2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Evaluation of the Ability of Petri Net Centralized Implementation Techniques

Ramón Piedrafita. José Luis Villarroel

Abstract— ***In this work we present an evaluation of the ability of interpreted and centralized implementation techniques of Petri nets. The main purpose of this work is the analysis of the performance of the Petri Net execution algorithms, comparing the performance of algorithms through a parameter that does not depend on the execution platform. This parameter is the ability of the implementation that is independent of the execution platform. Firstly there is a study of the size of data structures of the Petri Net execution algorithms. There is also a review of the term ability, defining the ability of the Representing Places technique, therefore allowing a comparative study of the ability of the different techniques***.

I. INTRODUCTION

Petri Nets (PN) is a formalism well suited to model concurrent discrete event systems. It has been satisfactorily applied in fields such as communication networks, computer systems, discrete part manufacturing systems, etc. Net models are often regarded as self-documented specifications, because their graphical nature facilitates communication among designers and users. Moreover, these models are executable and can be used to animate and simulate the behavior of the system and also for monitoring purposes once the system is readily working. The final system can be derived from a Petri Net model by means of hardware and software (code generation) implementation techniques. In this paper we assume that the reader is familiar with the basic concepts of Petri Nets [1].

In the last 25 years, researchers have devoted considerable attention to the software implementation of PN; see for example [2] [3] [4] [5] [6]. A software implementation is a program that triggers the firing of the net transitions, observing the marking evolution rules, i.e., it plays the token game. Depending on the criteria, a Petri net implementation can be classified as compiled or interpreted, sequential or concurrent, and centralized or decentralized.

Manuscript received September 7, 2011. This work was funded by the Discrete Event Systems Engineering Group and the Robotics, Perception and Real Time Group of the University of Zaragoza, Spain.

R. P. Corresponding author is with Aragón Institute for Engineering Research (I3A), Universidad de Zaragoza. 50018, Zaragoza, Spain (34 976 762335; fax: 34 976 761914; e-mail: piedrafi@unizar.es).

J. L. V. Author is with Aragón Institute for Engineering Research (I3A), Universidad de Zaragoza. 50018, Zaragoza, Spain (e-mail: jlvilla@unizar.es).

In a *centralized* implementation, the token player is executed by a single task, which is commonly called the coordinator. In this kind of implementation the algorithm to determine which transitions are enabled and can fire determines its performance. Several algorithms have been proposed in the literature as brute force, place driven or transition driven.

An analysis of centralized implementation algorithms was carried out in [7]. Brute Force (BF), Enabled Transitions (ET), Static Representing Places (SRP) and Dynamic Representing Places (DRP) algorithms were analyzed. The main ideas obtained in [7] are: (1) the implementation of the Enabled Transitions, Static and Dynamic Representing Places algorithms can lead to enormous savings in execution time compared to the Brute Force algorithm; (2) the choice of the most suitable type of algorithm to execute a Petri Net depends on the Petri Net behavior (effective concurrency vs. effective conflicts).

In conclusion, the best algorithm to implement a Petri net depends on its structure and on its dynamic behavior (making and events). Enabled Transitions is better in nets with few conflicts or with conflicts of small size. Representing Places is better in nets with a high number of conflicts or with conflicts of medium-great size.

In a second work [8] we have developed a technique which allows the choice in real time of the most suitable algorithm to execute a Petri Net in accordance with the behavior observed at any time. With this aim in mind, we decided to design a supervisor controller, which we have called Execution Time Controller (ETC). The aim of the ETC is to determine in real time which algorithm executes the Petri Net fastest and to change the execution algorithm when necessary.

In the case of system control, this minimizes the controller reaction time and also the power consumed by the controller. One application of the technique is the minimization of execution time of the Programmable Logic Controllers programs developed in SFC language.

The main purpose of the present work is the analysis of the performance of the Petri Net execution algorithms, analyzing the behavior of the algorithms compared to a parameter which does not depend on the execution platform. This parameter is the ability of the implementation that is independent of the execution platform.

In [9] defines the ability of a PN implementation as a mean number of enabled transitions per unit of time divided by the mean number of transitions analyzed in order to establish their enabling per time unit.

This ratio is a measurement of the number of unnecessary tests carried out in an implementation.

In this paper we present an evaluation of the ability of interpreted and centralized implementation techniques of Petri nets. Firstly there is a study of the size of data structures of the Petri Net execution algorithms. There is also a review of the term ability, defining the ability of the Representing Places technique, therefore allowing a comparative study of the ability of the different techniques [10].

