Software Architecture for the Real Time Scheduling of Workflow Management Systems based on a Petri net model

Fernanda Francielle de Oliveira [1], **Stéphane Julia**[1]

[1]Faculdade de Computação, Universidade Federal de Uberlândia,
FACOM-UFU, Av. João Naves de Avila, 2160, P.O. Box 593, 38400-902
Uberlândia-M.G.-Brazil

fernandafrancielle@yahoo.com.br, stephane@facom.ufu.br

Abstract. *In this paper, an approach is proposed to solve the scheduling problem of Workflow Management Systems. The proposed approach uses an activity diagram to show the main activities and the different routings of the Workflow Process. Based on the obtained activity diagram, the corresponding p-time Petri net model is produced. Hybrid resource allocation mechanisms are modelled by an hybrid Petri net in order to represent discrete and continuous resources. Then, a token player algorithm is applied to an example of "Handle Complaint Process" in order to obtain an acceptable scenario corresponding to a specific sequence of activities which respects the time constraints. Finally, a software architecture is proposed for the real time scheduling of Workflow Management Systems.*

Resumo. *Este artigo apresenta uma abordagem para o problema do escalonamento dos Sistemas de Gerenciamento de Workflow. A abordagem proposta utiliza um diagrama de atividades para mostrar as principais atividades e os diferentes roteiros do processo. Baseado no diagrama obtido, o modelo de rede de Petri correspondente é produzido. Mecanismos de alocação de recursos híbridos são modelados por uma rede de Petri híbrida a fim de representar recursos contínuos e discretos. Em seguida, um Jogador de Redes de Petri é aplicado ao exemplo de um "Serviço de Reclamações" a fim de obter um cenário admissível correspondente a uma sequência de atividades que respeita as restrições temporais. Finalmente, é proposta uma arquitetura de software para o escalonamento em tempo real dos Sistemas de Gerenciamento de Workflow.*

1. Introduction

The purpose of Workflow Management Systems [Aalst and Hee 2002] is to execute Workflow Processes. Workflow Processes represent the sequence of activities which have to be executed within an organization to treat specific cases and to reach a well defined goal.

Of all notations used in the Software Industry, UML [OMG 2005] is one of the best accepted. In particular, the activity diagrams of the UML notation seem to be very suitable for proposing approaches to represent Workflow Processes as these diagrams represent basic routings encountered in these Processes which are the sequential routing (the sequential execution of activities), the parallel routing (two or more activities executed

simultaneously) and the selective routing (when a choice must be made between two or more activities).

Several authors have already used UML notations for the specification of Workflow Processes [Eshuis 2002] and [Hruby 1998]. However, UML notations have their limitations when they are used for specifying the real time characteristics of Workflow Management Systems. For example, activity diagrams do not represent real time constraints in a formal way and they do not show in an explicit manner resource allocation mechanisms. As a matter of fact, late deliveries in an organization are generally due to the problem of resource overload. Consequently, the model used at a Workflow Management System level should consider resource allocation mechanisms. In particular, time management of Workflow Processes is crucial for improving the efficiency of Business Processes within an organization.

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 quantitative time constraints in the time Petri net case. Some authors have already considered Petri net theory as an efficient tool for the modelling and analysis of Workflow Management Systems [Aalst and Hee 2002], [Covès et al. 1998], [Li et al. 2003], [Wirtz e Giese 2000].

The dynamic behavior of a system imposes a scheduling of control flow. The scheduling problem [Esquirol and Lopez 1995] consists of organizing in time the sequence of activities considering time constraints (time intervals) and constraints of shared resources utilization necessary for activity execution. From the traditional point of view of Software Engineering [Pressman 1995], the scheduling problem is similar to the activity of scenario execution. A scenario execution becomes a kind of simulation which shows the system's behavior in real time. In the real time system case, several scenarios (several cases in a Workflow Management System) 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 activities which belong to different scenarios.

The fundamental difference between the traditional scheduling problem of Production Systems [Lee and DiCesare 1994] and the scheduling problem of Workflow Management Systems is the nature of the resources used to treat the activities [Julia and de Oliveira 2004]. In the Production System case, resources represent physical equipment and are represented by a simple token in a place. They are discrete type resources. In the Workflow Management System case, resources can represent physical equipment as well as human employees [Aalst and Hee 2002]. For example, it is possible to allocate a nurse in a hospital service to take care of several patients at the same time during her working day. In this case, the nurse could not be seen as a simple discrete token and a model based on an ordinary Petri net would not be able to represent the real features which exist at a Workflow Management System level. Some recent works [Aalst and Hee 2002], [Tramontina et al. 2004] emphasize the lack of results which significantly cover the characteristics of Workflow Management Systems when considering the scheduling problem.

In this paper, an approach based on UML notations and on a p-time hybrid Petri

net model is proposed to solve the scheduling problem of Workflow Management Systems.

2. Workflow Management Systems modelling based on activity diagrams

The "Handle Complaint Process" presented in [Aalst and Hee 2002] will be used to illustrate the specification activity of Workflow Processes. First, an incoming complaint is recorded. Then, the client who has complained and the department affected by the complaint are contacted. The client is approached for more information. The department is informed of the complaint and may be asked for its initial reaction. These two tasks may be performed in parallel. After this, the data is registered and a decision is taken. Depending upon the decision, either a compensation payment is made or a letter is sent. Finally, the complaint is filed.

