AN APPROACH BASED ON DYNAMIC UML DIAGRAMS AND ON A TOKEN PLAYER ALGORITHM FOR THE SCENARIO VERIFICATION OF REAL TIME SYSTEMS

Stéphane Julia, Elisângela Mieko Kanacilo
Faculdade de Ciência da Computação,
Universidade Federal de Uberlândia,
P.O. Box 593, 38400-902, Uberlândia-M.G-Brazil,
email: stephjl@aol.com, ekanashiro@hotmail.com

***KEYWORDS*—Scheduling, Model analysis, Verification, Real Time, UML, p-time Petri Net**

ABSTRACT

The objective of this article is to present an approach based on UML dynamic diagrams and on p-time Petri Nets for scenario verification of Real Time Systems. The main idea consists of translating the sequence diagrams which express the initial specifications of the system in a unique p-time Petri Net model which represents the global behaviour of the entire system. A Token Player algorithm used for the scheduling problem of Real Time Systems and which can be seen as a simulation technique of a formal model is then applied to the obtained p-time Petri Net model for scenario verification. The approach is illustrated through an example of a batch system which can be seen as a particular case of a Real Time System.

I. INTRODUCTION

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 in parallel and conflict situations which have to be solved in real time (without a backtrack mechanism) can occur if the same non-preemptive resource is called at the same time for the execution of tasks which belong to different scenarios.

Among all Object Oriented notations, UML [OM 1999] is one of the best accepted in 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. Some tools associated with the UML notations allows one to simulate the UML dynamic diagrams. 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 verify that the execution of several sequence diagrams simultaneously does not lead to a deadlock situation. Another limitation is that UML does not provide formal notations for specifying the time constraints of Real Time Systems like date intervals or time-limited offerings.

Petri Nets [Murata 1989] are very suitable to model Real Time Systems, as they allow for a good representation of conflict situations, shared resources, synchronous and asynchronous communications, precedence constraints and explicit time constraints, in the time Petri Nets 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 main idea of this paper is to present an approach that combines UML dynamic diagrams and a p-time Petri Net model for scenario verification of Real Time Systems.

II. SPECIFYING REAL TIME SYSTEMS USING SEQUENCE DIAGRAMS

The batch system in figure 1, which can be seen as a particular Real Time System, will be used 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 the operations. This production system executes two different recipes. Both of them use common equipment,that being the thermal exchange TE3, the R3 reactor and the thermal exchange TE4 in certain steps of their execution.

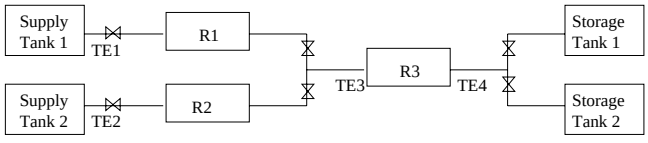

Fig. 1. Batch Production System

A sequence diagram shows the chronological order of the operations and the interaction between the objects for a particular scenario. The sequence diagram of figure 2 is used to represent the scenario corresponding to Recipe 1. Initially the object batch 1 calls the method of R1 for the processing operation. Then the 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 recipe and the object batch 1 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 batch1 from reactor R1 to reactor R3. Once batch 1, is transferred 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 the 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 for Recipe 2 is similar to the one of Recipe 1. To build it, it is necessary to change the objects batch 1,TE1 and R1 from the sequence diagram shown in figure 2, for the objects batch 2, TE2 and R2 respectively.

