Noname manuscript No.
(will be inserted by the editor)

Petri Net Based Process Monitoring

A Workflow Management System for Process Modeling and Monitoring

Albert Pla · Pablo Gay · Joaquim Meléndez · Beatriz López

Received: date / Accepted: date

Abstract Nowadays business process management is becoming a fundamental piece in many industrial processes. To manage the evolution and the interactions of the business actions it is important to accurately model the steps to follow and the resources needed by a process. Workflows provide a way of describing the order of execution and the dependencies between the constituting activities of business processes. Workflow monitoring can help to improve and to avoid delays on industrial environments where concurrent processes are carried out.

In this article a new Petri net extension for modeling together workflow activities with their required resources is presented: resource-aware Petri nets. Moreover an intelligent workflow management system for process monitoring and delay prediction is introduced.

Resource aware Petri nets include time and resources into the classical Petri net workflow representation, facilitating the task of modeling and monitoring workflows. The workflow management system monitors the execution of workflows and detects possible delays through resource-aware Petri nets.

In order to test this new approach, different services from a medical maintenance environment have been modeled and simulated.

Keywords Workflow · Petri Net · Workflow modeling · Workflow monitoring · Process monitoring · Resource-Aware Petri Nets

Albert Pla · Pablo Gay · Joaquim Meléndez · Beatriz López
University of Girona
Office 003/004 (eXiT Laboratory "La pizzeria"),
Building P4,
Campus Montilivi,
17001 Girona (Spain)
Tel.:+34 972 418391
E-mail: [albert.pla,pablo.gay,joaquim.melendez,beatriz.lopez]@udg.edu

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 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. The focus of this work is studying monitoring methods to deal with all the workflows in a environment (at the organization level). Monitoring means to be 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 method 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 [?], the authors provide a supporting tool to the user in order to modify running workflows. Regarding preventive strategies, a monitoring method should be able to predict when a workflow will fail before this happens. Preventive strategies are important since when a workflow exceeds a deadline can cause important problems in the system. In critical 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 any case, workflow monitoring is required to anticipate delays.

Our research concerns both workflow modeling and monitoring process in which shared resources are XXX and the target is detect, correct and prevent workflow deviations. The main contributions on the paper are: Petri net modeling which include shared resource definitions and a monitoring methodology for the early delay detection. this capabilities enhance future intelligent workflow management systems (i-WMS).

The scope of this work includes a widespread range of domains: not only manufacturing industrial processes, but also service oriented architectures, multi agent systems interactions, route scheduling or even device maintenance

planification. Particularly, our work is specially concerned with medical equipment maintenance business. We start from an infrastructure that gives support to the different parts involved in a maintenance operation process, and at the business level there are several workflows defined in order to minimize equipment downtimes. However, such optimization cannot be guaranteed if there is no way of monitoring the workflow status and to predict possible delays on their execution. Delays can be caused, for example, by an unattended request on behalf of the manufacturer support service, or by the sickness of a technician in charge of dealing with the maintenance operations. Those are human-dependent delays, but some others caused by resource overload due to concurrent execution of multiple workflows.

This paper is structured as follows: Firstly some basic concepts regarding workflows and Petri nets are given to the reader in order to facilitates the article comprehension; in Section 3 we present the state of the art regarding workflow modeling and monitoring; then, we introduce our intelligent workflow management system and the methodologies we developed to model and monitor both workflows and resources; In Sections 5 and 6 we present and discuss some of our research results; finally, we end the paper presenting the conclusion of our work and pointing the lines to continue this research.

2 Background

In this section we present a brief introduction to the workflow and the Petri net concepts and terminology in order to facilitate the reader the realized work.

2.1 Workflows

To manage the evolution and the interactions of the business processes it is important to accurately model the steps to follow in the activity, the resources needed and the flow of information between the different parts involved (suppliers, manufacturers, clients, etc.). Workflows provide a way of describing the order of execution and the dependent relationships between the constituting activities of the business processes [33]. Workflows usually run concurrently, sharing a limited number of resources which some times are provided by third party companies.

A ***workflow*** consists in a graph of interconnected actions which represents the tasks and interactions to be realized by a mechanism, a person, a staff, an organization, etc. Workflows can model business process, exchange of messages and software procedures or information.

A ***workflow*** instance is a workflow which is being executed in a concrete time instant. For example, a workflow can model the business process required to do maintenance in a medical equipment; then, when a medical device requires a maintenance operation, a workflow instance is created. When several workflows are coexisting in a common framework (e.g. an organization, an

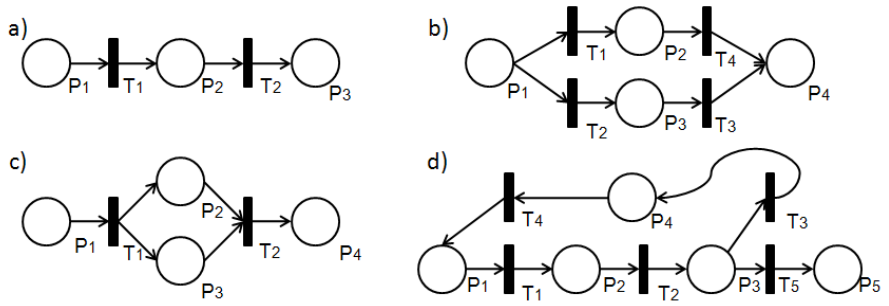

Fig. 1 a)Petri net routing sequence. b)Petri net choice. c)Petri net parallel execution. d)Petri net iteration.

industry, a server, etc.) where they share resources, actors or information we can refer them as a workflow environment. Workflows can be controlled and monitored by a ***workflow management system*** (WMS).

The WMS is responsible for monitoring the status of the different workflows and to store in a log the different events related to the workflow deployment. WMS can also be responsible of the assignment of resources to workflows and their schedules.

2.2 Petri Nets

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 tokens from t input place and creates new ones 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 [25].

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 are 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. Using the classical representation, the only way to discern between both tokens is to duplicate the Petri net and to put each token in different nets, increasing

significantly the size and complexity of Petri nets in real problems. In order to avoid this duplication, the *colored Petri net* extension was created[2]. 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)[7] 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.

