its initial marking is shown in Figure 6.

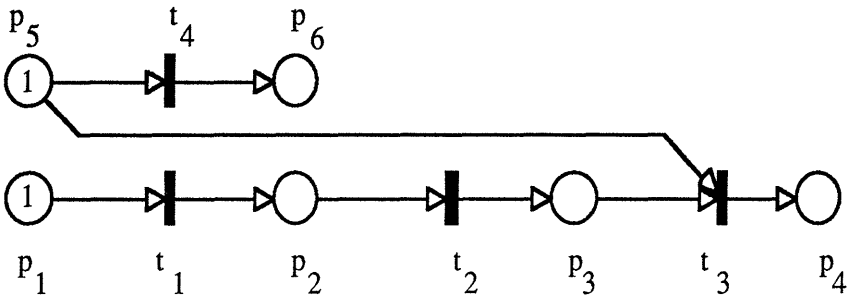

Fig. 6 Initial marking for Example 1

It is assumed that all the transitions in this net have zero firing times which implies that all the firings that can occur are concurrent. The automation of the execution of this Petri Net requires that a rule of ordering of the firings be defined. As described in the previous section, one can implement the breadth-first and depth-first rules.

If we assume that t_1 fires first, the sequence of firings with the breadth-first stragegy is : t_1, t_4, t_2. The transition t_3 cannot fire because the place p_5 no longer contains a token. The first step of the execution corresponds to the firings of transitions t_1 and t_4, i.e., the transitions enabled for the initial marking. The second and final step of the execution corresponds to the firing of transition t_2, i.e., the firing of the unique transition enabled by the marking resulting from the initial step. The final marking of the net is shown in Figure 7.

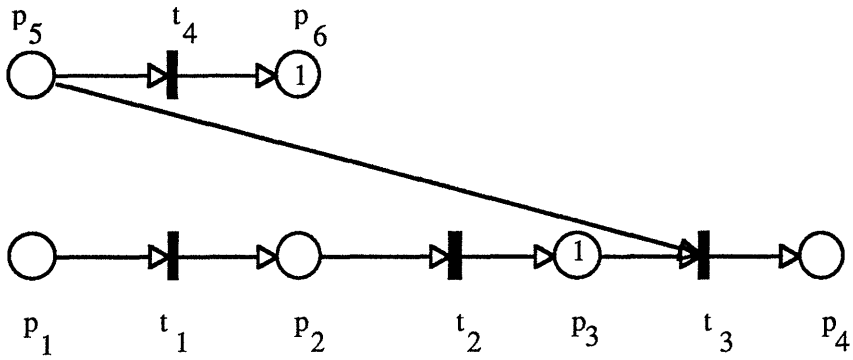

Fig. 7 Final marking with breadth-first rule for Example 1

If the depth-first rule is invoked and if t_1 fires first, the token produced in p_2 is then immediately fired by t_2 which produces a token in p_3. At this stage, a conflict occurs between t_4 and t_3. In accordance with the depth-first rule, this conflict must be resolved. If we assume that t_3 is granted priority, then the final marking is the one shown in Figure 8. The unique step of the process corresponds to the sequence of firings: t_1, t_2, t_3.

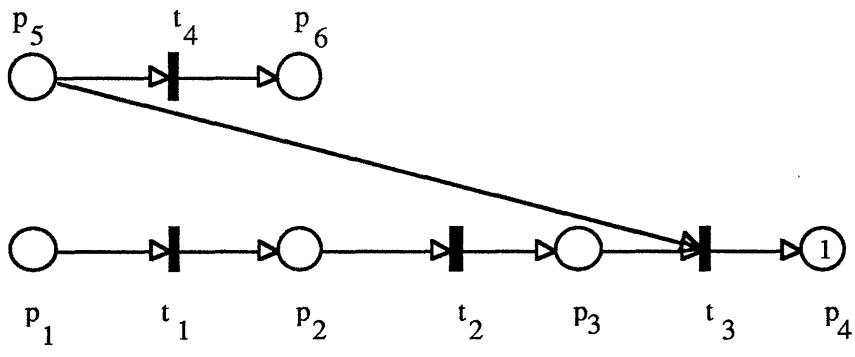

Fig. 8 Final marking with depth-first rule for Example 1

Comparison of the final markings in Figs. 7 and 8 demonstrates that the breadth-first and depth-first rules can lead to different results.

