PETRI NET BASED AGENTS FOR COORDINATING RESOURCES IN A WORKFLOW MANAGEMENT SYSTEM

Albert Plà, Pablo Gay, Joaquim Meléndez, Beatriz López

University of Girona, Campus Montilivi, P4 building, 17003 Girona, Spain
apla@eia.udg.edu, pgay@eia.udg.edu, beatriz.lopez@udg.edu, joaquim.melendez@udg.edu

Keywords: Workflow modeling, Workflow monitoring, Petri nets, Resources, Delay prediction

Abstract: We present a new framework for business process management based in a Petri net extension called Resource-Aware Petri Nets. This extension considers resources representation at the application level and allows the monitoring of the whole system with its dependencies. Thus, to solve resource usage conflict, agents are proposed to take care of monitoring workflow instances. This new comprehension of dependencies also allows the creation of a delay prediction system based in historical data from the workflows itself. In this paper we expose our methodology for modeling workflows through our extension which is based in classical approaches. Also a monitoring and delay prediction workflow is introduced and analyzed. In order to test our approach, we have extracted workflows from real cases and tested our framework simulating different kind of situations and resources, getting promising results since our prototype can provide early detection of workflows delays.

1 INTRODUCTION

Nowadays business process management is becoming a fundamental piece in many industrial processes. In today's economy, suppliers, manufactures and retailers are working together in order to reduce the production costs and to maximize the productivity. To manage the evolution and the interactions of the business actions it is important to accurately model the steps to follow in the process, the resources needed and the flow of the messages between the different parts involved (suppliers, manufacturers, clients, etc.). Workflows (WFs) provide a way of describing the order of execution and the dependent relationships between the constituting activities of the business processes.

Workflows usually model single and unique business processes, nevertheless, in real life environments, processes represented by workflows are rarely executed individually. Workflows are usually executed concurrently, sharing a limited number of resources sometimes even with external processes. In consequence, a delay in an ongoing workflow can impact other pending workflows, causing a cascade effect in the performance of the rest of the system due to dependencies or to resource occupation. For this reason it is important to monitor not only a single workflow execution, but also the whole system, as a delay can echo in the rest of executions. Conversely to previous works, the focus of this work is studying monitoring methods to deal with all the workflows in a environment (at the organization level). Monitoring means that the workflow management system (WMS) is aware of the states of the whole system regarding the current workflows actives and the resources available to carry them out.

Moreover, an intelligent monitoring methodology should be able to avoid, or somehow, reduce the effects of unexpected behaviors, so, corrective and preventive strategies are needed.

Regarding corrective strategies, when a workflow deadline is reached or close to be reached, a time out message or a running out of time alarm should be fired. For example, in (Blake, 2005), the authors provide a supporting tool to the user in order to modify running workflows Regarding preventive strategies, WMS should be able to predict when a workflow will fail before it happens. Preventive strategies are important as when a WF exceeds a deadline can cause important problems in the system. In critical

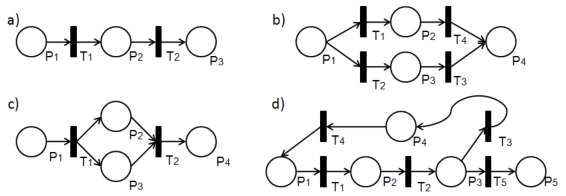

Figure 1: a) Petri net routing sequence. b) Petri net choice. c) Petri net parallel execution. d) Petri net iteration.

domains, such as medical device maintenance, a delay in a workflow could involve the unavailability of medical equipment causing delays on hospital operations, delays in surgeries and actually impacting on patients health.

In this line, the research on Artificial Intelligence have been centered in including new intelligent capabilities to WMS. E.g. CAKE (Bergmann et al., 2006) takes benefit of case based reasoning to improve workflow architectures and to increase the collaboration among them. Regarding the agent community, they are also interested in workflows but emphasizing its use for agent coordination.

Our work concerns both, Artificial Intelligence and agents, as provides intelligent capabilities to WMS for monitoring concurrent workflows. We extend previous Petri net models in order to add resource representation at the application level (instead of the workflow level, i.e. the resource required to develop a service or a task). We call them resource-aware Petri nets. This petri net extension allows us to monitor the state of all the running workflows of the system using agents. Resource-aware Petri nets (RAPN) also permit to predict possible delays without waiting until the workflow deadline so corrective and preventive strategies can be applied. Regarding the monitoring phase, we present an agent-based WMS which monitors workflows sharing resources inside an organization by individually monitoring the different workflows but using the whole environment information to diagnose the reason of unexpected or abnormal behaviors.

2 RELATED WORK

The lack of standardization in workflow representation has been a trending research topic during the last years. This absence of unification has led to a highly diversified types of workflow representations. Some authors use other fields' representation models such as UML activity diagrams (Kalnins and Vitolins, 2006), Business Process Management Notation (Muehlen and Recker, 2008) (BPMN) or different types of petri nets (called WF-nets) (Rinderle et al., 2004; Van der Aalst et al., 2003). Other researchers have chosen to develop specific languages for workflow representation such as YAWL (Brogi et al., 2006).

