VERIFICATION OF REAL TIME UML SPECIFICATIONS THROUGH A SPECIALIZED INFERENCE MECHANISM BASED ON A TOKEN PLAYER ALGORITHM AND THE SEQUENT CALCULUS OF LINEAR LOGIC

Stéphane Julia, Michel dos Santos Soares
Faculdade de Computação
Universidade Federal de Uberlândia,
P.O. Box 593, 38400-902, Uberlândia-M.G-Brazil,
email: stephjl@aol.com, michelssoares@yahoo.com.br

KEYWORDS—**UML, Petri Net, Linear Logic, Scheduling, Real Time System, Batch System**

Abstract

The objective of this article is to present an approach based on UML dynamic diagrams, on time Petri Nets and Linear Logic for scenario verification of Real Time Systems. The main idea consists of translating the sequence diagrams which express the initial specifications of the system to a unique p-time Petri Net model which represents the global behaviour of the entire system. For the Petri Net fragments involved in conflict situations, symbolic production and consumption dates assigned to tokens are calculated using a non-conventional (max;+) algebra based on the sequent calculus of Linear Logic. These dates are then used to solve conflict situations within a token player algorithm used for scenario verification of Real Time specifications and which can be seen as a simulation tool for UML interaction diagrams. The approach is illustrated through an example of Real Time System used at the global coordination level of a Batch System.

I. INTRODUCTION

The Object Oriented methods seem to be very suitable for proposing approaches to represent Real Time Systems, as real objects of the physical system are naturally transformed into software objects that can be easily implemented and verified. Of all Object Oriented notations, UML [OMG 1999] is one of the best accepted in the software industry. In particular, with the dynamic diagrams proposed by UML, it is possible to represent the communication mechanisms among several objects for a specific scenario. Therefore, UML notations have their limitations when they are used for specifying Real Time Systems. For example, it is not possible with a unique UML diagram to represent the set of all dynamic interactions that exist at a global system level. As a consequence, it becomes very difficult to guarantee that the execution of several sequence diagrams in parallel will not lead to time constraint violations.

Petri Nets [Murata 1989] are very well adapted to model Real Time Systems, as they allow for a good representation of conflict situations, shared resources, synchronous and asynchronous communication, precedence constraints and explicit time constraints in the time Petri Net case.

As was presented in [Cardoso 2001], translating sequence diagrams of UML in Petri Net models allows one to define an operational semantic for the sequence diagrams in order to know how these diagrams are executed in real time.

The dynamic behaviour of a system imposes a scheduling of control flow. The scheduling problem consists of organizing in time, the sequence of the operations considering time constraints (time intervals) and constraints of shared resources utilization necessary for operation execution. From the traditional point of view of Software Engineering, the scheduling problem is similar to the activity of scenario execution. A scenario execution becomes a kind of simulation which shows the system's behaviour in real time. In the real time system case, several scenarios can be executed simultaneously and conflict situations which have to be solved in real time (without a backtrack mechanism) can occur if a same non-preemptive resource is called at the same time for the execution of operations which belong to different scenarios.

In [Julia 2002], a simulation technic based on a token player algorithm was presented whose purpose was to verify real time UML specifications. The basic principle was to generate a class graph [Khansa 1996] each time a conflict situation for a shared resource was met in order to guarantee that a time constraint violation would not be reached. The class graph allows one to represent all possible evolutions of p-time Petri net fragments involved in conflict situations but has the disadvantage of the combinatory explosion. Another problem is that for each new scenario execution, new class graphs which are based on real numerical dates and durations must be calculated.

In this paper, an approach, based on the sequent calculus of Linear Logic [Girard 1987], will be proposed to solve conflict situations in order to accelerate conflict resolutions during the scenario execution. This will be realized calculating symbolic dates for the tokens in conflict by using a non conventional (max;+) algebra [Rivière 2001].