The organization of this paper is as follows. The centralized implementation of PNs is exposed in the Section II. Section III contains a study of the size of the treatment lists and formation of the algorithms and their influence on the behaviour of the algorithms. This first step will allow the completion of the study of the ability of the different techniques. Section IV qualitatively studies the ability of the techniques Enabled Transitions and defines the ability of the Representing Places techniques to make it comparable to the ability of the other techniques; and also defines the static and dynamic abilities. Section V is a quantitative analysis of the ability of the techniques. This analysis is carried out on the Petri Net library, indicating that ability depends on the structure of the net, on the sequence of events and the Representing Places techniques, and on their correct or incorrect selection of representing places. Section VI contains a comparison with the results of the execution time tests, for the purpose of checking and endorsing them. Finally, Section VII includes a summary of the results and contributions of the work.

II. Petri Nets Centralized Implementation: Structure And Algorithms

A Petri net implementation has a strong dependency on the interpretation of the net model, namely, how inputs, actions and code are associated to the net elements.

For example, when PN are used in control applications, the following interpretation is normally adopted [11] :

- Immediate actions are associated to the transition firing (e.g., control signal changes, code execution)
- Level control signals are associated to marked places (if a place has tokens, some signal is raised)
- Predicates are associated with transitions and are additional preconditions for the firing of enabled transitions. Predicates are functions of system inputs or internal variables.

A Petri net with this interpretation is called a synchronized (with its environment) Petri net, or interpreted for control (its typical application).

In this interpretation [3] [9] [12] the full control part is executed by just one task, commonly called token player or Coordinator. Thus, the Coordinator makes the net evolve over time. The Coordinator is commonly an interpreter that works over a data structure that encodes the PN [12].

From a performance point of view, the main action performed by the coordinator is the finding of enabled transitions. Several techniques have been proposed for an efficient search of enabled transitions and subsequently reducing the overload introduced by the coordinator. Depending on the solution chosen, centralized implementation techniques can be classified into any of the following classes [13]:

Place-driven approaches. In the algorithms Static Representing Places and Dynamic Representing Places only the output transitions of some representative marked places are tested [14]. Each transition is represented by one of its input places, the Representing Place. The remaining input places are called synchronization places. Only transitions whose Representing Place is marked are considered as candidates for firing. The Representing Place of a transition can be static or can be dynamically selected as in [15] (in this case it is called triggering place). In the Dynamic Representing Places technique, if a Representing Place is marked but the transition is not enabled due to one or several of the synchronization places being disabled, any one of the unmarked synchronization places will be chosen as the new Representing Place. In this manner we can reduce the number of marked Representing Places and, hence, the number of tested transitions.

Transition-driven approaches. The objective of these techniques, also called “Enabled Transitions”, is that the token player deals only with fully enabled transitions [16]. In order to achieve this objective, a characterization of the enabling of transitions, different from the marking, must be provided taking into account the local structural information of the transitions. [17].

In the present work we have implemented four algorithms in which different enabled transition search techniques are developed:

- Brute Force
- Enabled Transitions.
- Static Representing Places.
- Dynamic Representing Places

However, in this paper we present the results using the better ones: ET and SRP [7].

2.1 Data structures

In the Enabled Transitions technique the following data structures will be available:

- Enabled Transitions List (ETL). Treatment list made up of the transitions with all marked input places.
- Almost Enabled Transitions List (AETL). Formation list which is built with the output transitions of the places marked in the firing of the transitions.

In the Static Representing Places technique, the following data structures will be available:

- Marked Representing Places list (MRPL) and Marked Synchronization Places list (MSPL). Treatment Lists with the marked Representing Places and Synchronization Places.
- Marked Representing Places list next cycle (MRPLnext) and Marked Synchronization Places list next cycle (MSPLnext). Formation Lists with the Representing Places and Synchronization Places that will be marked in the next cycle by the firing of the transitions.

2.2 Algorithm Execution Cycle

Program 1 presents the basic treatment cycle of the Coordinator for the ET technique and Program 2 for the SRP technique. In the programs we can distinguish three phases: (1) enabling analysis and start of firing, and (2) end of transition firing and (3) list update.