Confusion

The Petri Net of Figure 9 illustrates the occurence of confusion. It is assumed that all transitions have zero firing times.

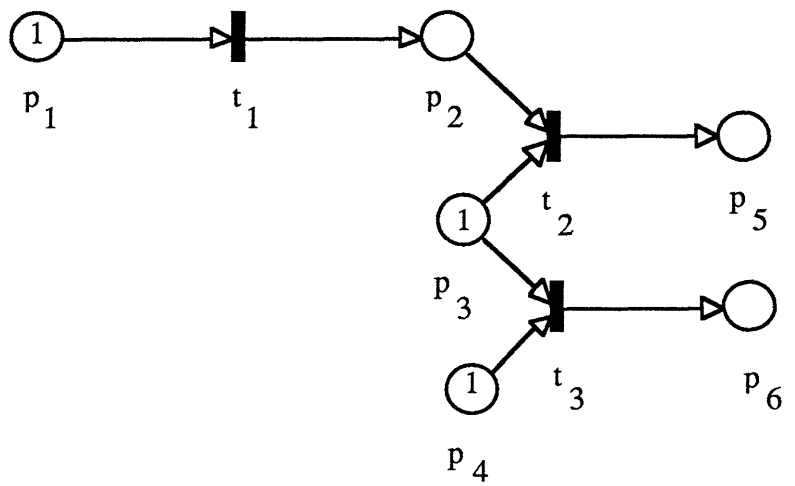

Fig. 9 Example 2: Confusion

With the breadth-first rule, if t_1 fires first, only places p_3 and p_4 are considered for the next firing. No conflict needs to be resolved and t_3 fires; the final marking is shown in Figure 10. The unique step corresponds to the firing of t_1 and t_3.

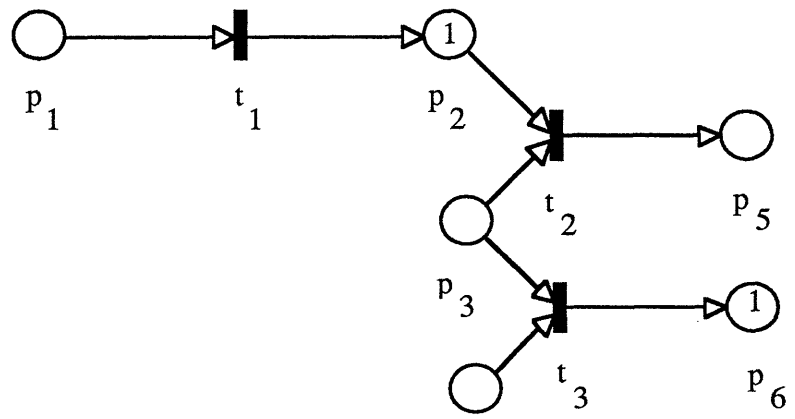

Fig. 10 Example 2 - final marking with breadth-first rule

If the adopted rule is depth-first, the place p_2 is considered first for the next firing: it is possible to implement the *pull-out* strategy, i.e., transition t_2 will fire without any conflict resolution. The final marking of the net is illustrated in Figure 11.

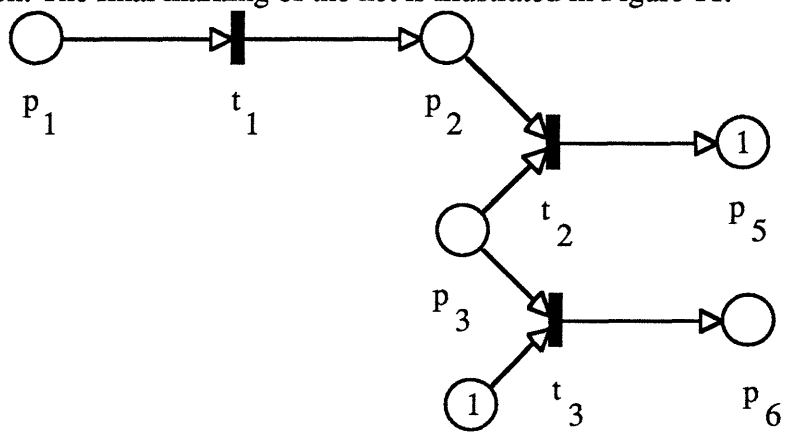

Fig.11 Example 2 -- final marking with depth-first and pull-out rule

The unique step in this case corresponds to the firing of t_1 and t_2. If the pull-out rule is not implemented, the conflict between t_2 and t_3, and the final marking will correspond to one of the two situations illustrated in Figures 10 and 11.This example shows again that the depth-first and breadth-first rules can lead to different results.