The proposed approach uses an activity diagram to show the main activities of the system and the different routings of the Workflow Process. The corresponding activity diagram for the "Handle Complaint Process" is the one of figure 1. The advantage of such an example is that it shows the different kind of routings which can exist in a Workflow Process. Swim lanes are used on the activity diagram to indicate in an informal way the principal resources (human and/or equipment resources) used to treat the different activities.

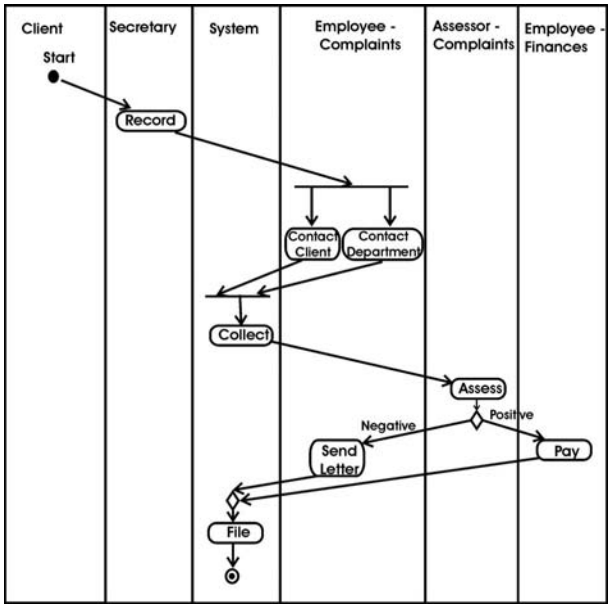

Figure 1. Activity Diagram

3. Workflow Management Systems modelling based on p-time hybrid Petri net

A way of guaranteeing that software specifications are consistent is to use formal methods. But, in practice, the software engineers, because of the complex mathematical definitions of formal notations, prefer to work with methods based on semi-formal notations, like the UML notation for example. It is then of fundamental interest to find a good compromise between the conviviality of semi-formal methods and the rigor of formal methods

by progressively incorporating formal notations within semi-formal specifications, in the context of a multi-formalism approach.

3.1. Routings definition based on ordinary Petri nets

Translating UML diagrams in Petri Net models, as was presented in [Cardoso and Silbertin-Blanc 2001] for example, allows one to define an operational semantic for UML dynamic diagrams in order to know how these diagrams are executed in real time. Nevertheless, it is important to emphasize the difficulty of producing algorithms that allow the automatic translation of a semi-formal model into a formal model. As a matter of fact, an activity diagram could only be seen as an intermediary level of specification between the textual specification of a Workflow Management System and the corresponding formal specification because of the limitations of UML notations for specifying real time characteristics such as resource allocation mechanisms for example.

Based on the activity diagram of figure 1, it is possible to obtain the corresponding Petri net model by representing each activity by a specific place with an input transition which shows the beginning of the activity and an output transition which shows the end of the activity.

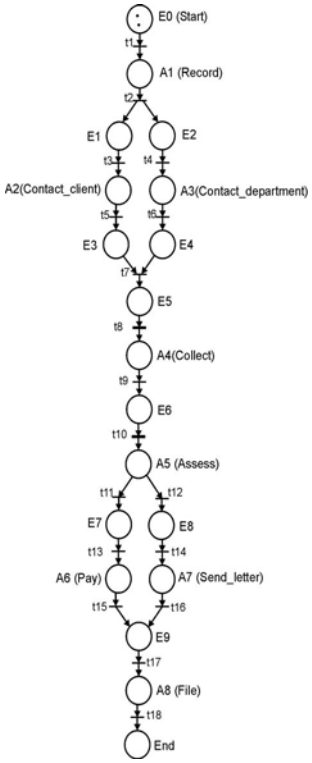

Figure 2. Routing Definition

Such a Petri net based model is shown in figure 2. The activities of the "Handle Complaint Process" are represented by the places A_i (for i=1 to 8) and the waiting times between the sequential activities are represented by the places E_j (for j=0 to 9). As a matter of fact, at the end of an activity, the next can only be initiated if the corresponding

resource is immediately available, which is not necessarily the case. For example, looking at the activity diagram of figure 1, at the end of the "record" activity, the activities "Contact Client" and "Contact Department" could only be performed if at least one employee of the complaints department is available. On the contrary, there will exist waiting times between the "record" activity and the activities "Contact Client" and "Contact Department". Such waiting times are represented in the Petri net of figure 2 by the places E_1 and E_2. The waiting places E_3 and E_4 are then necessary to synchronize the end of the activities "Contact Client" and "Contact Department". These waiting places are not represented in an explicit manner on the activity diagram.

3.2. Explicit time constraints based on a p-time Petri net model

As the actual time taken by an activity in a Workflow Management System is non-deterministic and not easily predictable, a time interval can be assigned to every workflow activity. The existing waiting times between sequential activities can be represented in the same way by an interval whose minimum and maximum bounds will depend on the earliest and latest delivery dates of the considered case (for each Client Complaint, there exists a specific case represented by a token in the Petri net model which represents the Workflow process). As was shown in [Julia and Valette 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 [Khansa 1996] 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 depends on the time situation of the tokens (date intervals associated with the tokens).