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)
(c) SCS European Council / SCS Europe BVBA, 2003
ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

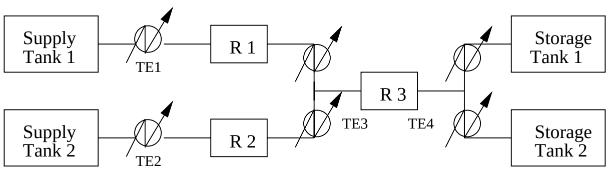

Fig. 1. Batch Production System

II. REAL TIME SYSTEMS MODELLING BASED ON UML DIAGRAMS AND ON P-TIME PETRI NET

A Real Time System used at the global coordination level of the Batch System given in figure 1 will be considered to illustrate the specification activity of Real Time Systems using UML diagrams. A batch is a quantity of material which is transformed passing through different equipment and respecting a specific recipe which defines the sequence of operations. The production system of figure 1 executes two different recipes. Initially, two batches (1 and 2) are stored in their respective supply tanks (1 and 2). Recipe 1 consists on transferring batch 1 from supply tank 1 to reactor R1 to a processing stage utilizing thermal exchange TE1. After batch 1 is processed in R1, it is transferred to reactor R3 to another stage in processing, passing through TE3. When finalized, batch 1 is deposited in storage tank 1, passing through TE4 for the final product liberation. Recipe 2 behaves in a similar manner and consists of transferring batch 2 from supply tank 2 to storage tank 2, passing through reactors R2 and R3 and using thermal exchanges TE2, TE3 and TE4. The advantage of considering such a system is that some of the main features which generally appear in Real Time Systems will be considered. The execution of different recipes can be seen as a typical example of parallelism, and the utilization of common equipment (for example, reactor R3 will be used by both recipes) is an example of resource sharing.

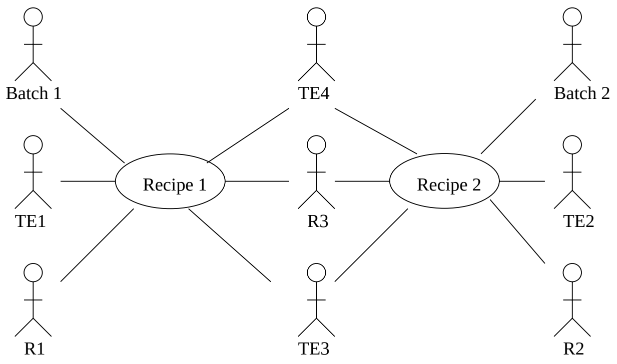

Fig. 2. Use Case Diagram

The proposed approach uses a Use Case diagram to show the main functions of the Real Time System from a user's point of view and the relationship between the system and the environment. Analysing Real Time Systems from an Object Oriented approach, the actors of the Use Case diagram are generally good candidates for objects. From the Use Case diagram of figure 2, it is possible to note that the execution of recipe 1 needs the physical equipment: TE1, TE3, TE4, R1 and R3. In the same manner, the execution of recipe 2 needs the physical equipment: TE2, TE3, TE4, R2 and R3. As actors are good candidates for objects, it is easy to conclude that there exists a recipe class which has a method that corresponds to the treatment of the corresponding batch, a reactor class which has a method that corresponds to the processing of a batch into a physical reactor, and a thermal exchange class which has a method called for transport operations. As a matter of fact, each actor of the Use Case diagram will have to communicate with its corresponding software object.