3 Related Work

This section presents a brief state of the art concerning Workflow Modeling and Workflow Monitoring which are the two main issues which our work faces.

3.1 Workflow Modeling

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 (see Figure 2). Some authors use other fields' representation models such as UML activity diagrams[21] or different types of petri nets (called Workflow-nets)[28,4]. On the other hand other researchers have chosen to develop specific languages for workflow representation.

Unified Modeling Language[30] (UML) is a standardized modeling language used in the the field of software engineering. It allows to specify, to design and to document object-oriented software across several types of diagrams. The UML activity diagram describes the process of the different software activities step by step and their routing across different situations and cases. It offers different kinds of splittings such as OR-splits, AND-splits or conditional splits.These tools allows the activity diagram to represent almost any kind of information flow making the diagram able to describe the basic behaviors of a workflow. Many authors, such as Kalnins et al.[21], propose to use UML or to extend UML[10,36] in order to model workflow while others have discussed its suitability. Dumas et al.[14] analyzed the UML notation by representing the workflow patterns defined in [31] and concluded that, unlike alternative commercial languages, UML provides support for waiting and processing states and decomposition tools while they syntax and semantics

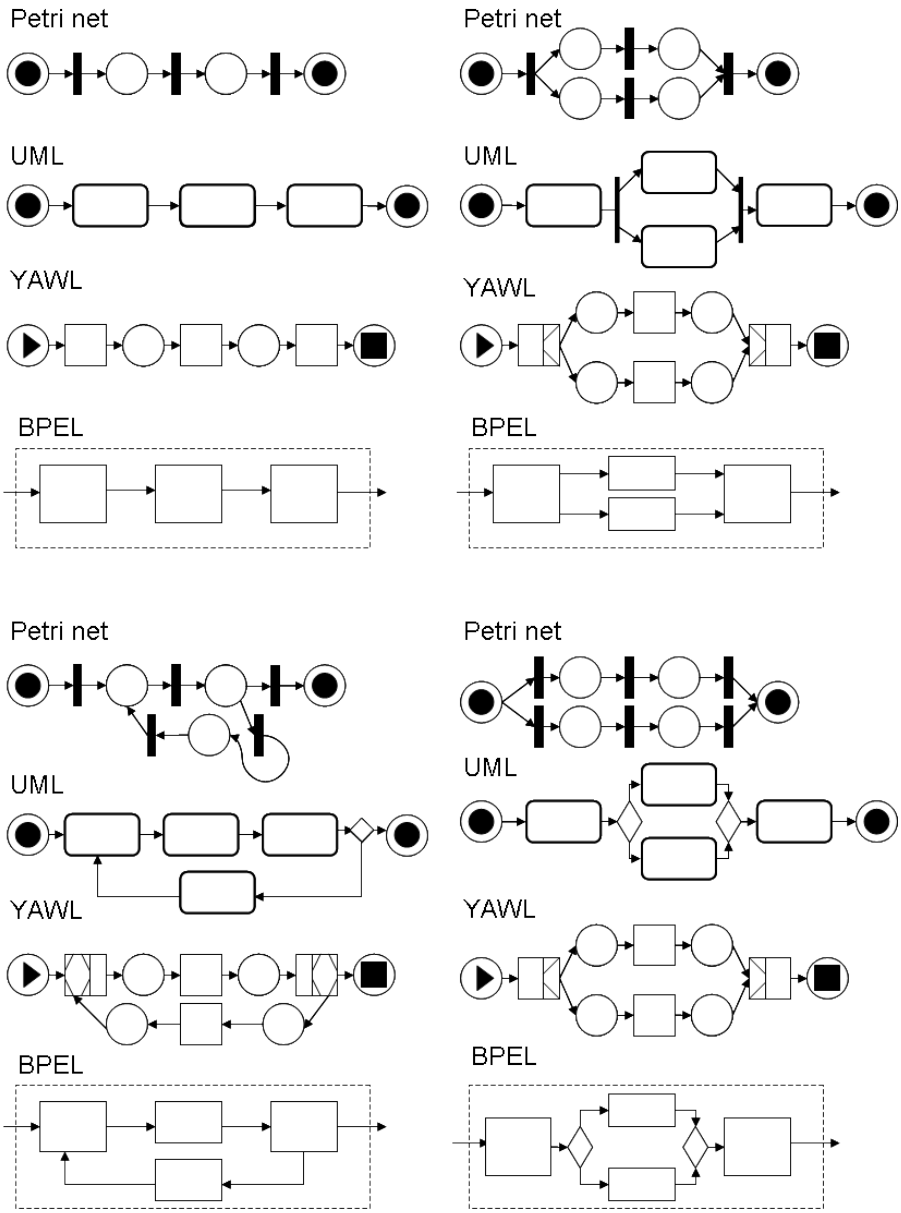

Fig. 2 Workflow pattern examples in different modeling languages. Top left: sequential rouging. Top right: concurrent execution. Bottom left: iterative routing. Bottom right: exclusive or choice.

presents a lack of precision for complex actions such as cancellation patterns or Multiple Instance Patterns and that UML doesn not fully capture important kinds of synchronization such as the discriminator and the N-out-of-M join.

Other fields modeling languages have been also used to represent workflows. A similar tool to UML for workflow modeling is Business Process Management Notation[24] (BPMN), it provides a graphical notation for specifying business processes in a Business Process Diagram. BMPN is highly understandable for almost any user familiarized with the workflow modeled, however its absence of mathematical and formal notation and the impossibility of linking BPMN with the execution process[20], reduce its possibilities. BPMN is closely related with BPEL (Business Process Execution Language)[1] as it can be directly translated to it. BPEL originally was created to model web service interactions but it has also been used to model workflows, specially in service oriented architectures (SOA)[26], but also in scientific processes[35] or to model grid computing interactions[35]. Moreover, another interesting point of BPEL, is that can be easily transformed to other languages such as YAWL[12], BPMN (mentioned above) or to Petri net struchtures[18] as Hinz et al. and Brogi et al. show on their works.