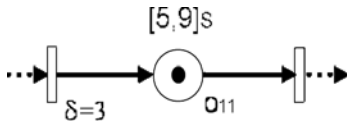

Figure 3. Visibility Interval

For example, in figure 3, if the arrival date of the token in the place O_{11} 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 Petri net of figure 4, static intervals are associated with the activity places. For example, the static interval $[20, 30]_s$ associated to the place A_2 which represents the "Contact Client" activity means that the employee who has to contact the client will need at the minimum 20 time units and at the maximum 30 time units. The static intervals associated with the "Collect" activity in A_4 and the "File" activity in A_8 are equal to $[0; 0]$ because the duration of these activities are negligible compared to other activities of the "Handle Complaint Process".

As was said before, the waiting times will depend on the earliest and latest delivery dates of each case. It is considered that the maximum allowed duration of a case for the "Handle Complaint Process" is 105 time units. Knowing the beginning date of a case, it is possible to calculate the previsional visibility intervals associated to the tokens in the waiting places using constraint propagation techniques classically used in scheduling problems based on activity-on-arc graphs [Esquirol and Lopez 1995].

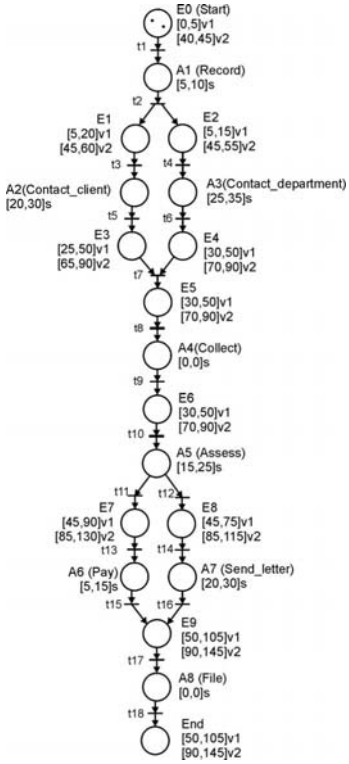

Figure 4. p-time Petri net

On the Petri net of figure 4, two cases represented by two tokens in place E_0 are considered. The first one can be initiated at date 0 which is represented as the minimum bound of the visibility interval v1 in E_0. The second case can be initiated at date 40 which is represented as the minimum bound of the visibility interval v2 in E_0. Because each case has to be processed in 105 time units at the maximum, the maximum bound of the previsional visibility intervals attached to the last place "End" of the process are equal to 0 (beginning date of case 1) + 105 (maximum duration of a case) = 105 (maximum date to deliver the first case) for the first case and 40 (beginning date of case 2) + 105 (maximum duration of a case) = 145 (maximum date to deliver the second case) for the second case. Bounds of other previsional visibility intervals attached to the waiting places are calculated applying propagation techniques. For example, the minimum bound of the visibility interval of case 1 in E_1 is obtained considering the minimum duration of the "Record" activity in A_1 making: 0 (minimum bound of the visibility interval v1 attached to E_0) + 5 (minimum bound of the static interval in A_1) = 5 (minimum bound of the previsional visibility interval v1 in E_1 corresponding to case 1). In the same way, the maximum bound of the visibility interval v1 in E_9 is obtained by considering the maximum duration of the "File" activity in A_8 making: 105 (maximum bound of the visibility interval v1 in the place "End") - 0 (maximum bound of the static interval in A_8) = 105 (maximum bound of the visibility interval v1 in E_9). Applying a similar reasoning to all waiting places and considering the different routings of the process, the minimum and maximum bounds of the waiting places for both cases (visibility intervals v1 for case 1 and v2 for case 2) are obtained as shown in figure 4.

3.3. Resource allocation mechanisms based on hybrid Petri nets

The last step of the modelling activity is to allocate the different types of resource used to realize the different activities. As already mentioned before, there exists different kinds of resources in Workflow Processes. Some of them are discrete type and can be represented by a simple token. For example, a printer used to treat a specific class of documents will be represented as a non-preemptive resource and could be allocated to a single document at the same time. On the other hand, some other resources can not be represented by a simple token. This is the case of most human type resources. As a matter of fact, it is not unusual for an employee who works in an administration to treat simultaneously several cases. For example, in an insurance company, one employee could treat normally several documents during a working day and not necessarily in a pure sequential way. In this case, a simple token could not model human behavior in a proper manner. As a solution to this problem, some of the human type resources will be represented by a real number (equal to one hundred when the resource is one hundred percent available) which will show the resource's availability. For example, if a maximum of ten patients can be allocated to a nurse in a hospital during her working day, the nurse's availability will be represented by a continuous resource equal to the real number one hundred and ten percent of this value will be allocated to each patient she will take care of.