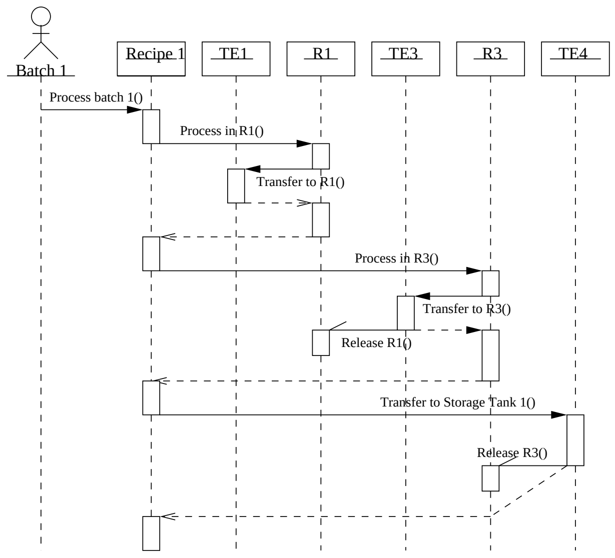

Fig. 3. Sequence Diagram for Recipe 1

After defining the main objects of the system, each function corresponding to a Use Case is given through a specific sequence of operations modelled by a sequence diagram that shows the communication mechanisms among the involved objects. This type of diagram explicitly shows the chronological order of operations. The scenario execution of recipe 1 and the interactions between the involved objects are shown in the sequence diagram of figure 3. Looking at this diagram, it is quite clear that initially, the actor Batch 1 calls the method of the software object which will execute recipe 1. After that, the object Recipe 1 calls the method of R1 for the processing operation. Then, object R1 calls the method of TE1 to transfer batch 1 from supply tank 1 to reactor R1 and, at the end of the first processing stage, a message is sent to the object Recipe 1 which can call the method of the object R3 to the second processing stage. The object R3 calls the method of the object TE3 to transfer batch 1 from reactor R1 to reactor R3. Once batch 1 is transfered to R3, an asynchronous message is sent to R1 so that it becomes available and a synchronous response is sent to object R3 so that the processing in reactor R3 begins. At the end of the processing in reactor R3, the thermal exchange TE4 is requested to transfer batch 1 to storage tank 1 for product liberation. At the end of the transfer operation, an asynchronous message is sent to R3 so that it becomes available for other operations. The sequence diagram of recipe 2 is similar to the one of recipe 1. To build it, it is necessary to change the actor Batch 1 and the objects Recipe 1, TE1 and R1 from the sequence diagram shown in figure 3, by the actor Batch 2 and the objects Recipe 2, TE2 and R2, respectively.

One way of analysing specifications given through semi-formal UML notation is to formally define an operational semantic for the UML dynamic diagrams based on a formal notation which shows how these diagrams are executed in real time and which allows one to analyse qualitativaly and quantitativaly real time UML specifications. In particular, by analysing sequence diagrams separately, it is not possible to verify if a conflict situation can occur. For example, during real time execution, both sequence diagrams may have to request some common objects at the same time interval. Another limitation of the sequence diagrams in the real time system case is that explicit time constraints, like the initial date of a scenario execution, do not appear formally on these diagrams.

Petri Nets allows one to describe internal behaviour of objects as well as synchronous and asynchronous communications between objects (synchronous and asynchronous communications are represented by communication places (semaphore type)). Based on the interactions between objects specified in a sequence diagram, it is possible to obtain a Petri Net model which shows the interactions between the Petri Net objects (for each object, there is a corresponding Petri Net template which shows the internal behaviour of the object) involved in the execution of the scenario. Applying some of the reduction rules [Murata 1989] of the Petri Net theory which allow one to eliminate some of the communication places and of the waiting places, a reduced Petri Net model can be obtained where each object of the sequence diagram is represented by a non-preemptive resource. Some of these resources may be used by different scenarios (objects which belong to several sequence diagrams). As a consequence, the merging of these shared places produces a unique global Petri Net model which represents the global behaviour of the entire system. As was shown in [Julia 2000], explicit time constraints which exist in a Real Time System, can be formally defined using a p-time Petri Net model. The static definition of a p-time Petri Net is based on static intervals which represent the permanency duration (sojourn time) of tokens in places and the dynamic evolution of a p-time Petri Net model depends on the time situation of the tokens (date interval associated with the tokens). Figure 4 shows the p-time Petri Net model