Regarding the resource representation in workflows, despite some languages such as the mentioned UML provide tools for representing resources itself, many workflow modeling languages do not integrate resources into its representations. A recent work on workflow representations that we should take into account is (Lombardi and Milano, 2010) which proposes the condition task graphs (CTGs). In a CTG, the arcs are labeled with probabilities. Tasks have resources associated. But due to the nature of the conditional branch of the graphs, the particular resources requirements for the execution of a given workflow can vary. Then the authors propose a methodology to optimize the resources requirements. From our perspective, such conditional representations could also be used for monitoring, without the need of specific workflow management system. The authors point out in their future work that their current research concerns the applicability of their approach to BPM.

Workflow monitoring has also been studied from different points of view. E.g. Van der Aalst et al.(Van der Aalst and Pesic, 2006) combine Petri nets (used to model the behavior of a service flow) and event logs (used to model real behavior of a service flow) in order to detect deviations and to store data for a further mining process. Then, in (Rozinat et al., 2009) the historical data is used for simulation, so that a short time projection can be obtained on the workflow outcomes.

As for the use of Petri nets for workflow monitoring, Frankowiak et al.(Frankowiak et al., 2009) developed a micro controller-based process monitoring in order to control the correct procedure of a manufacturing chain where every Petri net transition was linked to a micro controller input. The logistic field has also been a hot research topic when it comes to Petri net monitoring(Van der Aalst, 1993).

Other previous related works regarding to workflows monitoring come from the multi-agent community. For example, in (Wang and Wang, 2002) a multi-agent system is proposed to monitor the workflows associated to a given business process, so that they improve the system capabilities to deal with changes in the environment. In (Zarour et al., 2005) an agent based system is also proposed to deal with coordination and management of workflows between virtual enterprises.

3 BACKGROUND

In this paper high level Petri nets are used to model and monitor workflow services. For this reason, in the next paragraphs we introduce basic Petri net notation and terminology, which were presented by C.A. Petri (Petri, 1962).

The simple or classical Petri net can be defined as a directed bipartite graph with two kind of nodes called places and transitions which are connected by arcs. A place p is called an *input place* of transition t if exists an arc that directly connects p to t. A place p is called an *output place* of transition t if exists an arc that directly connects t to p. Moreover, arcs cannot connect two nodes of the same class. Places can contain tokens. During the Petri net execution, the position and number of tokens may vary. In the graphical representation places are drawn as circles, transitions are rectangles or bars, tokens are represented black dots and arcs by arrows (see Figure 1).

A transition t is enabled when each input place p of t contains one or more tokens. An enabled transition can be *fired*. Firing the transition t erases one token from t input place and creates a new one to its output place. The state of a Petri net is defined by the distribution of its tokens along the net, this can also be referred as *marking*. More information about Petri nets bases and history can be found in (Murata, 2002).

In order to include different domain particularities such as time or priorities, Petri nets have been enriched with extensions which represent theses different domain particularities. This new kind of nets were called *high level Petri nets*. In the classical Petri net, tokens have no kind of information incorporated, in consequence, it is impossible to distinguish between them. In practical terms, if a token corresponds to the status of real-life workflow, when two tokens are inside the same Petri net there is no possibility to relate each token with its correspondent flow. Using the classical representation, the only way to discern between both tokens/process is to duplicate the Petri net and to put each token in the different nets, which presents a problem for real process modeling as when different kinds of processes appear into a workflow, the size of the Petri net increases significantly. In order to avoid this duplication, the *colored Petri net* extension was created(Van der Aalst, 1993). Colored Petri nets assign a type or an identifier to each token so the confusion between tokens disappears.

Another common extension for the classical Petri nets is the inclusion of time which can be included in different ways. *Transition-timed* (T-timed) Petri nets (PN) associate time to the transition. In T-timed PN An interval of time can be assigned to each transition and they can only fire during this time interval, therefore, tokens remains at the input places at least until this time arrives. *Place-timed* PN associate time ts_i to the places. This means that when a token t arrives to a place p it must stay there at least ts_p time units although its transition fires. Finally, *token-timed* PN (or *dense-timed* PN)(Abdulla et al., 2006) is an extension of Petri nets in which each token is equipped with a real-valued clock so the time spent for every token can be registered. In this article, when timed Petri nets are mentioned is referring to this token-timed PN extension.

4 AGENT-BASED WORKFLOW MANAGEMENT SYSTEM

Our system is based on Petri nets for workflow modeling, taking into account the existing resources in the system. Workflows and resources are handled by the workflow management system (WMS), as shown in Figure 2. The main components of the WMS is the Monitoring System (MS) which is responsible of linking the WMS with the modeled process and also of monitoring the evolution of the workflow. The WMS access to the following data:

- Library of workflow models, which contains an archive of the workflows modeling the business processes of an organization or corporation.
- Resource data base, which stores the information related to the available resources of the system (type of resource, status, amount, etc.).
- Running workflows memory, which contains the state of the workflows currently running in the system.