Decision-making Processes

This example concerns the analysis of the dynamics of decision-making processes; the rules implemented are based on token selection. The Petri Net model of the organization considered in this example is shown in Figure 12. The organization consists of two decision-makers who receive information for a common task. The commander DM_2 assesses the data that he receives from the environment by using always the same algorithm. In the same way, the subordinate DM_1 assesses the input from the environment with one algorithm. Then, he sends some information resulting from this assessment to his commander. The latter fuses his own result with this information and, on this basis, produces a command by using always the same algorithm. In turn, this command is sent to the subordinate DM_1. Eventually, DM_1 is responsible for producing a response on the basis of the command that he receives and of the results of his own assessment. This model could describe the relationship between the home office and a branch of a corporation [7].

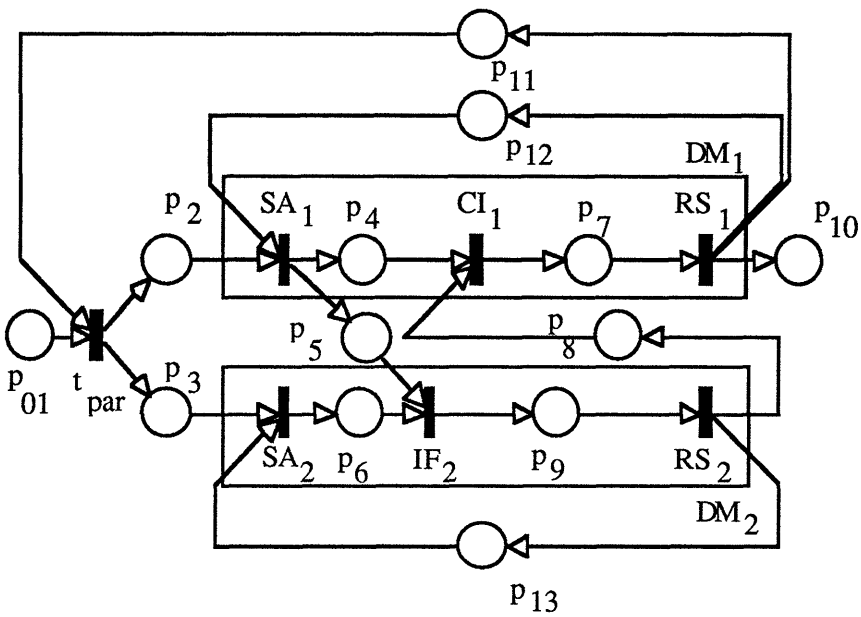

Fig. 12 Petri Net model of two-person organization

In this model, (a) each token representing an item of information is distinguishable with respect to its time of entry in the net, T_n, its time of entry in the place where it stands, T_d, and the class C of inputs associated with it; (b) a rule of selection of tokens, e.g., FIFO or LIFO, is associated with each place; and (c) the rule of enablement of transitions requires that tokens with the same time of entry in the net T_n be in the input places.

The analysis of any Predicate Transition Net that includes time as one of the attributes is a complicated process that depends on the specific grammar used in executing the net. In such a context, simulation may be used to yield insight on the dynamics of the process from three standpoints:

(i) evaluation of measures such as the *throughput rate*, *response time* and *synchronization* for different scenarios. The quality of the performance of an organization is scenario-dependent: Two organizations can exhibit the same performance for certain scenarios and achieve very different levels for other scenarios.

(ii) evaluation of local measures of organizational performance. The dynamics of queues of items of information for different decisionmakers and at different stages of the decision-making process can be observed.