In the ET technique, ETL contains all enabled transitions at the beginning of the cycle. From this list the fired transitions and the disabled transitions (effective conflicts) must be extracted in the execution cycle. AETL is built with the output transitions of the places that will receive tokens after the firing of the transitions. When ETL is updated for the next cycle, the enabling of the transitions in AETL is verified.

```
loop forever
  while elements in ETL do
    T = next_element (ETL) ;
    // enabled transition analysis
    if enabled (T) and predicate(T) then
      // transition firing update ETL
      Demark_input_places (T, ETL) ;
      Transitionsfired.add(T);
    end if ;
  end while ;
  while elements in Transitionsfired do
    T = next_element(Transitionsfired) ;
      // update AETL
    Mark_output_places (T, AETL);
  end while ;
  Clear(Transitionsfired);

  // update ETL with AETL
  ETL.update(AETL);
  Clear(AETL);
end loop
```

Program 1. ET Coordinator Treatment Loop

In the case of the SRP technique, MRPL contains the marked representing places and the MSPL the marked synchronization places. The output transitions of a marked representing place are verified for enabling If a represented transition fires the verification process ends because the rest of represented transitions become disabled (effective conflict). MRPLnext and MSPLnext are built with the places that become marked in a treatment cycle. Finally, MRPL and MSPL are incremented with MRPLnext and MSPLnext respectively.

```
loop forever
  while elements in MRPL do
    Rplace = next_element (MRPL);
    Transitionsrepr=
          RPlace.transitionsrep ;
    while elements in Transitionsrepr do
     T = next_element(Transitionsrepr) ;
     // enabled transition analysis
     if enabled(T) and predicate(T) then
       // transition firing
       // update MRPL and MSPL
       Demark_input_places(T,MRPL,MSPL);
       Break () ;
     else
  // only in DRP
  Change_representing_place (T) ;
    end if ;
    end while ;
  end while ;
  while elements in Transitionsfired do
    T = next_element(Transitionsfired) ;
   // update MRPLnext and MSPLnext
    Mark_output_places(T,MRPLnext,
                          MSPLnext);
  end while ;
  // update MRPL with MRPLnext
  // update MSPL with MSPLnext
  MRPL.update(MRPLnext);
  MSPL.update(MSPLnext);
  Clear(MRPLnext); Clear(MSPLnext);
end loop ;
```

Program 2. RP Coordinator Treatment Loop

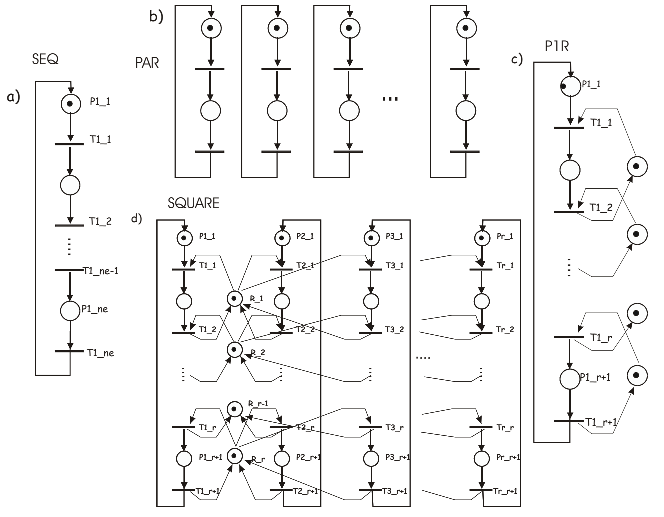

Figure 1. Petri Nets Library

III. The Size Of The Lists

A library of Petri Nets has been developed for carrying out the ability evaluation. Some of these models are well-known and frequently used in literature. The library comprises the following nets:

- *SEQ*. Petri Nets with one sequential process with ne (1..100) states (Figure 1.a).
- *PAR*. Petri Nets with p (1..100) sequential processes with 2 places (Figure 1.b).
- *PR1*. Petri Nets with p (1..40) sequential processes with 2 states and a common resource. These belong to s^3pr net class [18].
- *P1R*. Petri Nets with 1 sequential process and r (1..40) resources (Figure 1.c). These belong to s^3pr net class [18].

- *SQUARE*. Petri Nets with r (5..15) sequential processes of r+1 states and r common resources (Figure 1.b) [7].
- *PR5*. Petri Nets of p (5..62) sequential processes of 6 states and 5 common resources [7].

Table 1 shows some aspects that have a great influence in the average cycle time of an implementation technique: the average size of the treatment and formation lists, and also the average number of tested transitions. Two techniques are considered, the ET and the SRP. In the case of SRP technique, two different sets of representing places are considered the “Bad Election” and the “Good Election”.

The size of the treatment lists depends on the actual marking of the PN. For example, the size of ETL (list of enabled transitions) is 1 in a sequential PN, equal to the number of process in a PAR PN, whilst in a PR1 PN takes the values p and 1, so the average is (p+1)/2.

The size of the formation list depends on the number of fired transitions, and it depends on the conditions associated to transitions. The formation list management affects also the implementation performance. For example, in PAR PN a number from 0 to p transitions can be fired, so the AETL list can have a size that varies from 0 to p.