With this information the WMS uses the following methodology to start and to monitor workflows:

1. Receives a request for a business activity A_i
2. Search in the workflow library for the pattern associated to A_i, $Pattern(A_i) = Wf_i$.
3. If Wfi is not running with other parameters in the WMS, then the WMS loads the workflow from the workflow library.
4. A new token t_i is created and placed into the corresponding workflow.

The key issue and the main contribution of this work concerns how workflows and resources are modeled using Petri nets and monitored by agents. The workflow modeling and monitoring are detailed below.

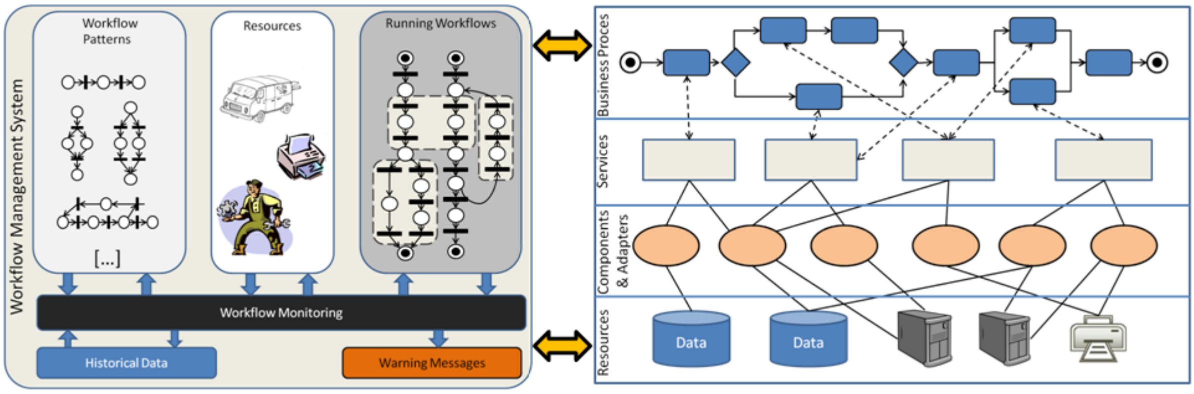

Figure 2: The workflow management system is responsible for modeling and monitoring the workflow and sends warnings when a possible delay is detected

4.1 Workflow modeling

Workflow modeling using high level Petri nets has been broadly studied among last years(Eshuis and Dehnert, 2003; Van der Aalst, 1998). As figure 1 shows Petri nets can represent typical workflow behaviors like sequential routings, parallel executions, choices, iterations, etc. Modeling a workflow with a Petri net is a simple task but it can take a long time as all the situations and details must be taken into account.

Definition 1 *A Petri net is a 3-tuple* $\langle P,T,A\rangle$ *where*

- *P is a finite set of Places*
- *T is a finite set of Transitions*
- $P \cap T = \emptyset$
- *A is the set of arcs which connect P with T and vice versa.* $A : (P \times T) \cup (T \times P)$

When modeling a workflow with PN, the most common type of transition is the one which represents tasks or services (e.g. *send_message*), it is fired just at the moment where the activity starts. When transitions represent the making of a decision they do not start any new service, they just chose if a path must be followed or not and they are fired by the system (or by the actor which takes the decision). External events (e.g. user inserts a coin into a printer) are also represented as transitions. The firing of the transition occurs when the event happens. Finally some transitions are just used for routing tasks (e.g. throw a concurrent execution of processes) and they are fired by the system. It is important to notice that there exists some dependencies between different transition types: a decision type transition always comes after a service type one; decision type transitions never come alone, there must be at least two complementary decisions so the WF net is path complete.

As mentioned above, in WF nets places represents *conditions* (some authors refers to *conditions* as *states*). A place indicates the status and the conditions of a workflow in a concrete point, in other words, a place p is the pre-condition of its input transition and a place p is the post-condition of its output transition.

Finally, in WF nets, tokens (which are colored) represent cases. Every time a token appears in the input place i means that a business new process p has started inside the workflow. Tokens are colored so the processes can be distinguished

Our work is specially focused on delays prediction, conversely, we need to take care of the kind and number of resources needed for every task inside the workflow so we can evaluate the time workflows will spend waiting for an available resource. In order to satisfy this requirement we extended the workflow net representation with a new *resource* element. We called this extension *resource-aware Petri nets* (RAPN, Definition 5). RAPN incorporate resources to high level Petri nets(Abdulla et al., 2006). Resources (Definition 2) are related with sets of consecutive transitions (forming subpaths, Definition 3) where the first transition (t_s) is the one which allocates the resource and the last (t_e) is the one which releases it. If there are not available resources of the required type by a transition (t_s) this transition cannot be fired until a resource of the desired type is released.

Definition 2 *A Resource is defined as a tuple* $\langle r,Q\rangle$ *where r is the kind of resource and Q the amount of resources of type r available in the system. Therefore R is a finite set of resources.* $R = \{\langle r_1,Q_1\rangle,\dots\langle r_n,Q_n\rangle\}$ *where n stands for the resources cardinal.*

Definition 3 *A Transition Subpath (TS) is the set of connected nodes between two transitions where* t_s *is the starting transition of the subpath and* t_e *the last one,* $TS = \langle t_s,t_e\rangle$.