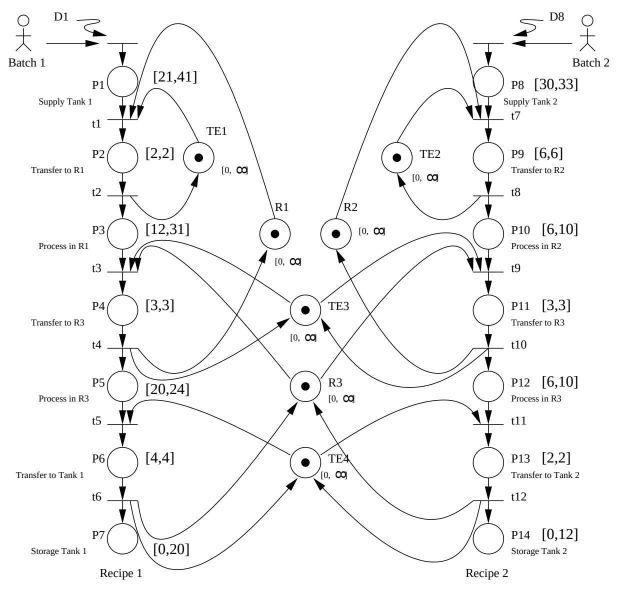

Fig. 4. p-time Petri net model for Recipe 1 and Recipe 2

which corresponds to the whole system (recipe 1 + recipe 2). Each recipe is represented by a production route (a sequence of transitions and places) and each object of the sequence diagrams is represented by a resource (a shared resource if the object appears in both recipes, like R3 for example). For example, as specified on the sequence diagram in figure 3, after the processing of batch 1 in R1, the resource R1 becomes available only when batch 1 is transfered from reactor R1 to reactor R3.

The static intervals associated to each place of the model in figure 4 and the execution beginning of each recipe, specified by the firing date of the first transition of each recipe (D1 for recipe 1 and D8 for recipe 2), will depend on specific production plans.

III. CONFLICT SITUATION ANALYSIS IN A P-TIME PETRI NET

In p-time Petri Nets, as shown in [Julia 2000], conflict situations for shared resources are visible during a time interval (there exists the notion of Conflict Time Interval), and not at a single time point. For example, considering the p-time Petri Net of figure 5 (the Petri net fragment involved in the conflict for the shared resource R3), if a token appears in P3 at date D3=23 and a token appears in P10 at date D10=36, then, the visibility intervals of these tokens are $[(\delta_{p3})_{min};(\delta_{p3})_{max}] = [12+23;31+23] = [35;54]$ for the token in P3 and $[(\delta_{p10})_{min};(\delta_{p10})_{max}] = [6+36;10+36] = [42;46]$ for the token in P10. The conflict time interval associated to the pair $(t_3; t_9)$ is given by the intersection of these visibility intervals: $[35; 54] \cap [42; 46] = [42; 46]$. So, an effective conflict between t_3 and t_9 is able to occur during the interval [42 ; 46].

Using a class graph algorithm, it is possible to verify if a special class, called "death token class", which represents a time constraint violation (the resource necessary in order to respect a time interval associated to a place of a p-time Petri Net is not available at the right time), can be reached when considering a Petri Net fragment involved in a specific conflict state. But the class graph has some limitations; the state space can be very large and the duration a token can remain in the same place has to be delimited by real numerical values and not symbolic ones. In particular, each time a new simulation is executed, new conflict time intervals are calculated and, for each conflict situation, a class graph has to be generated. As a consequence, the global simulation speed is reduced.