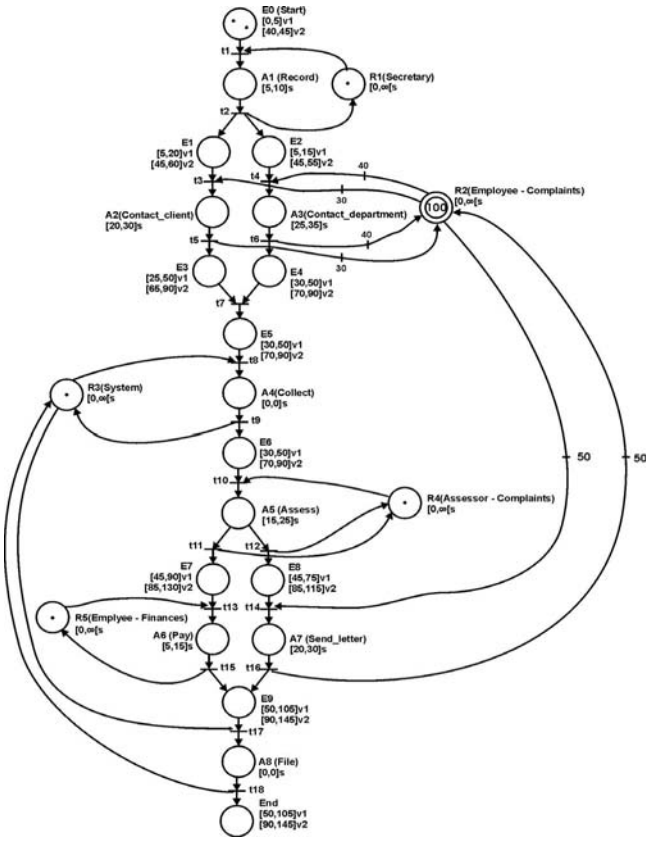

Figure 5. Resource allocation mechanisms

Hybrid resource allocation mechanisms (discrete + continuous resources) can be modelled by the hybrid Petri nets of Rene David and Hassane Alla [David and Alla 1992]. In particular, when considering Workflow Systems, all transitions of the hybrid Petri net

model will be discrete ones (none of the transitions of the model will be of continuous type).

The resource allocation mechanisms of the "Handle Complaint Process" are represented in figure 5. The "Secretary", the "Assessor-Complaints", the "Employee-Finances" and the "System" are represented as discrete resources. These resources will treat activities in a pure sequential way (one after the other). For example, the "Secretary" represented by a token in R1 can only register the data of a client at a single stroke and in a non-preemptive way. On the contrary, the employee of the complaints department is represented by the real number one hundred in R2. This continuous resource can be used to treat the activities: "Contact-Client", "Contact-Department" and "Send-Letter". The "Contact-Client" activity will need 30 percent of the employee's availability. The "Contact-Department" activity will need 40 percent of the employee's availability. And, finally, the "Send-Letter" activity will need 50 percent of the employee's availability. This means that, to a certain extent, the employee represented in place R2 could process more than one activity on a given period of time. For example, he could write a letter (50 percent of R2 for the"Send-Letter" activity) and, at the same time, try to phone a client (30 percent of R2 for the "Contact-Client" activity) to obtain more information about a case.

4. Scheduling principle

A natural approach to solve the scheduling problem of production systems represented by activity-on-arc graphs with disjunctive constraints (resource allocation mechanisms in the non-preemptive case) [Esquirol and Lopez 1995] is to combine a constraint propagation mechanism which allows for the calculation of admissible date intervals (visibility intervals in the p-time Petri net case) and a branch and bound algorithm whose goal is to find an optimal sequence of operations which respect the time constraints (date intervals). The limitation of such an approach is that the solution is generally given through a perfect sequence of operations which will be difficult to follow in real time when the system is a Workflow Process that considers human behavior.

Other scheduling approaches are based on token player algorithms which are specialized inference mechanisms applied in real time to Petri net models. The normal working of a token player algorithm is to fire transitions as soon as they become available. Such a strategy allows one to schedule the operations of a system in real time and gives the necessary flexibility to take into account possible perturbations (like delays for example). Nevertheless, it was shown in [Silva and Valette 1989] that it is not always the best strategy to fire transitions as soon as they are enabled. As a consequence, a traditional token player algorithm will hardly respect the time delivery constraints.

The approach proposed in this paper is based on a combination of time constraint propagation mechanisms (the ones used to obtained the visibility interval attached to the waiting places in the Petri net of figure 2) and a token player algorithm which uses a conflict resolution mechanism whose purpose is to calculate a sequence of activities that respects the disjunctive constraints (resource allocation mechanisms) as well as the time constraints (visibility intervals).

4.1. Conflict resolution principle

In the p-time Petri Nets, as presented in [Julia and Valette 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 6 (the Petri net fragment involved in the conflict for the shared resource R2), the conflict time intervals associated to the pairs of transitions (t3,t14) and (t4,t14) are given by the intersections of the visibility intervals $[45, 60]_{v2} \cap [45, 75]_{v1} = [45, 60]_{(t3,t14)}$ and $[45, 60]_{v2} \cap [45, 75]_{v1} = [45, 60]_{(t4,t14)}$. It means that effective conflicts between t14 and t3 and between t14 and t4 are able to occur during the date interval $[45, 60]$.