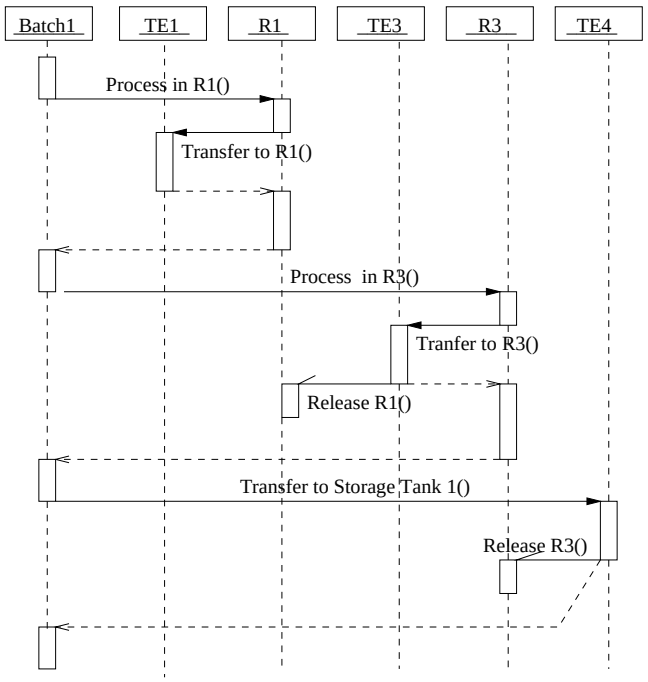

Fig. 2. Sequence diagram for recipe 1

By analysing sequence diagrams separately, it is not possible to verify if a conflict situation can occur that could be dangerous. For example, during real time execution, both sequence diagrams may have to request some common objects at the same time interval. Another limitation of this specification model is that explicit time constraints like the initial date of a scenario execution do not appear on the diagram. The following part of this article will show how to deal with these limitations using a p-time Petri Net model.

III. PETRI NET BASED MODELING

A. Autonomous Petri Net model

It has been shown in [Kanacilo 2002] that Petri Nets allow one to describe synchronous and asynchronous communication mechanisms between objects. Based on the interactions between the objects specified in the sequence diagram of figure 2 and merging all the Petri Net objects involved in the scenario execution, the Petri Net model of figure 3a is obtained. On this Petri Net model, each object called by the object batch 1 of the sequence diagram is represented by a non-preemptive resource. The Petri Net model corresponding to Recipe 2 can be obtained in a similar way and is shown in figure 3b. In order to obtain the global model which corresponds to the whole system (Recipe 1 + Recipe 2), the models of Recipe 1 and Recipe 2 can be merged through the common places TE3, R3 and TE4 as illustrated in figure 4. The obtained global model shows the global behaviour of the entire system without explicit time considerations.

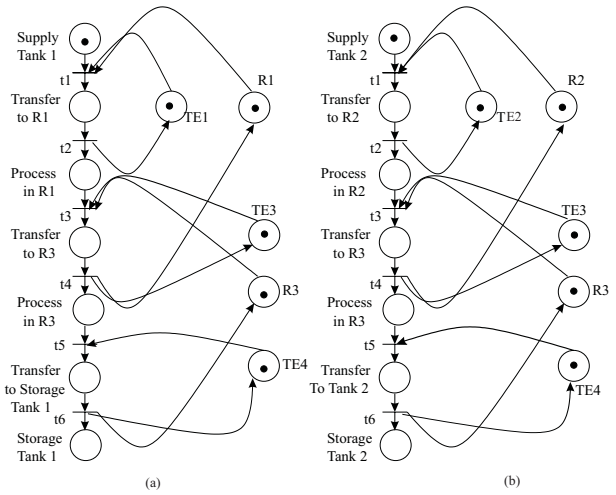

Fig. 3. Petri net model for recipe 1 and recipe 2

B. p-time Petri Net model

As was shown in [Julia 2000], explicit time constraints which exist in a batch system can be formally defined using a p-time Petri Net [Khansa 1996]. With this kind of time Petri Net, a static interval $I_{p_i} = [a_i, b_i]$ associated with each place p_i has to be defined. It represents the permanency duration (sojourn time) of a token in p_i. In the batch system case, these intervals represent the time the batches remain in reactors or in intermediary buffers. The dynamic evolution of a p-time Petri Net depends on the marking of the net and on the time situation of the tokens which is given by visibility inter-