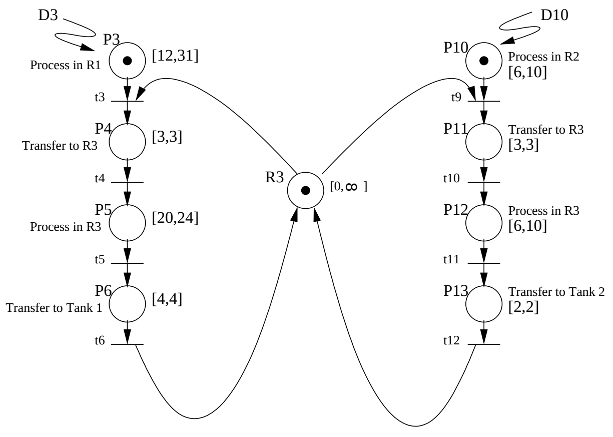

Fig. 5. Useful part of the conflict for R3

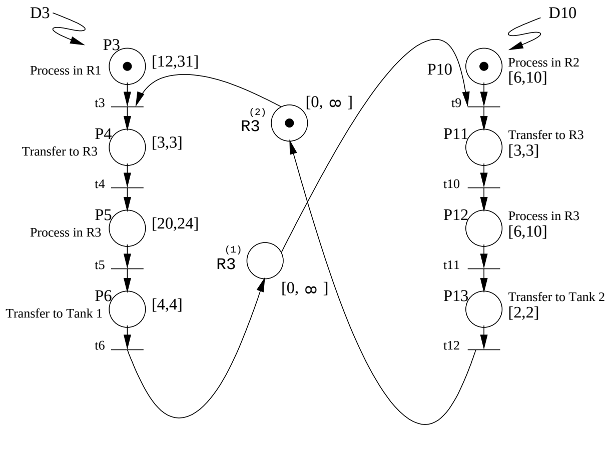

Fig. 6. Conflict Resolution for R3

A conflict situation for a shared resource in a p-time Petri Net is equivalent to a certain extent to a Watch Dog [Rivière 2001] (two transitions in structural conflict) represented by a t-time Petri Net (time

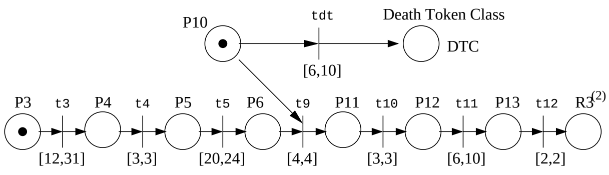

Fig. 7. Watch dog

intervals associated with transitions) where a specific place corresponds to the "death token class". Watch Dogs are commonly used to analyse behaviours which deviate from their normal evolution. As there exists an equivalence between a p-time Petri Net and a t-time Petri Net in case of the earliest firing strategy [Khansa 1996], it will be possible to transform a conflict state given by a p-time Petri Net model into a t-time Petri Net model corresponding to a typical case of Watch Dog where one of the transitions in structural conflict represents the normal evolution of the system and the other one signals a time constraint violation. For example, considering the conflict situation for the shared resource R3 shown in figure 5, if the reactor R3 is used to treat batch 1 and, after that, batch 2, then, the Petri Net in figure 5 can be transformed into the Petri Net of figure 6. The problem is then to verify that making this decision, the reactor R3 will be available in place $R3^{(1)}$ before the maximum bound of the visibility interval associated to the token in place P10, which represents the latest date authorized for the firing of transition t_9. On the contrary, a death token class will be reached. The Watch Dog modelled by the t-time Petri Net in figure 7 represents such a situation. In particular, this Watch Dog specifies that, at date D10 + 10 (D10 is the date when the token appears in P10 and 10 is the maximum bound of the time interval associated to transition t_{dt}), if the transition t_9 has not been fired yet (which means that the resource in $R3^{(1)}$ is not available at this moment for the firing of t_9 in figure 6), then the transition t_{dt} has to be fired and a death token class, which corresponds to a time constraint violation, will be reached. The place $R3^{(1)}$ does not appear in the Watch Dog because its static interval is $[0; \infty[$ and, as a consequence, the effect of this place on the temporal reasoning is useless.

In [Rivière 2001], it was shown that applying the sequent calculus of Linear Logic to a Watch Dog represented by a t-time Petri Net and using a non conventional (max;+) algebra, production and consumption symbolic dates assigned to each atom (token) involved in the studied scenario can be calculated. In particular, comparing the symbolic dates of the tokens which belong to the places in structural conflict, it is possible to know if the Watch Dog effectively invalidates the normal operations, i.e. if a "death token class" can be reached.