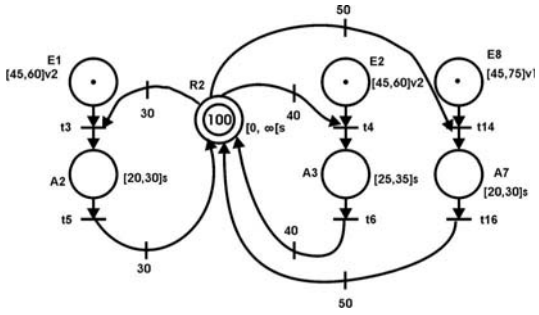

Figure 6. Conflict for a p-time Petri net

When a conflict is detected between two or more transitions in structural conflict, a decision mechanism has to look at a local level the consequences of a particular decision. For example, when considering the Petri net fragment of figure 6, if the decision of firing t14 at date $\delta = 45$ is made, then a token appears in A7 with the visibility interval $[20 + 45, 30 + 45] = [65, 75]_{v1}$ and the real number associated to the place R2 becomes equal to 100-50=50. From this new time situation, it is easy to see that a time constraint violation will happen on one of the visibility intervals associated to the tokens in E1 and E2. As a matter of fact, after the firing of t14, if t3 is fired before t4, a token is produced in place A2 with the visibility interval $[45 + 20, 45 + 30] = [65, 75]_{v2}$ and the real number associated to the place R2 becomes 50-30=20. In this case, the transition t4 could not be fired before 60 which is the maximum bound of the visibility interval in E2 because the real number in R2 will remain equal to 20 (which is less than the weigh necessary to fire t4) until date 65 (minimum bounds of the visibility interval in A2 and in A7) and a time constraint violation will happen. It is possible to show that after the firing of t14 at date $\delta = 45$, if t4 is fired before t3, a time constraint violation will happen in the same way with the visibility interval $[45, 60]_{v2}$ in E1. It is then possible to conclude that the firing of t14 at date $\delta = 45$ as soon as the token in E8 becomes available will not be allowed by the decision mechanism. The tool which allows one to represent all possible evolutions of a p-time Petri net is the class graph defined by Khansa [Khansa 1996]. Such a graph allows one to define the concept of "death token class" which corresponds to the possibility of a time constraint violation. The working of the decision mechanism based on the construction of a class graph when considering Petri net fragments involved in conflicts for shared resources was presented in [Julia and Valette 2000].

4.2. Token Player Algorithm

Figure 7 shows how the token player algorithm works. This 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. The token player uses a calendar containing a sequence of events scheduled in time. These events are the minimum and the maximum bounds

of the visibility intervals. Each time a minimum bound of a visibility interval is reached in the calendar, it eventually 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 (Petri net fragment) is isolated 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 time interval; in this case, there will not have the guarantee of respecting the delivery delays.

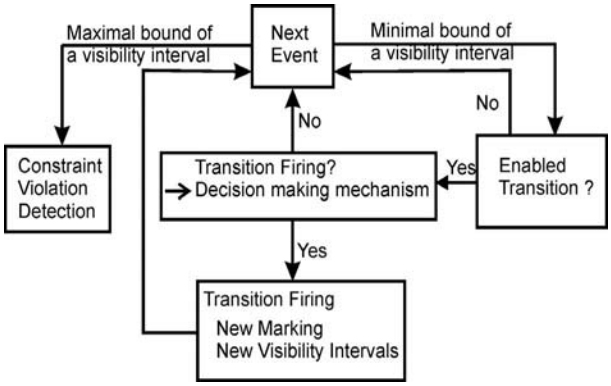

Figure 7. Token Player

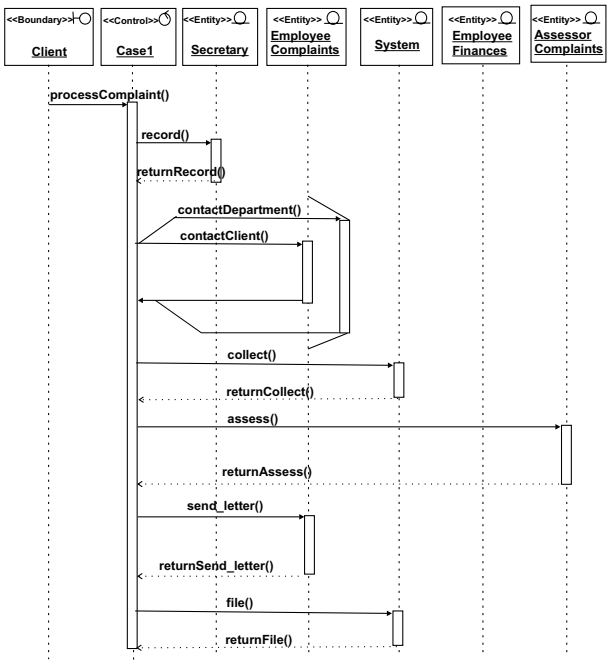

Figure 8. Sequence Diagram for case 1

When considering the previsional scheduling problem, for each case of the workflow process, a well defined scenario have to be specified. As a matter of fact, depending on the way the selective routings are treated, different scenarios can occur. The UML

diagram which allows one to specify scenarios is the sequence diagram which explicitly shows the chronological order of activities.