For ET the average size of the formation and treatment lists in each cycle are presented. In ET the sum of the average size of ETL and AETL lists is the average number of tested transitions in each cycle.

In the case of SRP technique, the average number of marked representing places, the average number of represented transitions and the average number of tested transitions are presented. In this technique, the number of tested transitions can vary from the number of representing places to the number of represented transitions. It depends on which transition is fired in an effective conflict.

The table shows, for example, that the size of ETL list in the ET technique is proportional to the number of processes in PR5, PR1 or SQUARE nets. This means that the execution time corresponding to the enabling test analysis increases proportionally to the number of processes. As can be seen in the presented tests, this kind of implementation presents poor performance for the cited nets.

In SQUARE nets the ETL list is proportional to the number of resources and processes and the size of AETL list can be quadratic. However, in the Representing Places technique the transitions are verified and the size of list in formation increases linearly. This fact means that the performance of ET implementation will make worse when the size of net increase, as the tests show.

IV. PN IMPLEMENTATION ABILITY

A. *Implementation ability*

In [9] defines the ability of a PN implementation as a mean number of enabled transitions per unit of time divided by the mean number of transitions analyzed in order to establish their enabling per time unit. This ratio is a measurement of the number of unnecessary tests carried out in an implementation.

Ability is a metric of the performance of the implementation, but does not take into account all the parameters which influence performance, such as the way to evaluate the enabling of a transition, the influence of the size of the lists on the execution time, or the time necessary to resolve the conflicts. However, this simple measurement can provide guidance on the behavior of a specific technique in the execution of a Petri Net.

As ability is a metric formulated in terms of number of transitions, the ability of the Representing Places techniques will need to be defined to make it comparable to the ability of the other techniques.

4.2 Ability of the Enabled Transitions techniques

If ETL is the set of Enabled Transitions and AETL the set of Almost Enabled Transitions in a cycle, the ability of technique h is:

$$h_{ET} = \frac{|ETL|}{|ETL| + |AETL|} \qquad (1)$$

Unlike the Brute Force technique, in the enabling test only fully enabled transitions are evaluated. Obviously implementation ability is greater than in Brute Force.

4.3 Ability of the Representing Places techniques

In Representing Places techniques (static and dynamic), only the output transitions of the marked representing places are evaluated in the enabling test. However, for example, if a transition fires in binary nets, the rest of the represented transitions are not evaluated. Hence in the enabling test the number of evaluated represented transitions may range from a minimum which would be equal to the number of representing places marked, through to a maximum which would be the totality of transitions with a marked representing place. Implementation ability is therefore improved.The formation and treatment lists must be updated in the transition firing stage in Representing Places techniques. The ability of the technique is a metric in terms of transitions. If ETL is the set of Enabled Transitions, MRPL the set of Marked Representing Places, trs the mean number of Representing Transitions whose enabling has been tested in the cycle, contemplating only the enabling test stage, the ability of the technique is:

$$h_{RP} = \frac{|ETL|}{|MRPL| * t_{rs}} \qquad (2)$$

Petri nets	Enabled Transitions		Representing Places							
			Bad election				Good election			
	ETL	AETL	MRPL	trepres	Tr tested	MRPL next	MR PL	trepr es	Tr tested	MRPLnext
P1R	1	0..1	$\frac{r^2+1}{(r+1)}$	$\frac{r^2+1}{(r+1)}$	$\frac{r^2+1}{(r+1)}$	0.. $\frac{(r+1)}{(r+2)}$	1	1	1	0..1
PR1	$\frac{p+1}{2}$	0.. $\frac{p+1}{2}$	p	p	p	0..1	1	$\frac{p+1}{2}$	1.. $\frac{p+1}{2}$	0..1
PR5	$\frac{p+3}{2}$	0.. $\frac{5p+6}{2}$	3	2.5*p+0.5	3.. 2.5*p+0.5	0..2.5	p	p	p	0..3
SQ. p=r	$\frac{p+\left\lceil \frac{r}{2} \right\rceil}{2}$	0.. $\left\lceil \frac{r}{2} \right\rceil (p+1)-\frac{p}{2}$	$\left\lceil \frac{r}{2} \right\rceil + 0.5$	$\frac{r}{2} * p + 0.5$	$\left\lceil \frac{r}{2} \right\rceil + 0.5$ .. $\frac{r}{2} * p + 0.5$	0.. $\left\lceil \frac{r}{2} \right\rceil + 0.5$	p	p	p	0.. $\left\lceil \frac{r}{2} \right\rceil$
SEQ	1	0..1					1	1	1	0..1
PAR	p	0..p					p	p	p	0..p

Table 1 Average Size of Treatment Lists and Formation Lists. r resources number, p processes number.