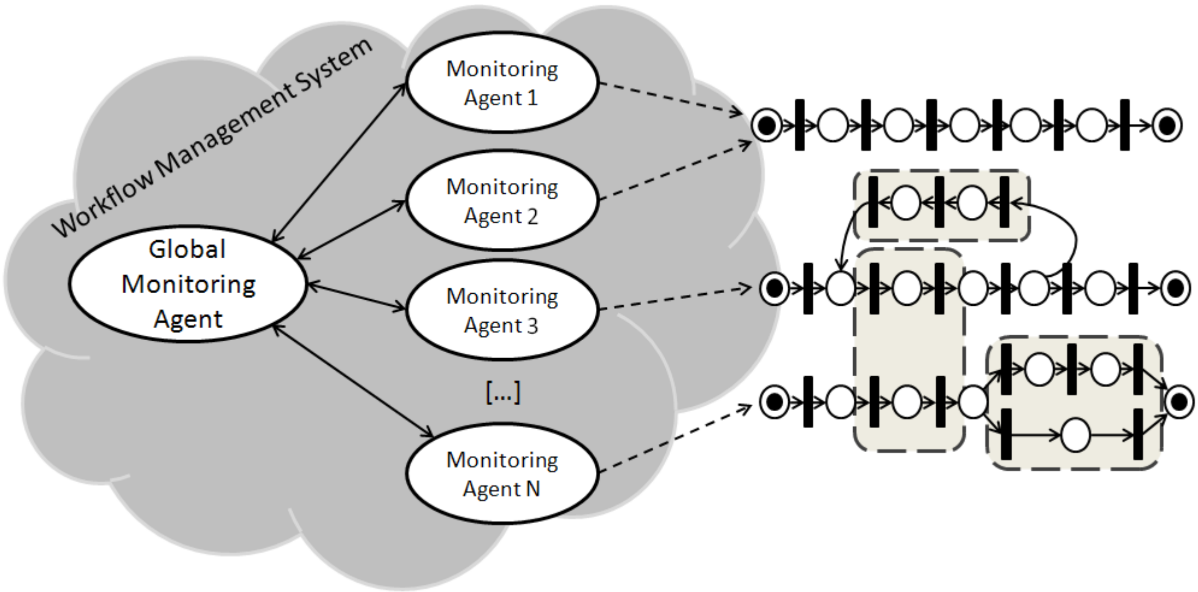

Figure 3: The workflow management system is responsible for modeling and monitoring the workflow and sends warnings when a possible delay is detected

Definition 4 *A transition subpath resource dependence (SD) defines the dependence between all the nodes of a subpath TS_i and a set of resources, $SD = \langle TS_i, \{\langle r_j, k_j \rangle\} \rangle$ where k_j is the amount of resources of type r_j needed.*

Definition 5 *A Resource-aware Petri net is a 6-tuple $\langle P, T, A, TO, R, D \rangle$ where*

- *P is a finite set of places*
- *T is a finite set of transitions*
- *$P \cap T = \emptyset$*
- *A is the set of arcs which connect P with T and vice versa $A : (P \times T) \cup (T \times P)$*
- *TO is a finite set of tokens which can store time information*
- *R is a finite set of Resources*
- *D is a finite set of transition subpath resource dependencies (SD)*
- *There exists an input place "i" and an output place "o" where:*
  - *Place i does not have any incoming arc.*
  - *Place o does not have any outcoming arc*
  - *Each node $n \in P \cup T$ where $n \neq i$ & $n \neq o$ has a path to o*
  - *Each node $n \in P \cup T$ where $n \neq i$ & $n \neq o$ has a path from i*

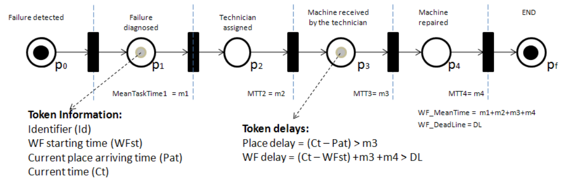

Figure 4: Delays in a workflow execution are estimated using the token information, the transition mean times and the workflow state.

4.2 Workflow Monitoring

Monitoring can be performed both at workflow and task level. Monitoring at the workflow level means that it is necessary to compare the evolution of the monitored workflow instance with the standard behavior of the generic workflow. Every time a new instance is started, a maximum deadline for the case resolution is assigned to it. When the workflow time execution is near the deadline, a warning message is send. However those warnings tend to arrive at the late phase of the workflow (even if they are caused for an early delay) so a restructuration of the workflow or resource addition may be difficult to implement. We consider than a lower-level monitoring of the workflow, in a task level, would result in an earlier detection of the delay.

For that purpose, we define the MS as a multiagent system with two different kinds of agents: *monitoring agents* which are responsible of collecting information about the development of a single workflow instance; and a *global monitoring agent* which is in charge of deciding if the different workflows are susceptible of suffering a delay or not.