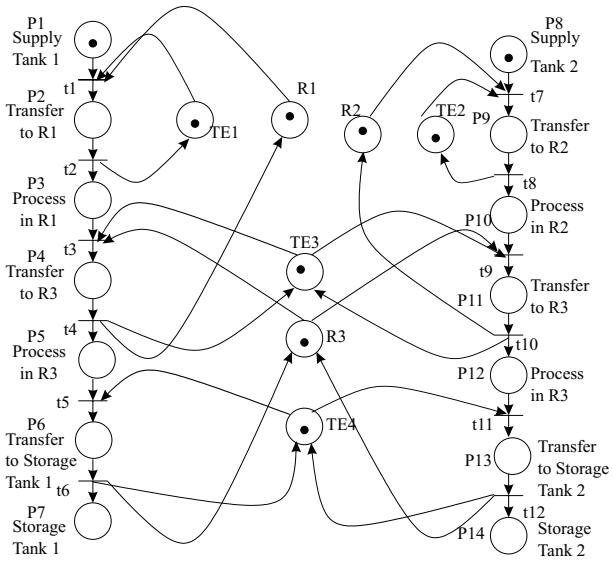

Fig. 4. Global model

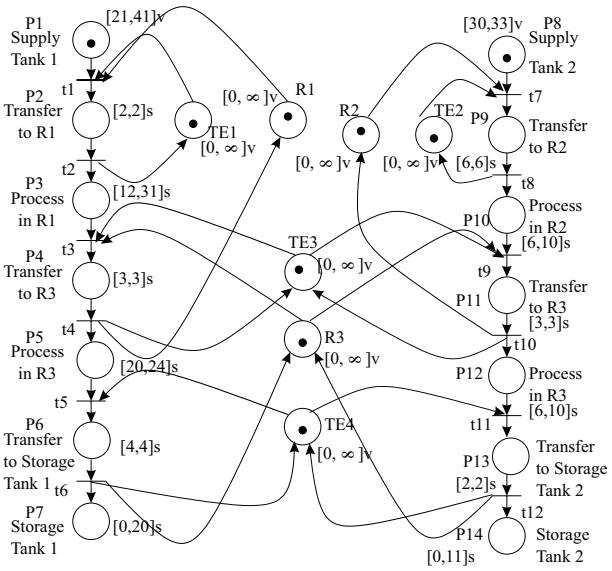

Fig. 6. p-time Petri Net model

vals $[(\delta_p)_{min}; (\delta_p)_{max}]$ associated with each token of the net. In particular, a visibility interval associated to a specific token represents the earliest date $(\delta_p)_{min}$ when the token in p becomes available for the firing of a transition and the latest date $(\delta_p)_{max}$ after which the token becomes non-available ("dead") and cannot be used for the firing of any transition.

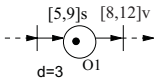

Fig. 5. Visibility interval

For example, in figure 5, if the arrival date of the token in place O_1 is $\delta = 3$, knowing that the static interval of this place is [5,9]s, then, the visibility interval of this token is [5+3, 9+3]v=[8,12]v. In the context of batch systems, the "death" of a token means a time constraint has been violated and the corresponding batch has been damaged.

Considering a specific production plan, the static and the visibility intervals can be associated to each place of the global model and the p-time Petri Net of figure 6 is obtained. The visibility intervals associated with the starting places of each recipe represent the time interval (interval of dates) inside which a scenario corresponding to a recipe must be iniciated.

IV. SCENARIO VERIFICATION

One of the approaches which permits 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 possibilities of deadlock which can be caused by a "token death". The token player algorithm presented in [Julia 2000] which is used for the scheduling problem of Real Time Systems allows one to treat conflict situations in real time when the system is modelled by a p-time Petri Net. One of the particularities of this algorithm is that it does not fire transitions, necessarily, at the earliest dates. It works on a short temporal projection that permits one to emphasize possible future conflicts. Applying this algorithm to a p-time Petri Net, an acceptable scenario is obtained. Before explaining how this algorithm works, it is necessary to define what a conflict is for a p-time Petri Net.

A. Conflict for a p-time Petri Net

With a p-time Petri Net, conflicts for shared resources are visible during a time interval and not only at a single time point. The definitions (enabling interval of a transition and conflict time interval) necessary to understand the conflict notion of a p-time Petri Net were given in [Julia 2000]. Figure 7 helps one to remember these definitions. The static intervals associated with the places P1, P2 and P3 are the following ones :

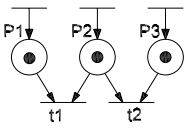

Fig. 7. Conflict for a p-time Petri Net

$[(d_{P1})\min;(d_{P1})\max]=[1;6]$
$[(d_{P2})\min;(d_{P2})\max]=[0;7]$
$[(d_{P3})\min;(d_{P3})\max]=[2;6]$

If at date 0 a token arrives in P1, at date 2 a token arrives in P3, and finally, at date 3, a token arrives in P2, then, the visibility intervals of these tokens are:

$[(\delta_{p1})min; (\delta_{p1})max]$=[1; 6]
$[(\delta_{p2})min; (\delta_{p2})max]$=[3; 10]
$[(\delta_{p3})min; (\delta_{p3})max]$=[4; 8]

The enabling interval of the transition t1 is then $[1; 6] \cap [3; 10] = [3; 6]$ and the enabling interval of the transition t2 is $[4; 8] \cap [3; 10] = [4; 8]$. The conflict time interval associated to the pair (t1;t2) is given by the intersection of the enabling intervals : $[3; 6] \cap [4; 8] = [4; 6]$.