However, in techniques such as Static Representing Places and Dynamic Representing Places, places rather than transitions are tested in the list update stage, meaning that ability, as defined, cannot be directly measured in these Representing Places techniques.

The ability of these Representing Places techniques taking into account the lists update stage will be defined.

Definition 1 Ability of the Representing Places techniques. MRPLnext is the set of marked Representing Places of the following cycle, and MSPLnext the set of marked Synchronization Places of the next cycle. The ability of the Static Representing Places and Dynamic Representing Places techniques will be:

$$h_{RP} = \frac{|ETL|}{|MRPL| * t_{rs} + \frac{|MRPLnext| + |MSPLnext|}{f_s}} \quad (3)$$

f_s, the synchronization factor, is the mean number of input places of a transition.

5.4 Static and dynamic abilities

The ability of the technique is not a fixed value, but depends on the time specific behavior of the net and its evolution. For example, in nets interpreted for control, it depends on the sequence of events which reaches the net. In industrial control systems there are many control cycles in which there are no events which reach the control system.

The ability of the techniques is different if events reach or do not reach the net. The following are defined:

Definition 2. Static ability. Mean ability of the technique when transitions are not fired.

Definition 3. Dynamic ability. Mean ability of the technique when transitions are fired. As example, in the representing places techniques:

$$he_{RP} = \frac{|ETL|}{|MRPL| * t_{rs}} \quad (4)$$

$$hd_{RP} = \frac{|ETL|}{|MRPL| * t_{rs} + \frac{|MRPLnext| + |MSPLnext|}{f_s}} \quad (5)$$

Static ability he depends on the current marking of the net. Dynamic ability he depends on the current marking and evolution of the net. Current marking defines the content of the treatment list. The evolution of the net (firing transitions) defines the content of the formation list.

5. STUDY OF THE ABILITY OF THE IMPLEMENTATION TECHNIQUES.

In SEQ nets there is only one marked representing place and one enabled transition. If the only transition possible fires in a cycle, there will be a marked representing place of the next cycle and an Almost Enabled transition. Moreover, as fdes=1, then trs=1. The ability of the Representing Places technique is as follows:

$$h_{RP} = \frac{1}{1+t_{rf}} = h_{ET}; t_{rf} \in N, t_{rf} \le 1 \qquad (6)$$

In which trf is the number of transitions fired in the cycle.

In PAR nets the number of marked representing places is equal to the number of enabled transitions and the number of marked representing places of the next cycle is equal to the number of Almost Enabled transitions. In PAR nets there are no synchronization places. Moreover, as fdes=1, then trs=1.

In PAR nets The number of Almost Enabled transitions is equal to the number of fired transitions, and hence shall have a minimum of zero transitions fired and a maximum of p transitions fired, in which p is the number of Petri Net processes.

$$h_{RP} = \frac{p}{p+t_{rf}} = h_{ET}; t_{rf} \in N, t_{rf} \le p \qquad (7)$$

In SQUARE nets. The mean number of enabled transitions is:

$$|ETL| = \frac{p + \left\lceil \frac{r}{2} \right\rceil}{2} \qquad (8)$$

The mean number of almost enabled transitions is:

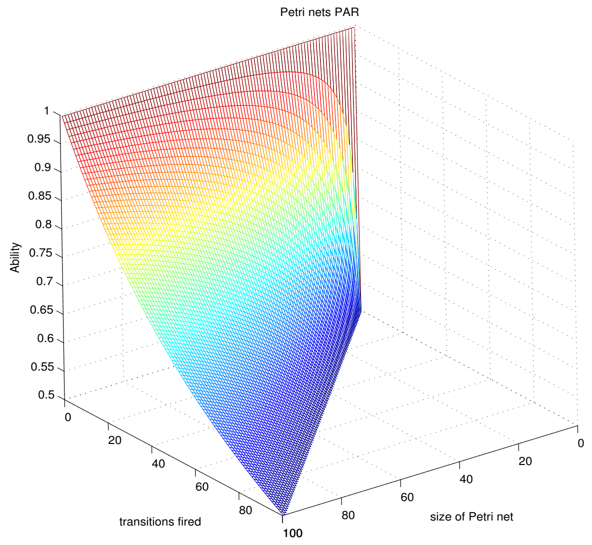

Figure 2. Ability of the Enabled Transitions and Representing Places techniques in PAR nets.

$$|AETL| = \left(\left\lceil \frac{r}{2} \right\rceil (p+1) - \frac{p}{2} \right) * \frac{t_{rf}}{\left\lceil \frac{r}{2} \right\rceil} \qquad (9)$$