Petri nets[27](PN) are an established tool for modeling and analyzing processes[3]. On the one hand, Petri nets can be used as a design language for the specification of complex workflows; on the other hand, Petri net theory and notation provides powerful techniques for workflow analysis. Moreover, Petri nets can be extended to high level Petri nets[13] which allows a more accurate representation of the workflow representation. For example, Eshuis et al.[15] pointed that a limitation of Petri nets is that a transition does not necessarily fires at the time instant when a task is carried, it can be fired at an aleatory time after the execution. To solve this problem they present reactive Petri nets[15]. Other examples of extended Petri nets applied to workflow modeling is the one presented in [17] where Ha et al. use timed colored Petri nets (allowing differentiation among tokens and the inclusion of time restriction) to model dynamic product development processes; or the use of hierarchical Petri nets (HPN), where each place substitutes a lower level Petri net, presented by Boualem in [11] or by Alt et al. who use HPN to represent grid workflows[9].

In [6], Van der Aalst et al. propose a new workflow modeling language named YAWL (yet another workflow language). To create YAWL, the authors analyzed the different existing workflow modeling languages and studied which ones were able to represent the higher number of workflow patterns[31]. Van der Aalst et al. realized that the Petri net based languages were the ones which fitted best so they took them as a start point. In this way, the authors created a more intuitive workflow language with a formal mathematical notation and supporting almost all the existing workflow patterns. Besides at this stage YAWL only supports the control-flow perspective, it is being developed and improved in other important workflow aspects such as the data and the resource perspectives.

An alternative technique of modeling workflows is the use of workflow patterns (WP). Workflow patterns refer to recurrent specific problems and proven solutions to the development of workflow applications. It is estimated that using WP any workflow can be represented. As seen above, WP are also used to evaluate the performance of a workflow modeling language. The main research in this field has been done by Van der Aalst et al.; in [31] they present a classification and an analysis of the existing workflow patterns. In 2006 Russell et al. reviewed the list provided by Van der Aalst adding 20 more patterns which defined specific cases of the existing ones[32].

Regarding the resources representation in workflows, despite some languages such as the mentioned UML or BPEL provide tools for representing resources itself, many workflow modeling languages do not integrate resources into its representations so they need to be extended. A recent work on workflow representations that we should take into account is [23] 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.

3.2 Workflow Monitoring

Workflow monitoring concerns our work since to predict delays it is necessary to monitor the evolution of the workflows and to store information about the events generated by the workflow execution. In this section we comment some of the work developed in this area among the recent years and how it is related with this MSc thesis.

In [19] a web based monitoring system for distributed workflow operations is presented. Hong et al. developed a software for supervising the evolution of the workflow and to store statistical data. Nevertheless, the software does not include any tools for predicting the evolution of the workflow or for the early detection of possible delays and it is not based in any particular workflow language. In [22] Kanana et al. propose a monitoring system based on triggers, as the above mentioned article.

Regarding workflow monitoring in service architectures, Van der Aalst et al.[5] 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 [29] 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.[16] developed a micro controller-based process monitoring in order to control the

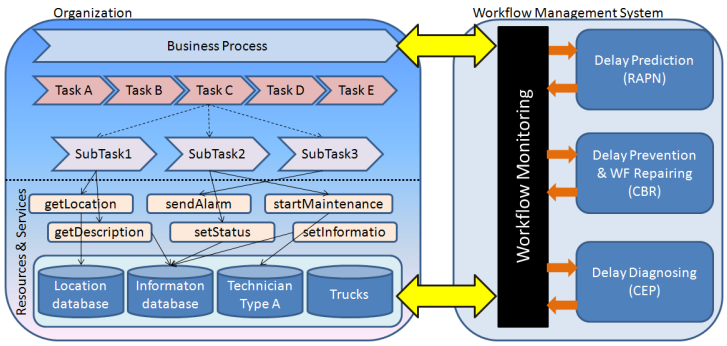

Fig. 3 Overview of the whole system.

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[2].

Other previous related works regarding to workflows monitoring come from the multi-agent community. For example, in [34] 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 [37] an agent based system is also proposed to deal with coordination and management of workflows between virtual enterprises.

4 Intelligent Workflow Management System

As Figure 3 shows, our proposal is to develop a workflow management system (WMS) which handles both the modeling and the monitoring of a business process and its resources. The WMS models the business process using high level Petri nets and monitors its development at the task level. This allows us to predict the possibility of delays in the development of the business process.

Before starting to describe our monitoring methodology we introduce the workflow management system (WMS) architecture.

The intelligent WMS uses Petri nets for workflow modeling, and takes into account all the resources available in the system. Workflows and resources are handled by the monitoring system (MS), as shown in Figure 4. The MS engine access to the following data:

- Library of workflow patterns, which contains the workflows modeling the business process activity.
- Resource data base, which contains the information related to the available resources of the system.
- Running workflows memory, which contains the workflows states current running in the system (workflow environment marking).

With this information, the WMS uses the following method to start workflows:

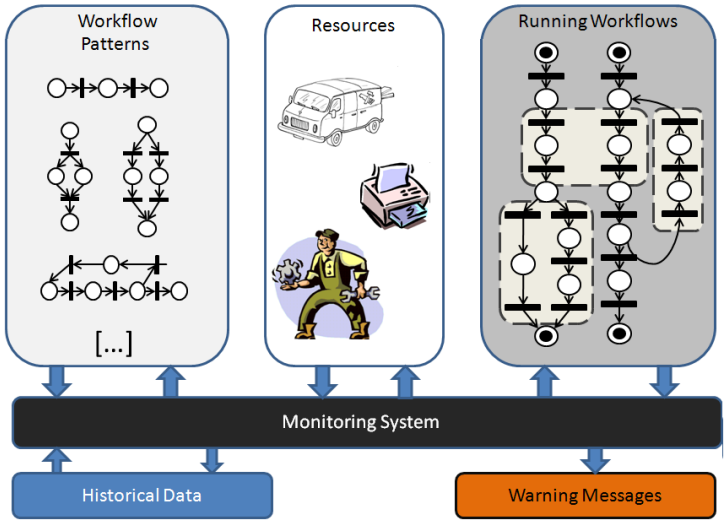

Fig. 4 The workflow management system is responsible for modeling and monitoring the workflow and sends warnings when a possible delay is detected.