Considering the p-time Petri Net of figure 6, if the transitions of the global model are fired as soon as they are enabled, at date δ=23 a token appears in place P3 and its visibility interval is [35;54]. The next event is the firing of t7 at date δ=30 which creates a new token in place P9 with its visibility interval [36;36]. At date δ=35, the token in place P3 becomes available for the firing of t3 which is in structural conflict with t9. It is then necessary to calculate the possible arrival date of a token in P10 between the dates 35 and 54 which correspond to the minimum and the maximum bounds of the visibility interval of the token in P3. After the firing of t8 at date δ=36, a token will appear in place P10 with a visibility interval equal to [36+6;36+10]=[42;46]. Since the visibility intervals of the resources are $[0;+\infty[$, the enabling interval of t3 is [35;54] and the enabling interval of t9 is [42;46]. The conflict time interval of the pair (t3;t9) is then equal to $[35;54] \cap [42;46]$=[42;46]. So, an effective conflict between t3 and t9 is able to occur during the interval [42;46]. As there are two shared resources TE3 and R3 that are input places for both transitions, two cases must be isolated and analysed separately. For example, the useful part of the net for conflict analysis of the shared resource R3 is represented in figure 8.

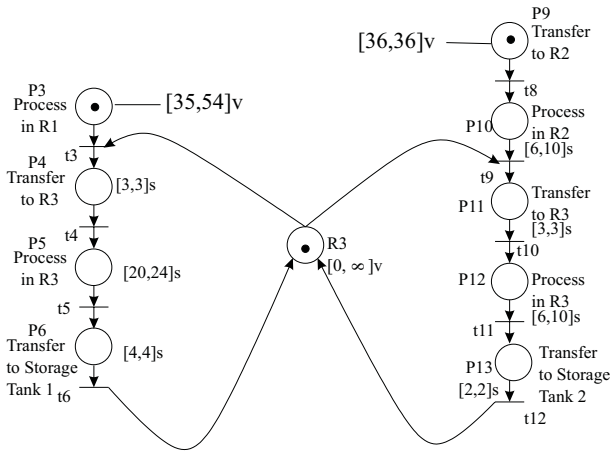

Fig. 8. Useful part of the conflict for R3

B. Conflict Resolution

Considering the conflict for resource R3 at date 35 illustrated in figure 8, it seems normal to fire t3 as soon as possible, since there is not token in P10 at that time. If t3 is fired at date 35, a new token appears in P4 and its visibility interval is equal to [35+3;35+3]= [38;38]. From this new state, the consequences of this decision (firing of t3 at date δ=35) on the rest of the net have to be analysed. In particular, it is important to verify that this decision will not cause a constraint violation, i.e. the death of a token. The tool which allows one to represent all the possible evolutions of a p-time Petri Net is the class graph [Khansa 1996].

Figure 9 represents the class graph of the net of figure 8 after the firing of t3 at date δ=35. The classes of the graph are the following ones:

$C_0 : M_0 = \{P4, P9\}, (3 \leq d_{P4} \leq 3) \wedge (1 \leq d_{P9} \leq 1)$
$C_1 : M_1 = \{P4, P10\}, (2 \leq d_{P4} \leq 2) \wedge (6 \leq d_{P10} \leq 10)$
$C_2 : M_2 = \{P5, P10\}, (20 \leq d_{P5} \leq 24) \wedge (4 \leq d_{P10} \leq 8)$

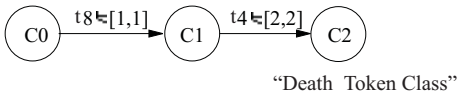

Fig. 9. Class graph for conflict of R3

The class C2 is a "death token class", which represents a deadlock in the case of a p-time Petri Net. In fact is is possible to see that for the class C2, the token in P5 has to stay in this place, at least for a duration equal to 20. The shared resource R3 will not be available before this duration for the firing of t9. On the other hand, the token in P10 can only stay in this place for a duration equal to 8 at most. If the token remains for a longer duration in this particular place, then the death of the token will occur and batch 2 will be damaged. As a consequence of this class graph, the transition t3 cannot be fired at date 35 when the token in P3 becomes available and the transition t9 will have to be fired before t3, expecting then that the death of the token in P3 will not occur later.