In our approach, besides the global execution time of the workflow, we monitor the time that tokens spend on each place. Every time a workflow is instantiated, a *monitoring agent* (Figure 3) which can controls the information involving both the token and the workflow is thrown. The agent detect possible delays in the workflow (Figure 4) as it can notice when a task is exceeding it's estimated execution time. As a delay in the execution of a task does not necessarily represents a delay in the workflow execution, the *monitoring agent* uses the information stored on the token and the time required to execute the remaining workflow to decide if the workflow is susceptible to suffer a delay (Pla, 2010).

As the *monitoring agents* only have information of a single workflow, when a possible delay is detected, the agents send a warning message to the *global monitoring agent*. As several *monitoring agents* can be sending similar warning messages regarding a related problem (e.g. the lack of a concrete type of resource or a neck bottle in the workflow interaction design), the *global monitoring agent* is responsible for interpreting the different incoming messages. For that purpose it is endowed with a Complex Event Processing engine which reasons what is happening in the global workflow environment and tries to diagnose the reason of the delays (Gay et al., 2010), sending the corresponding alarms when necessary.

These alarms allow system supervisors to restructure the workflow or to endow the system with more resources in order to avoid the delays. Moreover, the study of these warnings with data mining and statistical techniques can offer information about the performance of the services and to detect which are the weaker points of the architecture.

5 RESULTS

This first prototype of our work consists in 3 modules: a workflow framework, a workflow simulation engine and the workflow management system (Figure 5). The workflow framework can load Petri nets defined by XML or load Petri nets designed with the PM Editeur(Zdenk et al., 2001) graphical editor; it is responsible for firing transitions, moving tokens along the Petri net and all the work related with Petri nets. Moreover it allows the user to define resources and to associate them with different parts of the Petri net. The workflow simulation engine permits to recreate

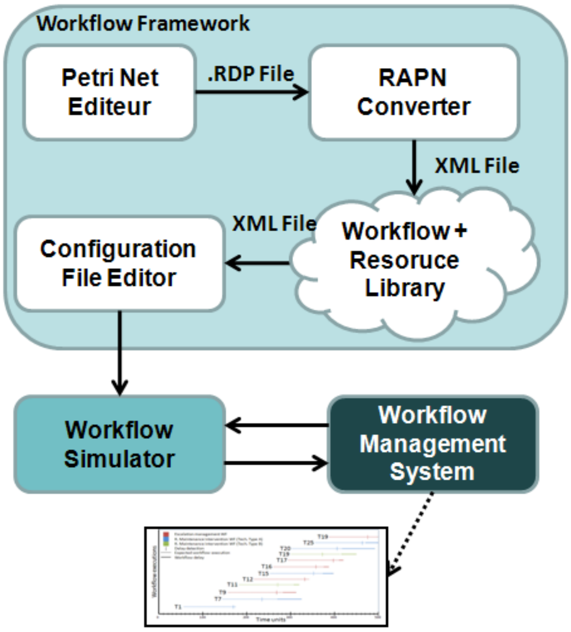

Figure 5: Architecture of the workflow simulation framework

the evolution of a workflow, given a set of parameters (the workflow modeling, the probability of a work instantiation, the standard deviation in the execution of a time and, number of resources in the system and the duration of the simulation) it simulates the execution of the workflow. Finally the workflow management system is responsible for monitoring the evolution of the workflow, detecting possible delays and to ask the workflow motor to trigger the transitions.

5.1 Experimental Setup

To test the performance of our system we modeled and simulated a set of workflows extracted from the AIMES project(AIMESproject, 2010). The workflows correspond to common activities in the medical device maintenance industry such as assigning a technician for a device repairing or reassigning a technician. Specifically the used workflows are *Reactive Maintenance Interventions (RMI)*, *Maintenance Event Escalation Management (MEEM)* and *Inventory and Installation of New Equipment (IINE)*.

The first one, RMI, describes the procedure to follow when a medical device throws a maintenance warning. In this case the system catch the warning and classifies the action, locates the source of the action, assigns a priority to the task and assigns the maintenance action to a technician. Finally the technician carries the action and the work ow finalizes. This workflow is composed by six tasks or services.

In this case, the resources needed to accomplish the workflow are technicians of a concrete type.

In MEEM the technical staff leader wants to assign a concrete task to an available technician. First of all, the staff leader looks which technicians are available and which tasks have not been assigned; then the staff leader defines a procedure to follow for a technician and finally the technician performs the assigned task (5 tasks). In this workflow two kinds of resources interfere in the development: the technical staff leader (which its amount will be always one as there is only one leader in each group) and technicians of a concrete type.

Finally, IINE describes the established procedure to follow when a new device arrives to a hospital. Firstly, a testing specialized technician makes the quality tests in order to check the proper working of the device and that all its documentation is attached, then the equipment is registered and installed. If the received equipment is a piece for an existing device an specialized technician embeds the equipment to its correspondent device, otherwise, an installer mounts the device where it corresponds. Nine different tasks are required for this workflow and three different kind of resources (although only two will be used at each instantiation).