With Linear Logic [Girard 1987], a marking M is a monomial in $\otimes$, that is a marking represented by $M = A_1 \otimes A_2 \otimes \cdots \otimes A_k$ where A_i are place names. For instance, the initial marking on the Petri Net in figure 7 is $P3 \otimes P10$. A transition is an expression of the form $M_1 \multimap M_2$ where M_1 and M_2 are markings. For example, transition t_3 on the Petri Net in figure 7 is noted $t_3 = P_3 \multimap P_4$. A sequent $M, t_i \vdash M'$ represents a scenario where M and M' are respectively the initial and final markings, and t_i is a list of non-ordered transitions. A sequent can be proved by applying the rules of the sequent calculus as there exists an equivalence between Petri Nets reachability and the proof of sequents in Linear Logic [Girault 1997]. A Linear Logic proof tree is read from the bottom, up and a proof stops when all the leaves of the tree are identity sequents ($P_1 \vdash P_1$, for example).

In a proof tree, each transition firing generates a symbolic date associated to each atom (token) as shown in [Rivière 2001]. In this article, D_i will denote a date and d_i a duration associated to a transition firing. A pair (D_p, D_c) will be associated to each atom of the proof tree; they respectively represent the production and the consumption date of atoms.

From the watch dog of figure 7, two scenarios can be derivated :

$$\begin{aligned} Sc1 &= P3 \otimes P10, t_3, t_4, t_5, t_{dt} \vdash P6 \otimes DTC \\ Sc2 &= P3 \otimes P10, t_3, t_4, t_5, t_9, t_{10}, t_{11}, t_{12} \vdash R3^{(2)} \end{aligned}$$

These scenarios are in conflict i.e. one scenario invalidates the other. As a matter of fact, Sc2 represents the normal evolution of the system when Sc1 represents a time constraint violation (a "death token class"). Table 1 shows the production and consumption dates in the scenario Sc1 case. Table 2 and 3 show the production and consumption dates in the scenario Sc2 case.

In a t-time Petri Net model, any enabling duration d_i takes its values within a time interval $[\delta_{i,min}; \delta_{i,max}]$. For example, when considering the scenario Sc1, the domain for the consumption date of atom P4 (the token in P4) is given by the time interval $[D_3 + d_{3,min} + d_{4,min}; D_3 + d_{3,max} + d_{4,max}]$. Based on a property shown in [Rivière 2001], it is possible to say that Sc1 invalidates Sc2 (which means that the death token class will be reached) if the maximum value of the enabling duration of t_{dt} in scenario Sc1 is smaller than the minimal sojourn time of atom P6 in scenario Sc2. For example, when considering the Petri Net in figure 5, if a token appears in P3 at date D3=23, its visibility interval is [35 ; 54], and, if a token appears in P10 at date D10=36, its visibility interval is [42 ; 46]. Then, it seems natural to fire transition t3 at date 35 when the token in P3 becomes available. From the symbolic production and consumption dates obtained when scenarios Sc1 and Sc2 are considered, the following results are obtained : the minimal production date of atom P6, which is equal to $D_3 + d_{3,min} + d_{4,min} + d_{5,min} =$ $23 + 12 + 3 + 20 = 58$, is bigger than the maximal consumption date of atom P10, which is equal to $D_{10} + d_{dt,max} = 36 + 10 = 46$. As a direct consequence of this result, the firing of t_3 as soon as the token in P3 becomes available will not be allowed by the conflict resolution mechanism which will be used at the global simulation level.