(iii) evaluation of measures over a limited period of time, e.g., during periods of high or low activity.

It will be assumed that in all stages, but the Situation Assessment (SA) stage, the decisionmakers use the LFIFO rule with priority given to the items of information that are in their memory places, i.e., those places that are part of the internal model of the decisionmaker (in this case, p_4 and p_7 for DM_1, and p_6, p_9 for DM_2.) Thus, for transitions IF_2, RS_2, CI_1, RS_1, tokens are fired in the order with which they enter the memory place of their preset.

Different conditions for the SA stages will be considered. Before having assessed any of the inputs that they have received and that they must process, the decision-makers may have to discriminate between them because they cannot perform their assessment on all of them at the same time: only one input can be assessed at a time.

The transitions in this example are associated with non-zero firing times. The processing times of the various stages, measured in some time unit, are presented below; t_{par} denotes the input partitioning stage.

Transition:	t_{par}	SA_1	SA_2	CI_1	IF_2	RS_1	RS_2
Time:	1	10	10	10	10	10	10

The scenario corresponds to an infinite queue of inputs, i.e., to the case where the organization always uses all its resources. The initial marking of the resource places is:

$$M^0(p_{11}) = 4; \quad M^0(p_{12}) = 2; \quad M^0(p_{13}) = 2.$$

case 1: Both SA_1 and SA_2 use the LFIFO rule.

case 2: SA_1 uses the LFIFO rule whereas SA_2 uses LLIFO.

case 3: SA_1 uses the LLIFO rule; SA_2 uses LFIFO.

case 4: Both SA_1 and SA_2 use the LLIFO rule.

The time of entry of an input in the organization, T_i, is the time at which the sensors begin to process it, i.e., in Petri Net formalism the time at which the transition t_{par} fires. The time of leaving from the organization, T_o, is the time at which the organizational response is obtained, i.e., in Petri Net formalism the time at which a token appears in the sink place. The delay, T, is the difference T_o-T_i. The quantity S is a measure of synchronization [6]; it measures the additional time tokens spend in places p_4 and p_6 waiting for the transitions to be enabled. S is measured for each token and is the sum of the values of the measure for IF_2 and CI_1.

The results for T_i , T_o, T as well as for the synchronization S for the first ten inputs which enter the net in each of these four casesare presented in Table 1.

TABLE 1 Synchronization and Delay - Cases 1 to 4

input #	case 1				case 2				case 3				case 4			
	T_i	T_o	T	S	T_i	T_o	T	S	T_i	T_o	T	S	T_i	T_o	T	S
1	0	51	51	20	0	51	51	20	0	51	51	20	0	51	51	20
2	1	61	60	20	1	-	-	-	1	-	-	-	1	-	-	-
3	2	101	99	40	2	101	99	40	2	101	99	20	2	101	99	40
4	3	111	108	40	3	151	148	110	3	131	128	90	3	61	58	20
5	51	151	100	40	51	201	150	90	51	161	110	30	61	151	90	40
6	61	161	100	40	101	251	150	90	101	191	90	30	101	201	100	40
7	101	201	100	40	151	301	150	90	131	221	90	30	151	251	100	40
8	111	211	100	40	201	351	150	90	161	251	90	30	201	301	100	40
9	151	251	100	40	251	401	150	90	191	281	90	30	251	351	100	40
10	161	261	100	40	301	451	150	90	221	311	90	30	301	401	100	40

The results obtained in case 1 are the same as in the case where the tokens have no identity. The steady-state of the process is K-periodic [3] with a period of one. It is reached after the sixth input and is characterized by a constant delay and synchronization. The same conclusions can be drawn in case 2, case 3 and case 4: all three processes are K-periodic with a period of one. In case 2, the steady-state is reached after the fifth input whereas it is reached after the sixth input in case 3 and case 4. However, one can see that the three tokens with attribute $T_n = 1$ are blocked in places p_3, p_4 and p_5, respectively. The processing of the corresponding input is **blocked** as shown in Table 1. This happens because there are always

two tokens in the input places of SA_1 and SA_2 where the LFIFO and LLIFO rules are used.