The sequence diagram in figure 8 is associated with the first case of the "Handle Complaint Process" (the token in place E0 in figure 5 with the visibility interval $[0, 5]_{v1}$). In this diagram, the object Case 1 associated with the UML stereotype "Control" represents the execution of case 1 when the complaint is not accepted (the compensation payment is refused). The objects associated with the UML stereotype "entity" are the ones which are informally defined through the swim lanes of the activity diagram in figure 1.

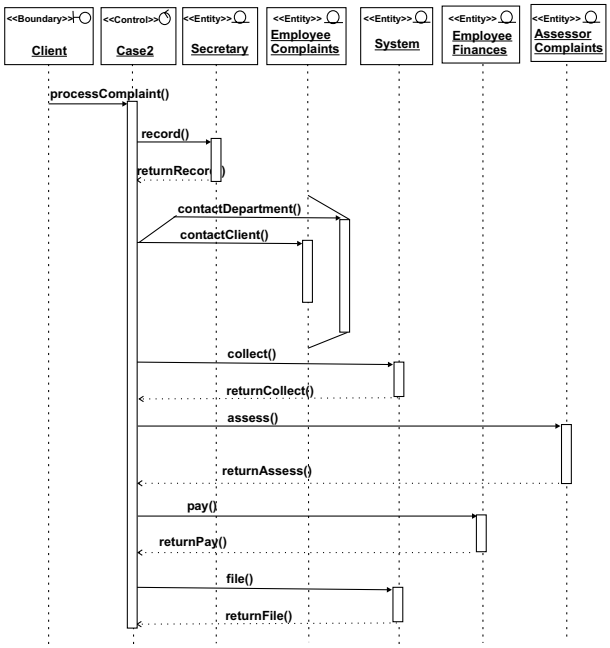

Figure 9. Sequence Diagram for case 2

The sequence diagram in figure 9 is associated with the second case of the "Handle Complaint Process" (the token in place E0 in figure 5 with the visibility interval $[40, 45]_{v2}$). In this diagram, the object Case 2 represents the execution of case 2 when the complaint is accepted (the compensation payment is accepted).

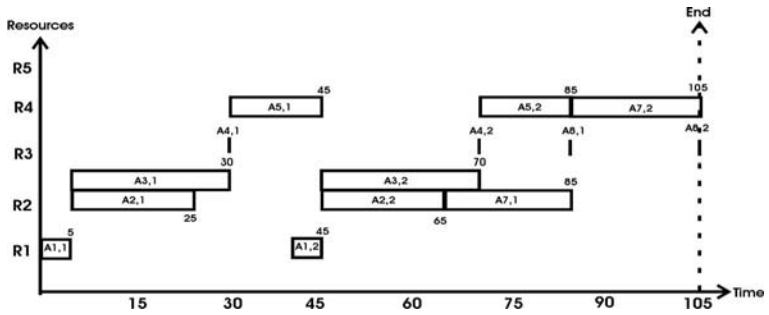

Figure 10. Admissible Sequence

Considering the scenarios specified by the sequence diagrams in figure 8 and 9, and applying the token player algorithm to the global model of figure 5, the result given

by the Gantt diagram of figure 10 is obtained. This diagram shows that the activity $A_{2,2}$ and $A_{3,2}$ (activities A2 and A3 of the second case of the Workflow Process) have to be processed before the activity $A_{7,1}$ (activity A7 of the first case of the Workflow Process) as was shown through the conflict resolution of the Petri net fragment in figure 6. It is also interesting to note that both cases respect their final visibility intervals in place "End": case 1 terminates on date $\delta = 85$ which belongs to the visibility interval $[50, 105]_{v1}$ and case 2 terminates on date $\delta = 90$ which belongs to the visibility interval $[90, 145]_{v2}$.

5. Software Architecture for the Real Time Scheduling

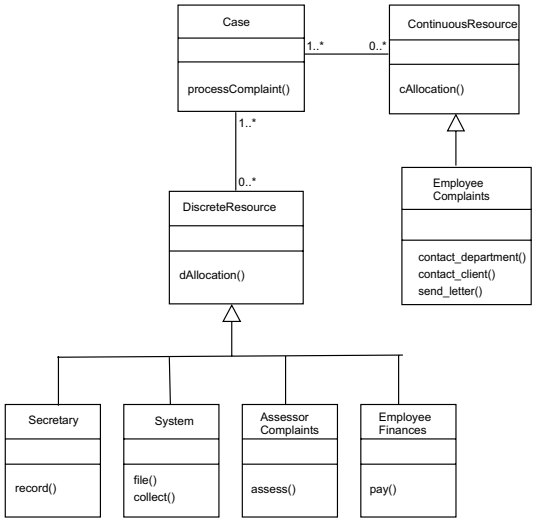

Figure 11. Class Diagram

One way of extending the results of the quantitative analysis (previsional schedule) to the design level is to consider a generic software architecture which allows one to apply the proposed scheduling approach (token player algorithm) in real time for the real time execution of workflow processes. In particular, the design of the chosen software architecture should show how components are structurally organized and the possible links between them.