Transition	Consumption Date	Production Date
$t_3 = P3 \multimap P4$	$P3(D3+d3)$	$P4(D3+d3)$
$t_4 = P4 \multimap P5$	$P4(D3+d3+d4)$	$P5(D3+d3+d4)$
$t_5 = P5 \multimap P6$	$P5(D3+d3+d4+d5)$	$P6(D3+d3+d4+d5)$
$t_{dt} = P10 \multimap DTC$	$P10(D10+ddt)$	$DTC(D10+ddt)$

Table 1. Consumption and Production dates for Scenario 1

Transition	Consumption Date
$t_3 = P3 \multimap P4$	$P3(D3+d3)$
$t_4 = P4 \multimap P5$	$P4(D3+d3+d4)$
$t_5 = P5 \multimap P6$	$P5(D3+d3+d4+d5)$
$t_9 = P10 \otimes P6 \multimap P11$	$P6(max(D3+d3+d4+d5, D10)+d9)$ $P10(max(D3+d3+d4+d5, D10)+d9)$
$t_{10} = P11 \multimap P12$	$P11(max(D3+d3+d4+d5, D10)+d9+d10)$
$t_{11} = P12 \multimap P13$	$P12(max(D3+d3+d4+d5, D10)+d9+d10+d11)$
$t_{12} = P13 \multimap R3^{(2)}$	$P13$ $(max(D3+d3+d4+d5, D10)+d9+d10+d11+d12)$

Table 2. Consumption dates for Scenario 2

IV. Simulation principle

One of the approaches which allows one to execute dynamically a Petri Net is the one based on a token player algorithm. A token player algorithm is a special inference mechanism which allows the firing of the enabled transitions. When the model is based on a p-time Petri Net, the token player algorithm must take into account the conflict situations in real time in order to avoid the possibility of deadlock which can be caused by a "death token class". The basic principle of such an algorithm was presented in [Julia 2000]. Figure 8 shows how this algorithm works. The basic difference between our proposed token player and other token player algorithms used in simulation tools based on Petri Nets

Transition	Production Date
$t_3 = P3 \multimap P4$	$P4(D3+d3)$
$t_4 = P4 \multimap P5$	$P5(D3+d3+d4)$
$t_5 = P5 \multimap P6$	$P6(D3+d3+d4+d5)$
$t_9 = P10 \otimes P6 \multimap P11$	$P11(max(D3+d3+d4+d5, D10)+d9)$
$t_{10} = P11 \multimap P12$	$P12(max(D3+d3+d4+d5, D10)+d9+d10)$
$t_{11} = P12 \multimap P13$	$P13(max(D3+d3+d4+d5, D10)+d9+d10+d11)$
$t_{12} = P13 \multimap R3^{(2)}$	$R3^{(2)}$ $(max(D3+d3+d4+d5, D10)+d9+d10+d11+d12)$

Table 3. Production dates for Scenario 2

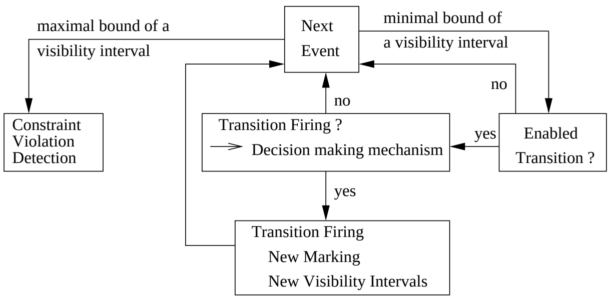
maximal bound of a visibility interval
Next Event
minimal bound of a visibility interval
no
no
Constraint Violation Detection
Transition Firing ?
Decision making mechanism
yes
Enabled Transition ?
yes
Transition Firing
New Marking
New Visibility Intervals