Combining these workflows we performed experiments in three different scenarios. In the first one RMI and MEEM are simultaneously executed. We only defined two kind of resources: Technical staff leader (1 in the system) and Technician type A (4 in the system); This scenario allows us to study the behavior of our methods in a simple experiment. In the second scenario a new workflow is added in the system: a MEEM requiring a technician of type B. In consequence, a new type of resource was added (Technician type B) and the rest of resources is modified: 1 Technical staff leader, 3 Technician type A and 1 B. The aim of this experiment is to study the performance of the workflow management system in a more complex scenario. Finally, in the third scenario, the three defined workflows are used. In it, we considered the type D technician as the same which appears in the RMI and in the MEEM. As technician D is used by all the workflows we considered to include a high number of this type available resources in the system, having 5 type D technicians, 1 staff leader (used by MEEM) and 2 type I and T technicians (used by IINE). This scenario allows us to study the workflow management system when many resources are involved in the business processes. In all the scenarios we have simulated an interval of 500 time units with a workflow starting probability $p = 0.05$ per time unit. In the first two scenarios, the kind of WF created was randomly chosen with the same probability for each WF type, while in the last one the probabilities are $3/6$ for RMI, $2/6$ for IINE and $1/6$ for MEEM.

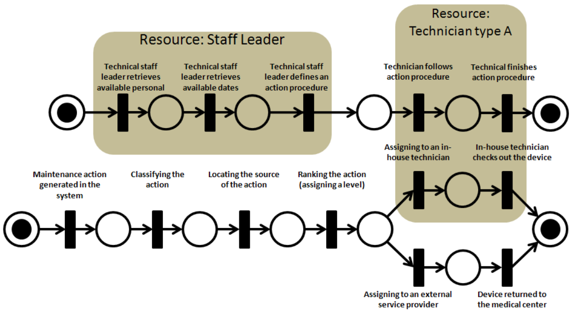

Figure 6: Top: Maintenance event escalation management. Bottom: Reactive maintenance intervention.

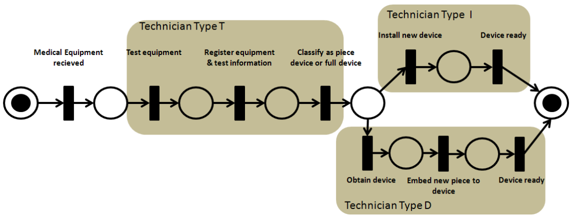

Figure 7: Inventory and Installation of New Equipment.

5.2 Results

The obtained results are shown as a flow execution diagram. The delayed workflows appear marked with their token identifier. The workflow executions are represented as lines where the dashed lines represents a normal execution (inside its maximum time of execution) and information of delayed plans are shown as solid lines. Moreover, the instants in which our tool predicted a delay for the workflow are marked with a vertical line.

Figure 8 shows the first scenario result. In this experiment two different workflows are sharing two different resources. The simulation among 500 time units generated 30 workflow cases where 7 of them resulted in a delay (T1, T8, T10, T15, T19, T22 and T26). All of them were predicted before they occurred by our system although 2 false positives (a delay was predicted but the workflow ended on time) were also predicted (T5 and T11). As it is a simple scenario the number of delays produced is small.

Figure 9 shows the second scenario result where the same two workflows share 3 different types of resources with a different quantity of them. The simulation generated 31 workflow cases where 10 finished out of time (T7, T9, T12, T15, T16, T17, T19, T20,

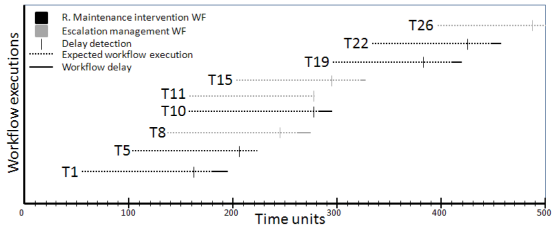

Figure 8: Result of the first scenario where the system resources are 4 type A technicians and a 1 technician staff leader.

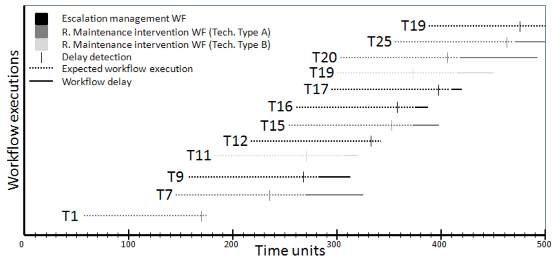

Figure 9: Result of the second scenario where the system resources are 3 type A technicians, 1 type B technician and a 1 technician staff leader.

T25 and T29). As happened on the previous scenario all the delays were successfully predicted, nevertheless, 2 on time workflows where classified as delayed workflows (T1 and T12). Moreover, due to the higher complexity of this experiment, it is important to notice that the number of delays respect the first scenario has increased.