Figure 11 shows how the objects of the sequence diagrams in figures 8 and 9 can be structured in a set of classes. The objects "Secretary", "System", "Employee Finances" and "Assessor Complaints" will inherit the characteristics of a discrete resource class whose main purpose is to allocate services in a discrete way through the method "dAllocation()". The object "Employee Complaints" will inherit the characteristics of a continuous resource class whose main purpose is to allocate services in a continuous way through the method "cAllocation".

Based on such a class structure, the collaboration diagram in figure 12 can be derived for the study of a specific case. It represents the existing links between the objects and the kind of communication. The methods "dAllocation" and "cAllocation" are called by the object Case 1 through asynchronous callings in order to allocate the continuous and discrete resources which will provide the methods necessary for the execution of the activities of the workflow process.

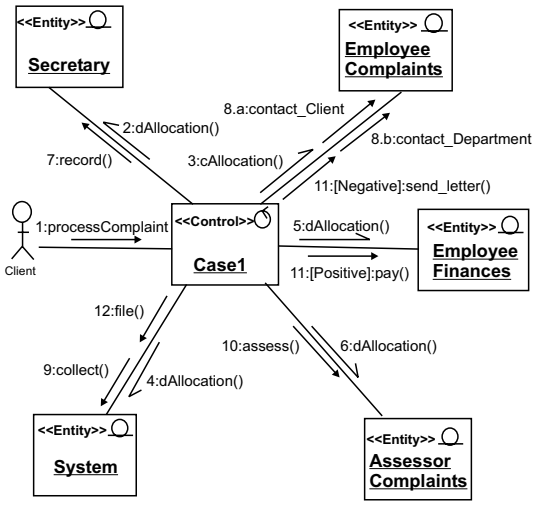

Figure 12. Collaboration Diagram for case 1

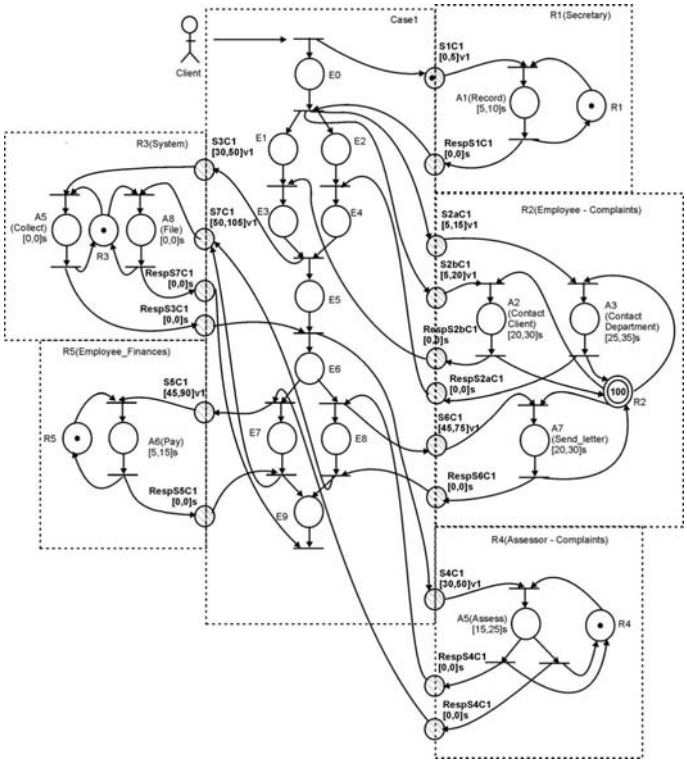

Figure 13. Petri net objects for case 1

After the last asynchronous calling from object Case 1, the Petri net model of figure 13 is obtained. Such a model shows the interactions between the different objects involved in the process of case 1. The internal behavior of each object is represented by a specific p-time Petri net model with continuous or discrete allocation mechanisms. The communication between the different Petri net objects is represented through communication places which are semaphore type. In particular, the methods necessary to execute the activities of the workflow process are called by the object Case 1 through synchronous callings as specified on the collaboration diagram in figure 12.

It would be easy to show that applying the reduction rules of the Petri net theory [Murata 1989] on the Petri net model of figure 13, eliminating the waiting places of the object Case 1 and some of the communication places, the global model of figure 5 would be obtained, which can be seen as a way of guaranteeing that the proposed software architecture will be equivalent to the analysis model used for the previsional schedule.

Finally, Figure 14 shows the internal behavior of the Petri net object R2 (Employee Complaints) when both cases (case 1 and case 2 in figure 2) are executed simultaneously. It is interesting to note that the mechanism used to solve conflict situations could be directly applied to this object which explicitly isolates a Petri net fragment. The visibility intervals associated with the communication places are calculated using the constraint propagation mechanisms at the level of the objects Case1 and Case 2 and are communicated to this object through asynchronous callings like the ones represented in the collaboration diagram of figure 12. Like that, the scheduling approach presented previously could be directly applied to such an architecture for the real time execution of workflow processes.