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

Workflows are modeled with Petri nets, since they are a well known tool for workflow modeling and they offer a wide range of extensions to facilitate this task. Then, WMS is also responsible of firing the Petri nets transitions while the workflows are interacting and advancing, so the workflow can be monitored. Moreover, using previous cases (historical data in Figure 4), WMS estimates durations of activities so delays, lack of resources or deviations can be detected. When those are detected, MS sends warning messages so the workflow can be restructured in order to minimize the impact of these problems. WMS also stores workflow historical data so further data mining can be done.

The key issues are how workflows are modeled, so the available resources are taken into account in the monitoring phase. Particularly, we introduce a new Petri net extension that is detailed below.

4.1 Workflow Modeling: Resource Aware Petri Nets

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 Petri

nets with new elements so that *resources* can be represented. We called this extension *resource-aware Petri nets* (RAPN). Previously to the formulation of RAPN, some preliminary definitions are provided as follows.

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
- $\mathrm{P} \cap \mathrm{T} = \emptyset$
- A is the set of arcs which connect P with T and vice versa. $A : (P \times T) \cup (T \times P)$

When PN are used to model a Workflow, they are called Workflow Nets (WF net). In WF nets, the most common type of transition is the one which represents tasks (e.g. *wheel_assembly*), 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.

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, tokens (which are colored) represent instances. 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

RAPN (Definition 5) incorporate resources to high level Petri nets[7]. 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. $R = \{\langle r_1, Q_1 \rangle, ... \langle r_n, Q_n \rangle\}$ 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, $\mathrm{TS} = \langle t_s, t_e \rangle$.

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
- $\mathrm{P} \cap \mathrm{T} = \emptyset$
- 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

Each RAPN represents a workflow definition, so, a token in a RAPN represents a workflow instance. A workflow environment can be mapped as several RAPN which share the same resources.

4.2 Workflow Monitoring: Predicting Delays

The workflow management system (WMS) is responsible for monitoring the development of a workflow instance.WMS is aware of the information about all the tokens that are currently inside a workflow: current time, the instant time when the workflow instance started, the number of available resources, which cases are occupying the resources, etc. During the monitoring *workflow time out alarms* can be generated as well as two kind of warning can be send: *Task Delay*(Definition 9) and *Workflow Delay*(Definition 11). The first one advises that the time spent in the execution of a concrete task its exceeding the task mean time; this do not necessary behaves a workflow delay as the lost time can be recovered during the execution of the remaining tasks. *Workflow Delay* alarm is triggered when the workflow estimated duration exceeds its deadline. Monitoring at the workflow level means that it is necessary to compare the evolution of the monitored workflow instance with the standard deviation (SD). For that purpose, we kept attached to each WF its mean time execution and its SD, which is obtained statistically from past executions.

4.2.1 Workflow time out alarm

Every time a new instance is started, a maximum deadline for the instance resolution is assigned to it. Usually this deadline is a higher time value than the mean execution time for the workflow.

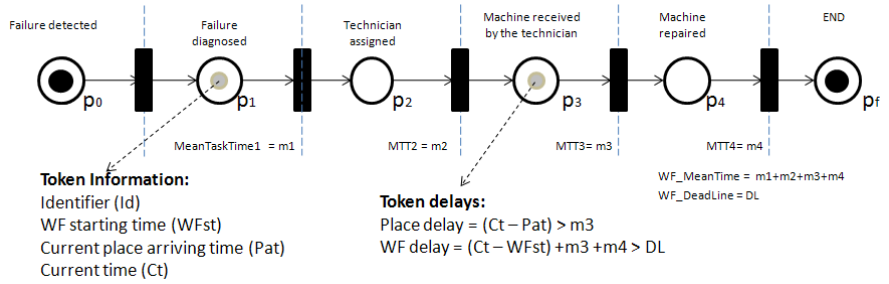

Fig. 5 Graphical representation about how the delays are estimated based on the token information, the transitions mean times and the workflow state.

When a workflow deadline is reached or close to be reached, the workflow management system sends a *time out* message or a *running out of time* alarm. 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. The exceeding of this deadline can cause important problems in the system. In service oriented architectures can imply the loss of messages, causing communication problems and even the restart of the process. In other 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. That is why it is important to predict possible delays the sooner the better so reescalation rules or modifications in the workflow can be done in order to avoid the delay or to minimize the impact of this retard. We consider than a lower-level monitoring of the workflow, in a task level, would result in an earlier detection of the delay.

4.2.2 Task delay warning

In our approach, besides the global execution time of the workflow, we monitor the time that tokens spend on each place. Moreover we endow tokens with information about the time it started the workflow, the instant it arrived to the current place and the current time stamp. This information allows us to detect possible delays in the workflow before it reaches its deadline (e.g. Figure 5) as we can notice when a task is exceeding it's normal execution time, sending *task delay* warning alarms.

4.2.3 Workflow delay warning

However, a delay in the execution of a task does not necessarily means a delay in the workflow execution as a faster execution of the rest of activities can avoid the global delay. In order to advance the workflow execution delay we use the information stored on the token and the time required to execute the worst case (the slowest path) of the pending workflow. If the sum of the spent

time in the previous tasks of the workflow with the mean time of the pending tasks is higher than the workflow deadline then a *workflow delay* alarm is triggered. Thus, the algorithm applied to predict workflow delays i s the one shown in Algorithm 1.

Algorithm 1 Delay detection algorithm

```
for each token TO_i do
  if ESD(TO_i) > μ_time(TO_i.getCurrentTransition) then
    TriggerTaskDelaywarning
  end if
  if EDW(TO_i) > wfμ_time(TO_i.WF) then
    TriggerWorkflowDelaywarning
  end if
end for
```

Definition 6 Transition mean time $\mu_{time}(t_i)$ is the mean of the time spent by tokens in t_i's input place before t_i is fired

Definition 7 The longest path of a workflow WF from a transition t_c is the slowest path to be executed starting from the current transition t_c to the endint transition t_e of the workflow. L(WF,t_c)=$\{t_c, t_{c+1}, ..., t_e\}$

Definition 8 The Elapsed Task Duration associated to a token TO_i is the difference between the arrival time of the token to a place p_j and the current time, $ETD(TO_i)$= $(Currenttime - arrivalTime(TO_i, p_j))$.