C. Token Player Algorithm

Figure 10 shows how the Token Player algorithm works. The algorithm has a decision making system which has to be used each time a conflict for a resource appears in the meaning of a p-time Petri Net model. The Token Player uses a calendar containing a sequence of events scheduled in time. These events are the minimum and the maximum bounds of all visibility intervals. Each time a minimum bound of a visibility interval is reached in the calendar, it means that a token becomes available. If the corresponding token enables a transition and if the transition is not in structural conflict, then the transition is fired. If the transition is in structural conflict, the conflict state is isolated (the corresponding Petri

Net fragment) and the conflict resolution mechanism is called. If the conflict resolution mechanism allows one to fire the corresponding transition as soon as the token becomes available, then the transition is fired. If a maximum visibility interval is reached in the calendar, the death of a token occurs. The only solution is then to relax a constraint: increase the value of the maximum bound of a static interval or delay the beginning of one of the recipes increasing the values of the bounds of the initial visibility interval of the corresponding recipe.

Applying the Token Player algorithm to the global model of figure 6, the following result is obtained: 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.

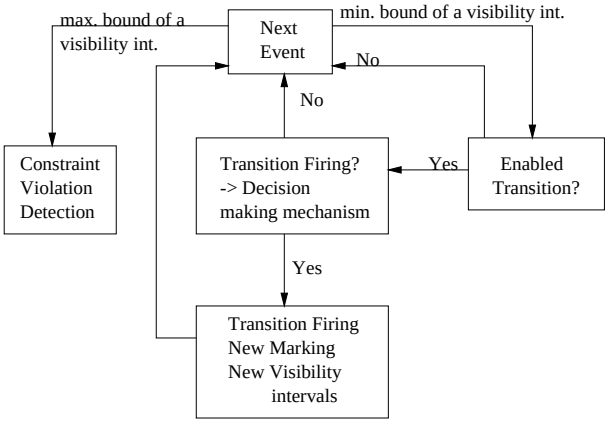

Fig. 10. Token Player Algorithm for a p-time Petri Net

As a result of the simulation, an acceptable scenario corresponding to the firing sequence is obtained. This scenario can be translated into the collaboration diagram illustrated in figure 11. In this diagram it is clear that batch 2 has to be processed in reactor R3 before batch 1.

V. CONCLUSION

UML dynamic diagrams are a very well accepted tool to model specifications through an Object Oriented approach. In particular, they are very well adapted for the specification of partial aspects of a system. Nevertheless, in the real time system case, they do not allow the detection, in an explicit way, of conflict situations. Most of the simulation tools available at this moment for the validation of specifications given through UML dynamic diagrams generally only allow one to simulate sequence diagrams separately and without taking into consideration explicit time constraints.

This article showed that the initial UML specifications (the set of sequence diagrams) can be translated into a unique p-time Petri Net which represents the global behaviour of the entire system with explicit time

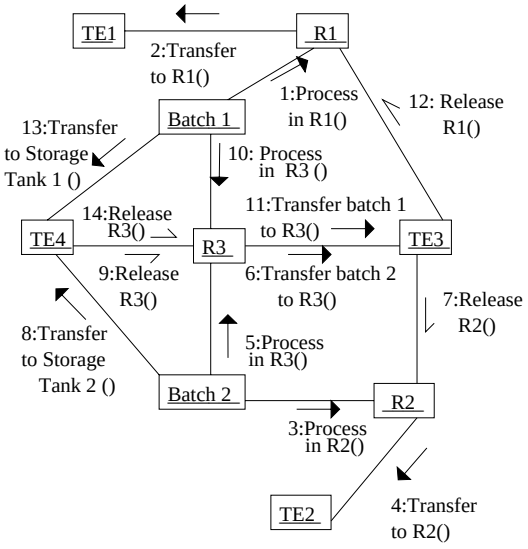

Fig. 11. Collaboration diagram

considerations. Based on this model, a Token Player algorithm, used for the scheduling problem of Real Time Systems was applied for scenario verification. The final result of the global simulation was given through a sequence of transition firings which can be translated in a collaboration diagram. The final diagram provides then an acceptable scenario which respects the initial set of constraints.

REFERENCES

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

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

[Kanacilo 2002] Kanacilo, E. M., Julia, S., (2002), "An UML/Petri Net approach for scenario verification of Real Time Systems", *Brazilian Petri Net Meeting*, Natal, Brazil.

[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.

[OM 1999] OMG (1999),"OMG Unified Modeling Language Specification version 1.3. Object Management Group".