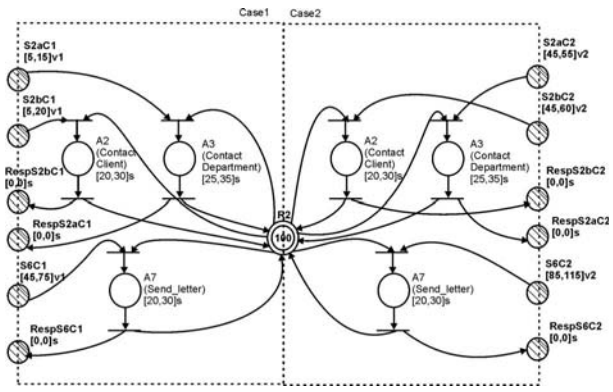

Figure 14. Petri net object "Employee-Complaints" for case 1 and case 2

6. Conclusions

This paper has shown that the initial UML specification (the activity diagram) of a Workflow Management System can be transformed into a unique p-time hybrid Petri net which represents the global behavior of the entire system with explicit time considerations and hybrid resource allocation mechanisms. Based on this model, a token player algorithm used for the scheduling problem of Workflow Management Systems has been applied. The final result has been given through an object based software architecture which allows one to apply the presented scheduling approach in real time for the real working of the Workflow Management System.

The advantages of such an approach are diverse. The fact of working with hybrid resources permits to represent the human behavior in a more realistic way. The originality of the time constraint propagation mechanism comes from the idea of combining on a same Petri net model duration intervals (static intervals) which represent durations of activities in a Workflow Process and date intervals (visibility intervals) which represent the waiting times of the cases between activities. The conflict resolution principle which can be used for shared hybrid resources allows to solve conflict situations in real time without a backtrack mechanism. Finally, the proposed software architecture is a way of

extending the result of the analysis level (previsional schedule) to the design level (real time schedule).

As a future work proposal, it will be interesting to study the possibility of introducing some fuzzy sets in the p-time hybrid Petri net model in such a manner that the natural flexibility of human behavior will be represented in a more expressive way at the Workflow Management System level.

References

Aalst, W. van der; Hee, K. van (2002). *Workflow Management: Models, Methods and Systems.* The MIT Press. Cambridge, Massachusetts. 368p.

Cardoso, J.; Silbertin-Blanc, C. (2001). Ordering actions in Sequence Diagrams of UML. *Invited talk in* 23^{rd} *International Conference on Information Technology Interfaces.* Croatia.

Covès, C.; Crestani, D.; Prunet, F. (1998). Design and Analysis of Workflow Processes with Petri nets. *In: 1998 IEEE International Conference on Systems, Man, and Cybernetics Proceedings of IEEESMC´1998.* vol. 1. p. 101-106.

David, R., Alla, H. (1992). *Petri Nets and Grafcet: tools for modelling discrete event systems.* Prentice Hall. 339p.

Eshuis, H. (2002). *Semantics and Verification of UML Activity Diagrams for Workflow Modelling.* PhD thesis, University of Twente, Enschede, The Netherlands.

Esquirol, P., Huguet, M.J., Lopez, P. (1995). (1999). Modelling and managing disjunctions in scheduling problems. Journal of Intelligent Manufacturing 6. pp. 133-144.

Hruby, P. (1998). Specification of Workflow Management Systems with UML. *In: Workshop on Implementation and Application of Object-oriented Workflow Management Systems.* Proceedings of the 1998 OOPSLA. Vancouver.

Julia, S.; de Oliveira, F. F. (2004). A p-time hybrid Petri net model for the scheduling problem of the Workflow Management Systems. *In: 2004 IEEE International Conference on Systems, Man and Cybernetics. Proceedings of IEEESMC´2004.* p. 4947-4952. The Hague, The Netherlands.

Julia, S.; Valette, R.; (2000). *Real time scheduling of batch Systems.* Simulation Practice and Theory. Elsevier Science. p. 307-319.

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.

Lee, D. Y.; DiCesare, F.k (1994). Scheduling flexible manufacturing systems using Petri nets and heuristic search. *IEEE Transactions on Robotics and Automation.*

Li, J., Fan, Y., Zhou, M. (2003). Timing Constraint Workflow Nets for Workflow Analysis. *IEEE Transactions on Systems, Man and Cybernetics. Part A: Systems and Humans. Vol. 33, N. 2, March 03.*

Murata, T. (1989). Petri nets: Properties, Analysis and Applications. *Proceedings of the IEEE.* 77(4), p. 541-580.

OMG - Object Management Group. (2005). “htp://www.omg.org”. Acesso em: 5 mar. 2005. 10(3). p. 123-132.

Pressman, R. S. (1995). *Software Engineering: A Practioner´s Approach.* 3 ed. Mc-Graw Hill.

Silva, M.; Valette, R. (1989). Petri nets and Flexible Manufacturing. *Advances in Petri nets (G. Rozemberg, Ed.), Lecture Notes in Computer Science. (Springer Verlag).*

Tramontina, G. B.; Wainer, J.; Ellis, C. (2004). Applying Scheduling Techniques to Minimize the Number of Late Jobs in Workflow Systems. *In: 2004 ACM Symposium on Applied Computing.* p. 1397-1403.

Wirtz, G.; Giese, H. (2000). Using UML and Object-Coordination-Nets for Workflow Specification. *In: 2000 IEEE International Conference on Systems, Man and Cybernetics. Proceedings of IEEESMC´2000.* vol.5, p. 3159-3164