Definition 9 A Task Delay associated to a token TO_i, $TD(TO_i)$, occurs when a token has an elapsed task duration in a given place p that has exceeded the t_j output transition mean time, $ETD(TO_i) > \mu_{time}(t_j)$

Definition 10 The estimated duration of a workflow instance represented by a token TO_i, $EDW(TO_i)$, is the current elapsed duration, plus the addition of the transition mean time which belongs to the longest path, that is, $EDW(TO_i) = (Currenttime - Wokflowstartingtime) + \sum_{t_i \in L(WF,t_c)} \mu_{time}(t_i)$

Definition 11 A workflow instance represented by a token TO_i has a delay, $WD(TO_i)$, if its estimated duration exceeds the workflow mean time, $WD(TO_i) = EDW(TO_i) > wf\mu_{time}(WF)$.

5 Experiments and Results

This section presents the results obtained in the research concerning this article. Firstly a brief description of the implemented prototype is described, then, some of the performed experiments are described and, finally, the experimentation results are presented.

5.1 Prototype

The first prototype of our work consists in 3 modules: a workflow framework, a workflow simulation engine and the workflow management system (Figure 6). The workflow framework can load Petri nets defined by XML or load Petri nets designed with the PM Editeur[38] 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 the evolution of a workflow, given a set of parameters (the workflow modeling, the probability of a workflow 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. All this software, except the PM Editeur, have been developed using Java technology.

5.2 Experimental Set Up

To test the performance of our system we modeled and simulated a set of workflows corresponding to a medical device reparation service from an hospital. They were extracted and adapted from the AIMES project [8].

5.2.1 Workflows

The workflows correspond to common activities in the medical device maintenance industry such as assigning a technician for a device repairing, reassigning a technician, locally repair a device, etc. They are the following:

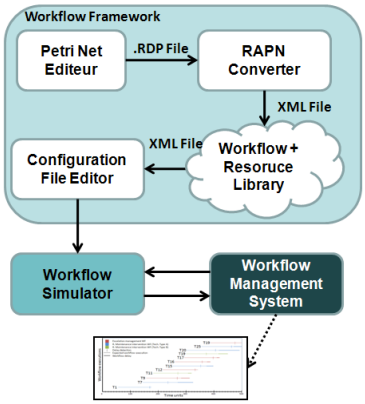

Fig. 6 Architecture of the workflow simulation framework

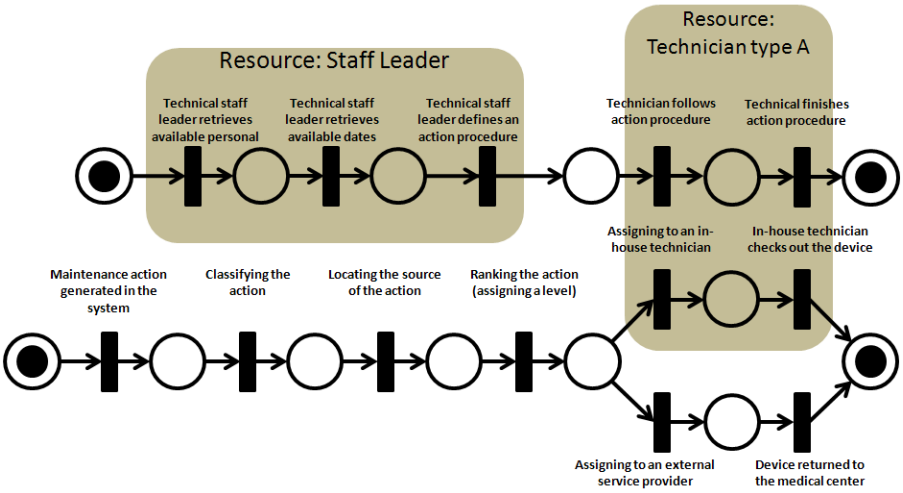

Fig. 7 Top: Maintenance event escalation management. Bottom: Reactive maintenance intervention.

- **Reactive Maintenance Interventions - RMI** (Figure 7 bottom): describes the procedure to follow when a 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 service and assigns the maintenance action to a technician. Finally the technician carries the action and the workflow finalizes. This workflow is composed by six services. In this case, the resources needed to accomplish the workflow are technicians of a concrete type.
- **Maintenance Event Escalation Management - MEEM** (Figure 7 top): 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 services). 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.
- **Inventory and Installation of New Equipment - IINE** (Figure **??**): It 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 popper 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 services are

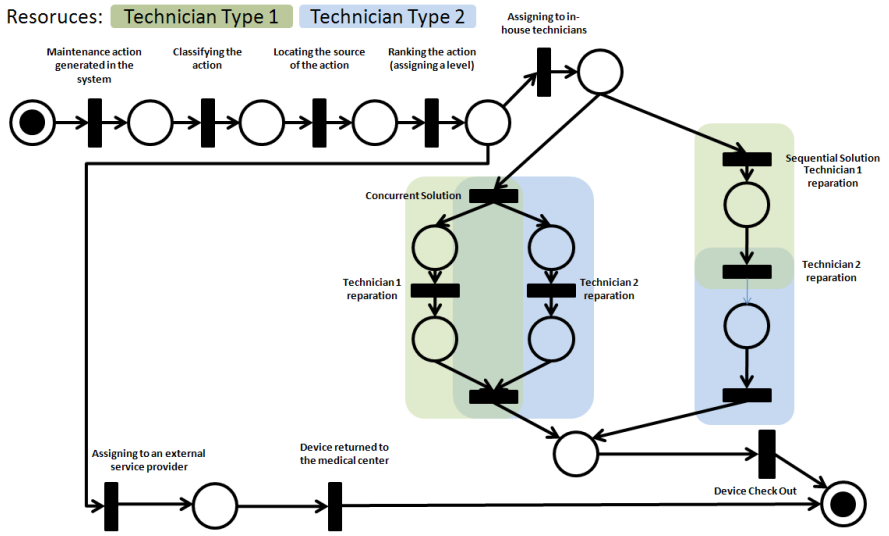

Fig. 8 Multiple Reactive Maintenance Intervention.

required for this workflow and three different kind of resources (although only two will be used at each instantiation).