Finally Figure 10 the results of the thirds scenario are shown. In it, the *installation and inventory of new equipment* is added to the first scenario. In the simulation 26 workflows have been instantiated and 12 of them have been marked as possible delayed workflows (T9, T11, T13, T14, T15, T16, T17, T18, T19, T21, T22 and T24). As both the complexity and the number of resources used in this scenario are higher than in the previous ones, the number of delays in the system is also higher. Ten of this marked workflows have been correctly classified as they have suffered a delay, while T14, despite the delay prediction, ended on time. The workflow defined by the T24 token has been classified as susceptible of suffering a delay, however the simulation ended before the workflow was delayed. Regarding the kind of workflows marked as delayed, 3 correspond to the IINE WF, 4 to the RMI WF and 5 to the EM WF.

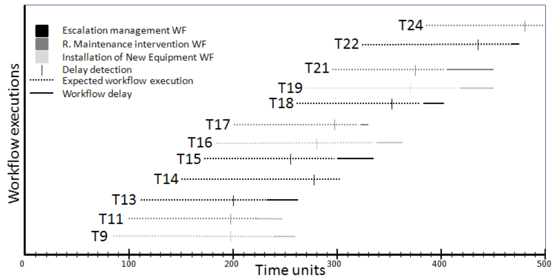

Figure 10: Result of the third scenario where the system IINE is added to the workflow environment.

5.3 Discussion

The obtained results in the different scenarios show that our prototype can provide an early detection of workflow delays. In some cases such as token 19(scenario1), token 7(scenario2) or token 22(scenario3) the detection is done up to 40 time units before the workflow deadline (32% of the workflow duration). This delay anticipation could be enough to restructure the scheduling of the workflow, especially in long duration workflows as medical device maintenance operations (which can have long term deadlines) or manufacturing processes (with midterm deadlines).

By comparing the first two presented scenarios we can notice that in the second one there is a higher number of delays. This fact is due to the lower number of available resources in the system. As more workflows are waiting for a resource to be released, more workflows may be delayed. The higher resource variety in the second scenario caused the ending of

PC\RC	Delay	On Time
Delay	7	2
On Time	0	21

(a)

PC\RC	Delay	On Time
Delay	10	2
On Time	0	19

(b)

PC\RC	Delay	On Time
Delay	10	2
On Time	0	14

(c)

Table 1: Table (a) shows the confusion matrix for the first experiment results. Table (b) shows the confusion matrix for the second experiment results. Table (c) shows the confusion matrix for the third experiment results.

Scenario	Monitoring agents used	Delays Produced	Delays Detected (TP)	Delays not detected (FN)	Erroneous Delay detection (FP)	On time predicted (TN)
SC. 1	30	7	7 (100%)	0	2 (8,7%)	21 (91,3%)
SC. 2	31	10	10 (100%)	0	2 (9,5%)	19 (90,5%)
SC. 3	26	10	10 (100%)	0	2 (12,5%)	14 (87,5%)
SC. 4	97	3	3 (100%)	0	2 (2,1%)	92 (97,9%)
SC. 5	52	10	10 (10%)	0	3 (7,1%)	39 (92,9%)
Mean	47,2	8	100%	0	7,98%	92,02%

Figure 11: Summary of the different scenarios executions.

some workflows that were instantiated after others. Despite this two remarkable differences, the delay prediction presented a similar behavior in both scenarios. The third scenario presents similar results to the second one. We can see how the coexistence of different types of WF and resources, as happened before, causes the ending of some workflows before older workflows have finished. It is important to notice that although the number of delays produced is the same, the number of erroneous predictions decrease respect to the second scenario.

Table 1 shows the confusion matrix of the three experiments presented above and Figure 11 a summary of the experiments performed. By analyzing the results we can notice that in any of the performed experiments appear false negatives (delayed workflows classified as on time workflows). This is an important fact as it means that all the delayed workflows are predicted. Regarding the false positives, we can see that there are 2 in each experiment. Taking into account that in each case we are monitoring around 30 workflows, this represents an 8% of the classified workflows, which is an acceptable percentage. Since our point of view, in the domain we are dealing with, a false positive is less harmful than a false negative as a false positive can result in a workflow checking by a supervisor while a false negative can produce a global delay on the system. Although the obtained results encourage us to follow this research direction, it is important to remember that the presented results were obtained from workflow simulations, not from real procedures. It would be interesting to apply the presented methodology to real data in order to analyze its performance in a real environment. It is also important to remark that the application of our proposed workflow monitoring system is conditioned to the knowledge of the business resources. Thus, it seems that could be straight applied inside a company, but can present difficulties in workflows involving external partners.

6 CONCLUSIONS

This paper has faced two important problems regarding workflow monitoring: how to model workflows including information about the resources needed to its execution and how to monitor a workflow for predicting possible delays in its execution.

For the first issue we presented the resource-aware Petri nets (RAPN), a Petri net extension based on workflow nets. RAPN are based in color dense-time Petri nets, which have been widely used to model workflows. Its main contribution is the addition of the resource concept, which is allocated when a concrete transition is fired and it is released when the last transition which needs the resource is fired.

Once the business processes are modeled, workflow management systems (WMS) are in charge for its monitoring. In this paper we proposed an agent-based workflow management system which monitors both the business process and the resource of the organization. Monitoring agents use RAPN to monitor the workflows in a task level so possible delays in the execution can be detected and predicted. When possible delays in the workflow executions are detected, a reasoning agent analyzes the information providing from the the different monitoring agents and diagnose the causes of the delays. This enacts the use of preventive actions such as prioritizing some executions or restructuring the workflow in order to avoid or minimize this delays.