Fig. 8. Token Player Algorithm for a p-time Petri net

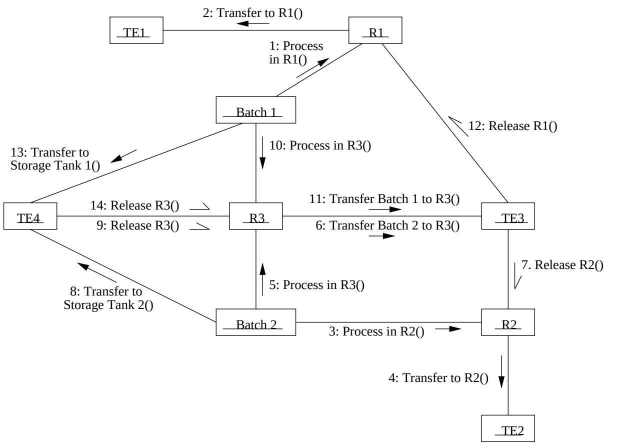

Fig. 9. Collaboration Diagram

models is that transitions are not fired necessarily as soon as they become enabled because of the conflict resolution mechanism presented in the previous section. For example, if the input transitions of P1 and P8 in figure 4 are fired at date D1=D8=0 (beginning of recipes execution), the result of the global simulation based on the token player algorithm is the following one : date 21 = t1 fired; date 23 = t2 fired; date 30 = t7 fired; date 35 = t3 enabled, but not fired; date 36 = t8 fired; date 42 = t9 fired; date 45 = t10 fired; date 51 = t11 fired; date 53 = t12 fired; date 53 = t3 fired; date 56 = t4 fired; date 76 = t5 fired; date 80 = t6 fired. As a result of the token player execution, an acceptable scenario corresponding to a specific firing sequence is obtained and can be translated into the collaboration diagram of figure 9 which represents the global behaviour of the Real Time System from the UML point of view. In this diagram, it is clear that batch 2 has to be processed in reactor R3 before batch 1 in order to respect the time constraints.

V. **CONCLUSION**

The principal advantage shown in this article is the possibility of accelerating the conflict resolution in a p-time Petri Net. This is realized by using the sequent calculus of Linear Logic and a non conventional (max;+) algebra which allows the calculation of symbolic dates, instead of real numerical values. As a direct consequence, the generation of a class graph, each time a conflict situation is met, is not necessary anymore and the global duration of simulation when considering Real Time UML specifications is reduced.

References

[Cardoso 2001] Cardoso, J., Sibertin-Blanc. (2001). Ordering actions in Sequence Diagrams of UML. *23 International Conference on Information Technology Interfaces.* Croatia.

[Girard 1987] Girard, Jean-Yves. 1987. Linear Logic. *Theoretical Computer Science.* 50:1-102.

[Girault 1997] Girault, F. 1997. *A logic for Petri nets, JESA Vol. 31, n.3, Eddition Hermes*

[Julia 2000] Julia, S., Valette, R. (2000). Real Time Scheduling of Batch Systems. *Simulation Practice and Theory, Elsevier Science.* pp. 307-319.

[Julia 2002] Julia, S., Kanacilo, E., M. (2002). An approach based on dynamic UML diagrams and on a token player algorithm for the scenario verification of real time systems. *14TH European Simulation Symposium, Simulation in Industry.* Dresden, German. p. 377-381.

[Khansa 1996] Khansa, W., Aygaline, P., Denat, J. P. (1996). Structural analysis of p-time Petri Nets. *Symposium on discrete events and manufacturing systems. CESA'96 IMACS Multiconference.* Lille, France.

[Murata 1989] Murata, T. (1989). Petri Nets: Properties, analysis and applications. *Proceedings of the IEEE* 77(4). pp. 541-580.

[OMG 1999] OMG Unified Modeling Language Specification version 1.3. *Object Management Group.*

[Rivière 2001] Rivière, N., Pradin-Chezalviel, B., Valette, R. (2001). Reachability and temporal conflicts in t-time Petri Nets. *9TH International Workshop on Petri Nets and Performance Models (PNPM'01).* Aachen-Germany. pp. 229-238.

STÉPHANE JULIA received his Ph. D. degree from the University "Paul Sabatier" of Toulouse (France) in 1997. He is currently a Professor at the Federal University of Uberlândia (Brazil) in the Computer Science Department. His current research interests include the application of Petri net theory in Real Time Systems. He is also interested in the relationships between Petri nets and UML notation.

MICHEL dos S. SOARES received his degree in Computer Science from the Federal University of São Carlos (Brazil) in 2000. He is currently doing his Master's Degree at the Federal University of Uberlândia (Brazil), where he is researching on UML, Petri nets and Linear Logic. He is also a Computer Science professor, in the field of Software Engineering.