In the steady-state, the delays for each input are identical in case 1 and case 4. In case 2, this delay increases by 50 percent. In case 3, the delay is reduced by 10 percent. However, since in the situations where a LLIFO rule is used the processing of one input is blocked, the delay for this input is infinite and the organization can use only three resources out of four for the other inputs. Thus, the thoughput rates decrease . When the input represents a threat for which a response must be provided in a certain window of opportunity [8], the LLIFO rule will degrade considerably the accuracy and timeliness of the organization.

In case 1, the synchronization of the organization is equal to 40 units of time in the steady-state. In case 2, it is equal to 90, a considerable degradation. In case 3, the synchronization is equal to 30 units of time in the steady-state. It represents therefore an improvement with respect to case 1. In case 4, the synchronization in the steady-state is the same as in case 1. Nevertheless, one must consider also the individual tokens that are blocked during the processing. In case 2 and case 3, the synchronization for the second token degrades considerably with respect to case 1. Indeed, if DM_1 uses the LFIFO rule and DM_2 the LLIFO rule, the item of information for which the process is blocked is in the input place of the SA stage of the latter, whereas it has been assessed by DM_1 and is in the memory place of his CI stage. Thus, the measure S for this input is infinite. The same situation occurs when DM_1 uses the LLIFO rule and DM_2 the LFIFO rule but, in this case, the degradation of the synchronization is due to the fact that DM_2 waits indefinitely in the IF stage for the data from DM_1 to arrive. In case 4, the second input is also blocked, but the two corresponding tokens remain in the input places of SA_1 and SA_2: it implies that none of the decision-makers will wait for the data from the other member for this input. From this standpoint, the synchronization of the activities for this input does not degrade.

The processing of the inputs in these four cases took place for a configuration in which there were four organizational resources and two resources for each decisionmaker. Table 2 contains the results for cases 1' to 4' in which the organizational resources are increased by one unit:

$$M^0(p_{11}) = 5; \quad M^0(p_{12}) = 2; \quad M^0(p_{13}) = 2.$$

All other conditions remain the same.

TABLE 2 Synchronization and Delay - Cases 1' to 4'

input #	case 1'				case 2'				case 3'				case 4'			
	T_i	T_o	T	S	T_i	T_o	T	S	T_i	T_o	T	S	T_i	T_o	T	S
1	0	51	51	20	0	51	51	20	0	51	51	20	0	51	51	20
2	1	61	60	20	1	-	-	-	1	-	-	-	1	-	-	-
3	2	101	99	40	2	-	-	-	2	-	-	-	2	-	-	-
4	3	111	108	40	3	-	-	-	3	-	-	-	3	101	98	20
5	4	151	147	40	4	-	-	-	4	-	-	-	4	61	57	40
6	51	161	110	40	51	-	-	-	51	-	-	-	61	151	90	40
7	61	201	140	40									101	201	100	40
8	101	211	110	40									151	251	100	40
9	111	251	140	40									201	301	100	40
10	151	261	110	40									251	351	100	40

In case 1', the process is K-periodic of period 2. In the steady-state, the thoughput rate and the synchronization are identical to case 1. In case 4', the process has a period equal to one. The synchronization and thoughput rate are identical to case 4. However, the second and third inputs remain blocked in the input places of the SA transitions. Thus, the organization can use only three out of its five resources to process the remaining inputs. In case 2' and case 3', the performance of the organization is totally degraded by the fact that the whole process is blocked. The execution has reached a **deadlock**, i.e., no transition can fire. As it is shown in Table 2, five inputs remain in the organization which cannot produce a response for any of them.

Because DM_1 and DM_2 do not use the same rules, the two items of information sent by DM_1 to DM_2 after the Situation Assessment stage do not correspond to the inputs that DM_2 is processing. Thus, since DM_2 has to wait for the information that he needs in order to proceed and since DM_1 has to wait for the commands from DM_2 to arrive, the activities of both decision-makers are blocked. This illustrates a situation where the lack of coordination leads to a severe degradation of the effectiveness of the organization.

Figure 13 shows the Petri Net representation of the state of the organization when the deadlock occurs.