And the ability of Enabled Transitions in Square nets is:

$$h_{ET} = \frac{\frac{p + \left\lceil \frac{r}{2} \right\rceil}{2}}{\frac{p + \left\lceil \frac{r}{2} \right\rceil}{2} + \left(\left\lceil \frac{r}{2} \right\rceil (p+1) - \frac{p}{2} \right) * \frac{t_{rf}}{\left\lceil \frac{r}{2} \right\rceil}}; t_{rf} \in N, t_{rf} \le \left\lceil \frac{r}{2} \right\rceil \qquad (10)$$

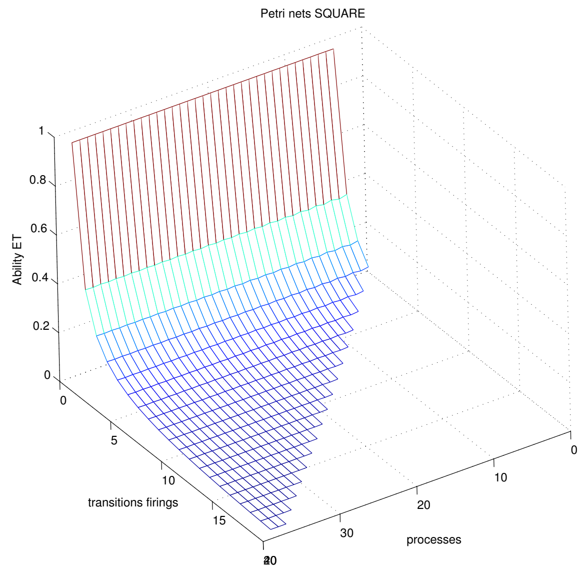

Figure 3. Ability of the Enabled Transitions techniques in SQUARE nets.

If those belonging to the sequential process are chosen as Representing Places p marked representing place shall exist. If the transition fires, fp places are marked, and so

$$|MRPLnext| + |MSPLnext| = f_p * t_{rf} \qquad (11)$$

$$h_{RP} = \frac{\frac{p + \left\lceil \frac{r}{2} \right\rceil}{2}}{p + t_{rf}}; t_{rf} \in N, t_{rf} \le \left\lceil \frac{r}{2} \right\rceil \qquad (12)$$

As observed in Figure 3, the ability of the Enabled Transitions technique in SQUARE nets starts from 1.0 (without transitions firing) but quickly decreases in line with the increase of the number of transitions fired towards values of less than 0.2. As observed in Figure 3 and Figure 4, when transitions do not fire, Representing Places ability is less than Enabled Transitions ability. On the other hand, the ability of the Representing Places technique does not drop as quickly as Enabled Transitions ability when the number of transitions fired increases.

If the resources are chosen as Representing Places and also the last place of each process (necessary to represent the last transition of the process), there shall be a mean of $\left\lceil \frac{r}{2} \right\rceil + 0.5$ marked Representing Places.

If the first transition possible within those represented fires, for example the first transition represented by resource R_1 (transition T1_1) (see Figure 1.d), when the Static Representing Places algorithm tests the enabling of the transitions represented by R_3, transition T1_3 cannot fire, the first transition of those represented which could fire would be T2_3. In each algorithm cycle, in each sequential process of the SQUARE net, only one transition can fire. The Ability if the first transition possible fires is:

$$h_{RP} = \frac{\dfrac{p + \left\lceil \frac{r}{2} \right\rceil}{2}}{\left\lceil \frac{r}{2} \right\rceil \dfrac{1+\left\lceil \frac{r}{2} \right\rceil}{4} + \left\lfloor \frac{r}{2} \right\rfloor \dfrac{1+\left\lfloor \frac{r}{2} \right\rfloor}{4} + 0.5 + t_{rf}} ; t_{rf} \in N, t_{rf} \le \left\lceil \frac{r}{2} \right\rceil \quad (13)$$

If the last transition possible fires, ability will be:

$$h_{RP} = \frac{\dfrac{p + \left\lceil \frac{r}{2} \right\rceil}{2}}{\frac{r}{2} p + 0.5 - \dfrac{\left(\left\lceil \frac{r}{2} \right\rceil - 1\right)\left\lceil \frac{r}{2} \right\rceil}{4} - \dfrac{\left(\left\lfloor \frac{r}{2} \right\rfloor - 1\right)\left\lfloor \frac{r}{2} \right\rfloor}{4} + t_{rf}} ; t_{rf} \in N, t_{rf} \le \left\lceil \frac{r}{2} \right\rceil \quad (14)$$

In Figure 4 the graphs of the ability of the Representing Places technique in SQUARE nets is observed. It is observed that if the process places are chosen as representing places, ability is much greater than when resources are chosen. Moreover, in the latter case ability also depends on which transition of those represented fires.