– **Multiple Reactive Maintenance Intervention - MRMI** (Figure 8: This workflow is an extension of the *Reactive Maintenance Intervention* (RMI) workflow presented before. As the previous one, this business process is started when a medical device throws a maintenance warning but is followed when the maintenance must be carried out by two kinds of technician. The tasks to follow are almost the same than the RMI but differ in the reparation task. In this workflow an exclusive or selection is done and it must be decided if the two technicians can work concurrently or if they must act sequentially. The resources needed in this process are two different kinds of technicians.

5.2.2 Scenarios

Combining the workflows presented in the previous paragraphs, we created 5 different scenarios to test the delay prediction procedure:

– **Scenario 1** There are two kind of resources in the organization: *technical staff leader*(1 in the system) and *technician type A*(4 in the system), and two different workflows:*reactive maintenance interventions* using a type A technician and *maintenance event escalation management* using a type A technician and a technical staff leader. The resource type A technician is shared by both workflows. The scenario simulated among 500 time units with a a workflow starting probability $p = 0.05$ per time unit; the kind of workflow started is randomly chosen with the same probability for each

workflow type. This scenario allows us to study the behavior of our methods in a simple experiment.

- **Scenario 2** There are three kind of resources in the organization: 1 *technical staff leader*, 3 *technician type A* and 1 *technician type B*. This scenario uses two kind of workflows: *reactive maintenance interventions* and *maintenance event escalation management*. However, this time, some of the reactive maintenance interventions must be carried by a type B technician so the *reactive maintenance interventions* have two kind of instantiations, one using a type A technician and one using a type B. The scenario simulated among 500 time units with a a workflow starting probability $p = 0.05$ per time unit; the probability of starting a *reactive maintenance intervention* using a type B technician is $p = 0.2$ while the probability of starting one of the other workflows is $p = 0.4$ for each one. The aim of this experiment is to complicate the scenario 1 in order to study the performance of the workflow management system in a more complex scenario.
- **Scenario 3** This scenario adds the *inventory and installation of new equipment* business process to the first scenario. We considered that the type D technician as the same which appears in the *Reactive maintenance intervention* and in the *maintenance event escalation management* . 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 (used by all the workflows), 1 staff leader (used by MEEM) and 2 type I and T technicians (used by IINE). We simulated a 500 time units period with a probability of instantiating a workflow $p = 0.05$ distributed in 3/6 for RMI, 2/6 for IINE and 1/6 for MEEM. Scenario 3 allows us to study the workflow management system when many resources are involved in the business processes.
- **Scenario 4** In this case only the *multiple reactive maintenance intervention* is used since we considerer that is an enough complex workflow itself. Moreover, MRMI permits to study the difference between the sequential and the concurrent use of resources. The simulation has been done for 500 time units with a probability of starting a new workflow of $p = 0.2$. In the simulation three resources of each kind has been defined.
- **Scenario 5** This scenario do not correspond to any of the business process presented before since it is a completely synthetic instance created to analyze the behavior of the system when many workflows share the same resource. Figure **??** describes the workflow environment where three sequential workflows share the same resource (*Resource 3*) moreover two of them share *resource 2* and one also uses the *resource 1*. The amount of resources available in the system for each workflow is directly proportional to the number of workflows which use them: six type 3 resources, four type 2 resources and two type 1 resources. As the rest of scenarios, we have simulated 1000 time units, the probability of instantiating a new worfklow used is $p = 0.01$ with the same probability for each workflow type.

5.3 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 9 shows the results obtained in the first experiment where two different workflows are sharing two different resources. The simulation among 500 time units generated 30 workflow instances 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 10 shows the results of the second scenario where the same two workflows share 3 different types of resources with a different quantity of them. The simulation generated 31 workflow instances where 10 finished out of time (T7, T9, T12, T15, T16, T17, T19, T20, 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 in Figure 11 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.

6 Discussion

The obtained results in the different scenarios show that our prototype can provide an early detection of workflow delays. In some instances 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 main-

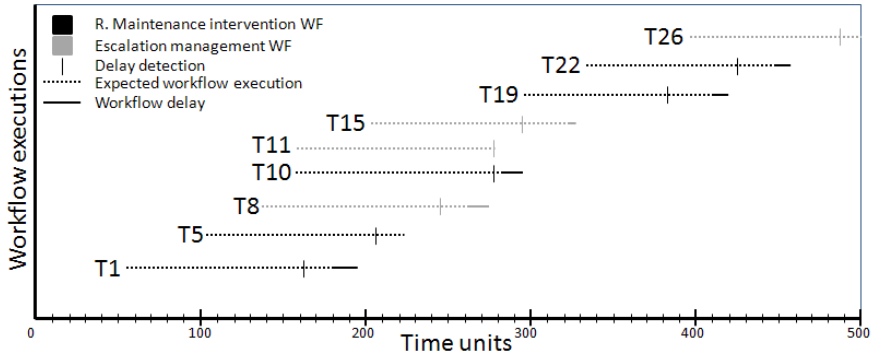

Fig. 9 Result of the first scenario where the system resources are 4 type A technicians and a 1 technician staff leader.