To test our approach, we simulated a medical equipment maintenance organization. The simulations we ran fulfilled our expectations, indicating that an anticipated delay alarm can be predicted in many different situations. There were also some cases (around the 6% of the cases and 20% of the predictions) where the prediction alarm was thrown despite no delay was finally produced (false positive) while all the delays where succesfully predicted. In the tested domain the false negatives have a much higher cost than the fasle positives as they behave the impossibility of applying a preventive action, in this sense our approach had an apropriate behavior as any false negative appeared.

As a future work, the incorporation of historical data both into the delay prediction process and into the workflow delay prevention is an important point to consider. Regarding the coordination of agents, we should contemplate to use a distributed approach in the MAS instead of a centralized one.

ACKNOWLEDGEMENTS

This research project has been partially funded by the Spanish Government and FEDER funds through the projects labeled TIN2008-04547, DPI2009-07891 and CTQ2008-06865-C02-02, DURSI AGAUR SGR grant 2009-00523 (AEDS) and BR10/18 Scholarship of the University of Girona granted to Pablo Gay. The authors thank the representatives of the contributing enterprises of the EU-Project "AIMES" (ITEA2-07017), for the constructive subject-specific collaboration. The project is supported by the Spanish Avanza I+D program (support code TSI-020400-2008-47) within the EU-programme ITEA2.

REFERENCES

Abdulla, P. A., Mahata, P., and Mayr, R. (2006). Dense-timed petri nets: Checking zenoness, token liveness and boundedness. *CoRR*, abs/cs/0611048.

AIMESproject (2008-2010). Deliverable 1.3: requirements specification.

Bergmann, R., Fremann, A., Maximini, K., Maximini, R., and Sauer, T. (2006). T.: Case-based support for collaborative business. In *Proceedings of the 8th European Conference on CBR (ECCBR06), Springer LNCS 4106*, pages 519–533.

Blake, B. (2005). Coordinating multiple agents for workflow-oriented process orchestration. *Information Systems and E-Business Management*, 1(4):387–404.

Brogi, A., Popescu, R., Brogi, A., and Popescu, R. (2006). Bpel2yawl: Translating bpel processes into yawl workflows.

Eshuis, R. and Dehnert, J. (2003). Reactive petri nets for workflow modeling. In *Application and Theory of Petri Nets 2003*, pages 296–315. Springer.

Frankowiak, M. R., Grosvenor, R. I., and Prickett, P. W. (2009). Microcontroller-based process monitoring using petri-nets. *EURASIP J. Embedded Syst.*, 2009:1–12.

Gay, P., Pla, A., Lopez, B., Melendez, J., and ReginaMunier (2010). Service workfloow monitoring through complex event processing. In *Proceedings of the 15th IEEE International Conference on Emerging Techonologies and Factory Automation (ETFA)*, Bilbao, Spain.

Kalnins, A. and Vitolins, V. (2006). Use of uml and model transformations for workflow process definitions. *CoRR*, abs/cs/0607044.

Lombardi, M. and Milano, M. (2010). Allocation and scheduling of conditional task graphs. *Artificial Intelligence*, 174(7-8):500–529.

Muehlen, M. and Recker, J. (2008). How much language is enough? theoretical and practical use of the business process modeling notation. *Advanced Information Systems Engineering*, pages 465–479.

Murata, T. (2002). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

Petri, C. A. (1962). *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn.

Pla, A. (2010). Predicting and diagnosing delays in a workflow environment. Master's thesis, Universitat de Girona, Girona, Spain.

Rinderle, S., Reichert, M., and Dadam, P. (2004). Correctness criteria for dynamic changes in workflow systems: a survey. *Data Knowl. Eng.*, 50(1):9–34.

Rozinat, A., Wynn, M., der Aalsta, W. V., ter Hofstede, A., and Fidge, C. (2009). Workflow simulation for operational decision support. *Data & Knowledge Engineering*, 68(9):834–850.

Van der Aalst, W. (1993). Interval timed coloured petri nets and their analysis.

Van der Aalst, W. M. P. (1998). The application of Petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66.

Van der Aalst, W. M. P. and Pesic, M. (2006). Specifying, discovering, and monitoring service flows making web services process-aware. *BPM Center Report BPM-06-09, BPM Center*.

Van der Aalst, W. M. P., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G., and Weijters, A. J. M. M. (2003). Workflow mining: a survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267.

Wang, M. and Wang, H. (2002). Intelligent agent supported flexible workflow monitoring system. In *A. Banks Pidduck et al: CAISE 2002, LNCS 2348*, pages 787–791.

Zarour, N., Boufaida, M., Seinturier, L., and Estraillier, P. (2005). Supporting virtual enterprise systems using agent coordination. *Knowledge and Information Systems*, 8:330?349.

Zdenk, M. S., Svdov, M., and Hanzlek, Z. (2001). Matlab toolbox for petri nets. In *22nd International Conference ICATPN 2001*, pages 32–36.