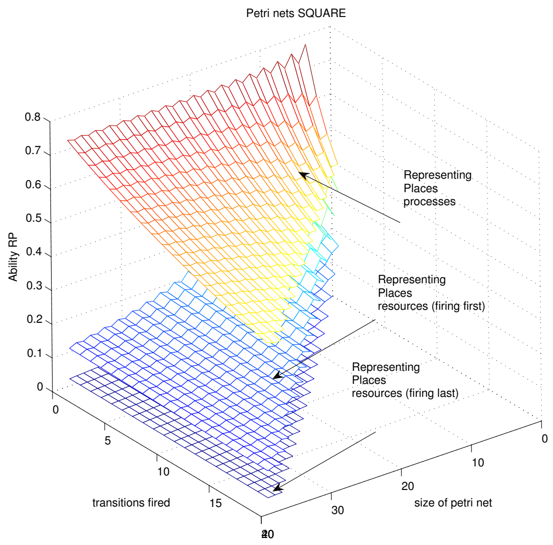

Figure 4. Ability of the Representing Places techniques in SQUARE nets.

The ability of the Representing Places technique with places representing those belonging to processes does not depend on the size of the net, and drops if the number of transitions fired increases. However, in any situation, the ability of the technique is better than when resources are chosen as representing places.

V. ABILITY AND EXECUTION TIME

The ability charts shown correspond to the results of the experiments carried out in which execution time has been measured. For example:

- In SQUARE nets, if processes are chosen as representing places, the Representing Places algorithm has a lower execution time than Enabled Transitions (see Figure 5), which is in keeping with the ability charts (see Figure 3 and Figure 4) in which it is observed that the ability of the Enabled Transitions technique drops quickly when the number of transitions fired increases.

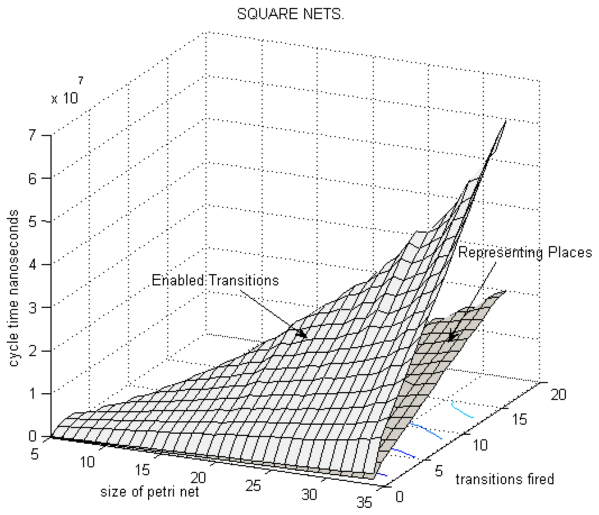

Figure 5 Execution Time in Square nets

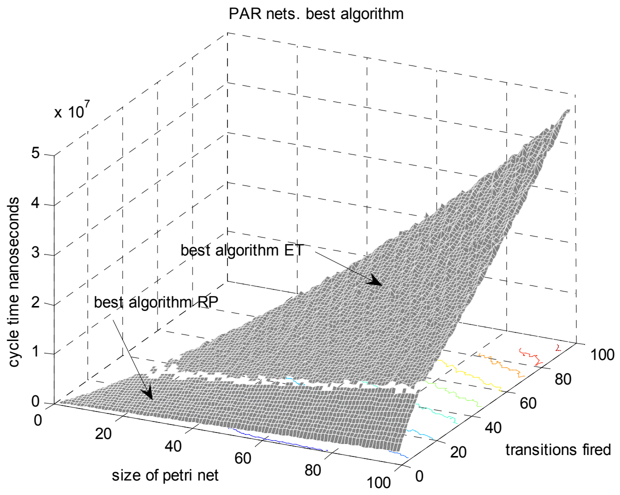

Figure 6 Execution Time in PAR nets

- Similarly, in PAR nets ability is identical in both Enabled Transitions and Representing Places techniques (see Figure 2). In this case execution time can be observed in Figure 6, showing that if few transitions fire, the best algorithm is Representing Places, whilst in other cases it is Enabled Transitions. This is due to the Representing Places technique handling four lists and the handling time of these four lists is greater.

The ability of the technique is a measurement which is independent of the implementation platform, providing guidance with regards to execution time results, which are, obviously, dependent on this platform. This independence of the implementation platform has been supplemented with the comparison to the execution time tests, which ratify the ability analysis.