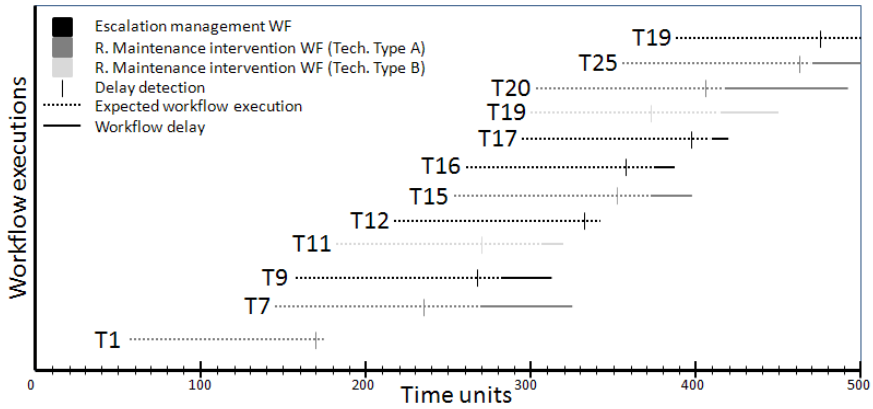

Fig. 10 Result of the second scenario where the system resources are 3 type A technicians, 1 type B technician and a 1 technician staff leader.

tenance 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 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.

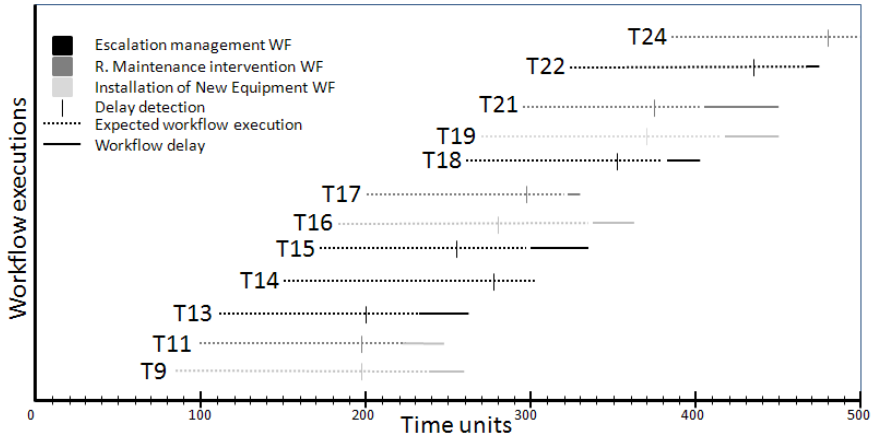

Fig. 11 Result of the third scenario where the system IINE is added to the workflow environment.

Table 1 shows the confusion matrix of the three experiments presented above and Table 6 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 instance 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

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.

Delayed Scenario	Monitoring Agents used	Delays produced	Delays detected (TP)	Delays not detected (FN)	Erroneous Delays d
SC. 1	30	7(100,00%)	0(0,00%)	2(8,70%)	
SC. 2	31	10(100,00%)	0(0,00%)	2(9,50%)	
SC. 3	26	10(100,00%)	0(0,00%)	2(12,50%)	
SC. 4	97	3(100,00%)	0(0,00%)	2(2,10%)	
SC. 5	52	10(100,00%)	0(0,00%)	3(7,10%)	
Mean	30	8(100,00%)	0(0,00%)	7,98%	

Table 2 Summary of the different scenarios executions

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.

7 Conclusions

Our work aimed to include a new perspective that can improve workflow efficiency by taking under consideration not only the time (as it has been done until now) but also the resources availability and the concurrent execution of workflows inside an organization. Actual tools for modeling workflows are not enough to represent all the variables that can affect their efficiency, as the available resources.

This work has faced two 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 defined the resource-aware Petri nets (RAPN), a Petri net extension which includes the information of the resources needed by each task. 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 its released when the last transition which needs the resource is fired.

Once the business processes are modeled, workflow management systems are in charge of its monitorization. The monitoring of RAPN in a task level, taking into account the organization resources and the rest of workflows which are executed in a workflow environment, allows to predict delays in a workflow execution and to generate its correspondent alarms. 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 tasks and to detect which are the weaker points of the architecture. In this sense, after delays are detected, other tools as the one presented in [**?**] can complement this work with diagnostic facilities.

To test our approach, we simulated a medical equipment maintenance organization deployed in a service oriented architecture. The simulations we ran fulfilled our expectations, indicating that an anticipated delay alarm can be predicted in many different situations. There were also some instances (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.

At the end, 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 an organization or company, however it can present difficulties on its deployment in workflows involving external partners since the workflow management system cannot trace all the resources implicated on the business process of different companies.

For that purpose, as a future work, we are considering to incorporate Multi Agent Systems technology to the WMS that could be more suitable to distributed environments. Another point is the incorporation of resource allocation algorithms in order to endow the WMS with more capabilities to avoid delays and to coordinate manufacturing works. Moreover, the incorporation of auction mechanisms to the WMS could reduce economic costs during the resouce allocation. The combination of this points will allow the WMS to deal with external services and providers in a better way.

References

1. Bpel project. http://www.eclipse.org/bpel/ *Accessed June 10th, 2010.* URL http://www.eclipse.org/bpel/
2. van der Aalst, W.: Interval timed coloured petri nets and their analysis (1993)
3. van der Aalst, W.M.P.: The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers **8**(1), 21–66 (1998). URL http://wwwis.win.tue.nl/~wsinwa/Publications/p53.PDF
4. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. Data Knowl. Eng. **47**(2), 237–267 (2003). DOI 10.1016/S0169-023X(03)00066-1. URL http://dx.doi.org/10.1016/S0169-023X(03)00066-1
5. van der Aalst, W.M.P., Pesic, M.: Specifying, discovering, and monitoring service flows making web services process-aware. BPM Center Report BPM-06-09, BPM Center (2006)
6. van der Aalst, W.M.P., Ter: Yawl: yet another workflow language. Information Systems **30**(4), 245–275 (2005). DOI 10.1016/j.is.2004.02.002. URL http://dx.doi.org/10.1016/j.is.2004.02.002
7. Abdulla, P.A., Mahata, P., Mayr, R.: Dense-timed petri nets: Checking zenoness, token liveness and boundedness. CoRR **abs/cs/0611048** (2006)
8. AIMESproject: Deliverable 1.3: requirements specification (2008-2010). URL http://www.aimes-project.eu/