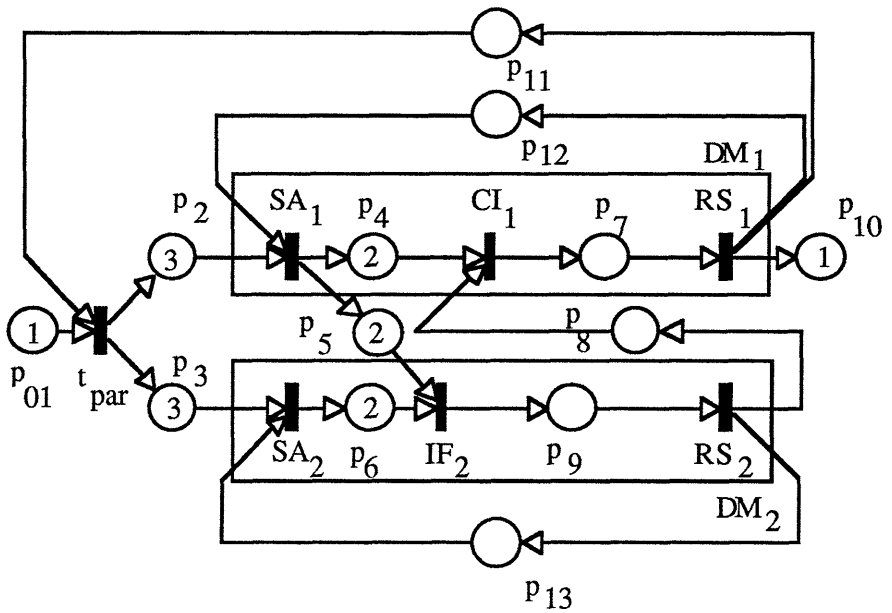

Fig. 13 Two-Person Organization with Deadlock

Places p_5 and p_6 contain tokens that do not have the same attribute T_n, and, consequently, rule 2 of enablement of transition IF_2 is not satisfied. Since the resource places p_{12} and p_{13} are empty, transitions SA_1 and SA_2 cannot fire and the tokens that have the same attribute T_n as the tokens in p_6 are blocked in p_2.

This type of situation would never occur if SA_1 and SA_2 used the LFIFO rule for the sequencing of the inputs: indeed, the interactional transitions would always fire as soon as the places of their preset contain a token, since these tokens would necessarily have the same attribute T_n.

CONCLUSIONS

Problems that appear either when implementing a Petri Net simulation on a digital computer, or modeling decisionmaking organizations with complex protocols have been discussed and alternative approaches to their resolution have been presented. These approaches

have been implemented in the form of rules that can be applied globally or locally at each node of a net. Several examples have been presented to illustrate the effect these rules can have on the execution of a net, its final marking, and its dynamic behavior.

REFERENCES

[1] A. H. Levis, "Information Processing and Decisionmaking Organizations: A Mathematical Description," *Large Scale Systems*, Vol.7, 1984, pp. 151-163.

[2] P. Remy and A. H. Levis, "On the Generation of Organization Architectures Using Petri Nets," *Proc. 8th European Workshop on Applications and Theory of Petri Nets*, Zaragoza, Spain, 24-27 June, 1987.

[3] H. P. Hillion and A. H. Levis, "Timed Event-Graph and Performance Evaluation of Systems," *Proc. 8th European Workshop on Applications and Theory of Petri Nets*, Zaragoza, Spain, 24-27 June, 1987.

[4] Open Architecture Development System Reference Manaual, *Design*, Meta Software Corp., Cambridge, MA, March 1987.

[5] H. J. Genrich and K. Lautenbach, "Systems Modeling with High-Level Petri Nets," *Theoretical Computer Science*, No. 13, pp. 109-136, 1981.

[6] J. L. Grevet, "Decision Aiding and Coordination Decision-Making Organizations," S.M. Thesis, LIDS-TH-1730, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, January 1988.

[7] K.L. Boettcher and A. H. Levis, "Modeling and Analysis of Teams of Interacting Decisionmakers with Bounded Rationality," *Automatica*, Vol. 19, No. 5, November 1983.

[8] P. H. Cothier and A. H. Levis, "Timeliness and Measures of Effectiveness in Command and Control," *IEEE Trans. on Man, Systems, and Cybernetics*, SMC-16, No. 6, Nov./Dec. 1986.