VI. Conclusions

In order to supplement the execution time tests and further analyze the behavior of the techniques, a study of the size of the formation and treatment lists of the algorithms has been carried out. This study clearly indicates the number of transitions which are examined per cycle in each algorithm, therefore allowing an initial estimate of which technique presents best performance in accordance with the type of Petri net run. This list size study will allow the completion of the study of the ability of the different techniques. Ability is a metric of implementation performance, but does not take into account all the parameters which influence this performance. However, this simple measurement can provide guidance on the behavior of a specific technique in the execution of a Petri Net. A quantitative study of the ability of the different techniques has been completed. As ability is a metric formulated in terms of number of transitions, the ability of the Representing Places techniques has been defined to make it comparable to the ability of the other techniques. The terms static ability and dynamic ability have also been defined and studied. It has been shown that a bad choice of Representing Places implies worse ability of the technique. The ability of the technique is independent of the implementation platform. This independence of the implementation platform has been supplemented with the comparison to the execution time tests, which ratify the ability analysis.

Acknowledgments

We wish to thank Manuel Silva and Cristian Mahulea, for their help.

References

[1] T. Murata, "Petri Nets- Properties, Analysys and applications," *Proceedings of the Ieee,* vol. 77, pp. 541-580, Apr 1989.

[2] G. V. Brams, "Reseaux de Petri: Theorie et Practique, Vols. I and II," Masson, 1982.

[3] J. M. Colom, M. Silva, and J. L. Villarroel, "On software implementation of Petri nets and colored Petri nets using high-level concurrent languages," *Seventh European Workshop on applications and theory of Petri nets, Oxford, July,* vol. 86, pp. 207-241, 1986 a.

[4] J. L. Briz and J. M. Colom, "Implementation of Weighted Place/Transition Nets based on Linear Enabling Functions," *Application and Theory of Petri Nets,* vol. 815, pp. 99–118, 1994.

[5] D. Taubner, "On the implementation of Petri nets," *Lecture Notes in Computer Science,* vol. 340, pp. 418-439, 1988.

[6] G. Bruno, A. Castella, G. Macario, and M. Pescarmona, "Scheduling hard real time systems using high-level petri nets," *Application and Theory of Petri Nets,* vol. 616, pp. 93–112, 1992.

[7] R. Piedrafita and J. L. Villarroel, "Performance evaluation of petri nets execution algorithms," in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, 2007, pp. 1400-1407.

[8] R. Piedrafita and J. L. Villarroel, "Adaptive Petri Nets Implementation. The Execution Time Controller.," in *9th International Workshop on Discrete Event Systems 2008*, Gothenburg, Sweden, 2008, pp. 1400-1407.

[9] J. L. Villarroel, "Integración Informática del Control de Sistemas Flexibles de Fabricación," in *Ingeniería Eléctrica e Informática* Zaragoza: Universidad de Zaragoza, 1990.

[10] R. Piedrafita, "Técnicas de Implementación de Sistemas de Control de Eventos Discretos," in *Informática e Ingeniería de Sistemas* Zaragoza: Universidad de Zaragoza, 2009.

[11] M. Silva, *Las Redes de Petri en la Automatica y la Informatica.*, AC ed. Madrid: AC, 1985.

[12] F. J. García and J. L. Villarroel, "Modelling and Ada Implementation of Real-Time Systems using Time Petri Nets," *Proc. of the 21st IFAC/IFIP Workshop on Real-Time Programming. Gramado-RS, Brazil. November,* 1996.

[13] J. L. Briz, "Técnicas de implementación de redes de Petri. ," *PhD thesis, Univ. Zaragoza,* 1995.

[14] R. Valette, M. Courvoisier, J. M. Bigou, and J. Albukerque, "A Petri Net Based Programmable Logic Controller," *Computer Applications in Production and Engineering,* vol. 11, pp. 103-115, 1983.

[15] R. Valette and B. Bako, "Software implementation of Petri nets and compilation of rule-based systems," *Lecture Notes in Computer Science,* vol. 524, pp. 296-316, 1991.

[16] D. Chocron, "Un Systéme de Programmation par RdP de Controleurs Industriels," Montreal: École Politecnique de Montreal, 1980.

[17] M. Silva and S. Velilla, "Programmable logic controllers and Petri nets: A comparative study," in *IFAC/IFIP Symposium on Software for Computer Control* Madrid, Spain, 1982, pp. 83–88.

[18] J. Ezpeleta, J. M. Colom, and J. Martinez, "A Petri-net based deadlock prevention policy for flexible manufacturing systems," *Ieee Transactions on Robotics and Automation,* vol. 11, pp. 173-184, Apr 1995.