9. Alt, M., Gorlatch, S., Hoheisel, A., Pohl, H.W.: Using high-level petri nets for hierarchical grid workflows. In: E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, p. 13. IEEE Computer Society, Washington, DC, USA (2006). DOI http://dx.doi.org/10.1109/E-SCIENCE.2006.149
10. Bastos, R., Dubugras, D., Ruiz, A.: Extending uml activity diagram for workflow modeling in production systems. In: HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9, p. 291. IEEE Computer Society, Washington, DC, USA (2002)
11. Benatallah, B., Chrz;stowski-Wachtel, P., Hamadi, R., O?Dell, M., Susanto, A.: Hiword: A petri net-based hierarchical workflow designer. Application of Concurrency to System Design, International Conference on **0**, 235 (2003). DOI http://doi.ieeecomputersociety.org/10.1109/CSD.2003.1207720
12. Brogi, A., Popescu, R., Brogi, A., Popescu, R.: Bpel2yawl: Translating bpel processes into yawl workflows (2006)
13. Buscemi, M., Sassone, V.: High-level petri nets as type theories in the join calculus. In: In Proceedings of 4th FOSSACS, volume 2030 of LNCS, pp. 104–120. Springer (2001)
14. Dumas, M., ter Hofstede, A.H.: Uml activity diagrams as a workflow specification language. pp. 76–90. Springer Verlag (2001)
15. Eshuis, R., Dehnert, J.: Reactive petri nets for workflow modeling. In: Application and Theory of Petri Nets 2003, pp. 296–315. Springer (2003)
16. Frankowiak, M.R., Grosvenor, R.I., Prickett, P.W.: Microcontroller-based process monitoring using petri-nets. EURASIP J. Embedded Syst. **2009**, 1–12 (2009). DOI http://dx.doi.org/10.1155/2009/282708
17. Ha, S., Suh, H.W.: A timed colored petri nets modeling for dynamic workflow in product development process. Comput. Ind. **59**(2-3), 193–209 (2008). DOI http://dx.doi.org/10.1016/j.compind.2007.06.016
18. Hinz, S., Schmidt, K., Stahl, C.: Transforming bpel to petri nets. In: W.M.P. van der Aalst, B. Benatallah, F. Casati, F. Curbera (eds.) Proceedings of the 3rd Int'l Conference on Business Process Management (BPM 2005), pp. 220–235. Springer Verlag, Nancy, France (2005). DOI http://dx.doi.org/10.1007/11538394_15. URL http://dx.doi.org/10.1007/11538394_15
19. Hong, H.S., Lee, B.S., Kim, K.H., Paik, S.K.: A web-based transactional workflow monitoring system. In: WISE '00: Proceedings of the First International Conference on Web Information Systems Engineering (WISE'00)-Volume 1, p. 166. IEEE Computer Society, Washington, DC, USA (2000)
20. Indulska, J.R.M.: How good is bpmn really? insights from theory and practice. In: 14th European Conference on Information Systems (2006)
21. Kalnins, A., Vitolins, V.: Use of uml and model transformations for workflow process definitions. CoRR **abs/cs/0607044** (2006)
22. Kanana, E., Farhi, M.: Monitoring information and data flows using triggers in a dynamic workflow environment. In: Proceedings of the 5th European Conference on Knowledge Managemen, pp. 175–179 (2004)
23. Lombardi, M., Milano, M.: Allocation and scheduling of conditional task graphs. Artificial Intelligence **174**(7-8), 500–529 (2010)
24. Muehlen, M., Recker, J.: How much language is enough? theoretical and practical use of the business process modeling notation. Advanced Information Systems Engineering pp. 465–479 (2008). URL http://dx.doi.org/10.1007/978-3-540-69534-9_35
25. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (2002). DOI 10.1109/5.24143. URL http://dx.doi.org/10.1109/5.24143
26. Pant, K.: Business Process Driven SOA using BPMN and BPEL: From Business Process Modeling to Orchestration and Service Oriented Architecture. Packt Publishing (2008). URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20/path=ASIN/1847191460
27. Petri, C.A.: Kommunikation mit automaten. Ph.D. thesis, Institut für instrumentelle Mathematik, Bonn (1962)
28. Rinderle, S., Reichert, M., Dadam, P.: Correctness criteria for dynamic changes in workflow systems: a survey. Data Knowl. Eng. **50**(1), 9–34 (2004). DOI http://dx.doi.org/10.1016/j.datak.2004.01.002

29. Rozinat, A., Wynn, M., van der Aalsta, W., ter Hofstede, A., Fidge, C.: Workflow simulation for operational decision support. Data and Knowledge Engineering **68**(9), 834–850 (2009)
30. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual, 2. edn. Addison-Wesley, Boston, MA (2005)
31. Russell, N., Arthur, van der Aalst, W.M.P., Mulyar, N.: Workflow control-flow patterns: A revised view. Tech. rep., BPMcenter.org (2006)
32. Russell, N., Arthur, van der Aalst, W.M.P., Mulyar, N.: Workflow control-flow patterns: A revised view. Tech. rep., BPMcenter.org (2006)
33. Tick, J.: Workflow model representation concepts. Nemzetkzi Szimpziuma 7 th International Symposium of Hungarian Researchers on Computational Intelligence Workflow Model Representation Concepts **7** (2002)
34. Wang, M., Wang, H.: Intelligent agent supported flexible workflow monitoring system. In: A. Banks Pidduck et al: CAISE 2002, LNCS 2348, pp. 787–791 (2002)
35. Wassermann, B., Emmerich, W., Butchart, B., Cameron, N., Chen, L., Patel, J.: Sedna: A bpel-based environment for visual scientific workflow modelling. In: In Workflows for eScience - Scientific Workflows for Grids. Springer Verlag (2007)
36. Wirtz, G., Weske, M., Giese, H.: Extending uml with workflow modeling capabilities. In: CooplS '02: Proceedings of the 7th International Conference on Cooperative Information Systems, pp. 30–41. Springer-Verlag, London, UK (2000)
37. Zarour, N., Boufaida, M., Seinturier, L., Estraillier, P.: Supporting virtual enterprise systems using agent coordination. Knowledge and Information Systems **8**, 330?349 (2005)
38. Zdenk, M.S., Svdov, M., Hanzlek, Z.: Matlab toolbox for petri nets. In: 22nd International Conference ICATPN 2001, pp. 32–36 (2001)