UNIVERSITAT DE GIRONA
ESCOLA POLITÈCNICA SUPERIOR
Department of Electrical, Electronics and Automatics Engineering

Màster Universitari en Informàtica Industrial, Automàtica,
Computació i Sistemes
(MIIACS)

Predicting and Diagnosing Delays in a Workflow Environment

Submitted by
Albert Plà Planas
apla@eia.udg.edu

Supervisor:
Dra. Beatriz López

July 9, 2010

Acknowledgments

First of all, I would thank Dr. Beatriz López, my supervisor in this Master Thesis for her ideas and support. I also would greet the members of the eXiT research group for all their help and advises, specially to Pablo, with whom I shared too long working sessions.

To my friends and colleagues Marià, Masi, Santfeliu, Ricard, Henrik and Albert. A coffee break is always necessary.

To the Wednesday night cooking crew. In this way the week is much shorter!

To my flat mates. Iotatranquils!

To the Besalú people, there's always something to chat and laugh about.

To Pep Guardiola and his team. $2 - 6$ will never be forgotten

And finally I would thank my parents and family, with them its all easier.

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-C02-0

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 thesis a new Petri net extension for modeling together workflow activities with the resources needed by the represented process is presented: resource-aware Petri nets. Moreover a workflow management system for process monitoring, delay prediction, diagnosing and repairing 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. When a delay is predicted a case-based reasoning tool proposes preventive actions in order to avoid or minimize the delay impact. Finally, when a delay cannot be prevented, a complex event processing system analyzes and diagnoses the delay causes.

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

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

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. 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 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 [12], 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.

When a delay is detected it is important to act quickly in order to minimize the delay impact to the whole system and to the workflow itself. The use of escalation and refactoring rules, provided by expert technicians, can help to soft the impact of a delay and can be a good solution when there is not many previous information about the system but there are some issues about how to proceed: how are refactoring rules affected by the resource availability? and by the number of current ongoing workflows? which rule must be followed when more than one seems suitable? The use of knowledge based techniques can help to minimize the delay propagation among the system. In this thesis a complex event processing system is applied for dealing with delay detection and diagnosis. However, this diagnostic work is not the main focus of the thesis; it comes from the collaboration of other researches at the *eXiT* research group. The focus of this work is dealing into prediction strategies.

When a delay is predicted it is important to act quickly too. In this case repairing strategies that take into account the current environment situation are needed. However, the number of factors involved in the decision to be made is huge, depends on the kind of ongoing workflows, their current execution state and the status of the available resources (overloaded, bro-

ken, etc.). The storage of historical data and the use of knowledge based reasoning techniques can provide a way to tackle with the reparation strategies. In this thesis a case-based reasoning tool is explored as way to exploit historical data to repair workflows when delays are predicted.

The scope of this thesis 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 can also happen due to concurrent execution of multiple workflows.

1.2 The Problem

There is a need of developing methods to deal with concurrent workflows in a given environment that share resources.

The hypothesis to be studied in this work is that Petri net extensions as workflow modeling technique together with case-base reasoning which use historical workflow data can be used to detect and predict delays.

1.3 Objectives and Methodology

The main goal of this thesis is to define a method to model simultaneously the business process workflows and the resources needed by them, and to develop a technique to monitor them in order to predict and repair delays. As this is not a trivial task, we divided the objective in simpler subgoals:

- Realize a study about the current state of the art concerning workflow

modeling, workflow monitoring and workflow mining.

- Define a method for modeling workflows together with the resources available in the system.
- Propose a technique to anticipate workflow delays.
- Monitor simultaneously all the system workflows so delays and abnormal behaviors can be detected and predicted taking into account their interaction and the shared resources.
- Propose a technique to repair workflows so predicted delays can be avoided or its impact minimized.
- Test the monitoring system in a simulated environment in order to validate its performance.
- Use a technique for refactoring the existing workflows when a delay is detected in order to reduce the effect of it in the whole system.

As a research work, this thesis follows the scientific methodology generate and test[31]. Regarding the generation step, since there are a myriad of methods to consider, the thesis is focused on the following methodologies:

- Petri nets for workflow modeling, monitoring and delay prediction.
- Case based reasoning for workflow repair.

Regarding the test step, the methodology followed to realize the work is the typical iterative project life-cycle(Figure 1.1) which consists in four different steps: identification, design, implementation and evaluation. The identification stage is focused on the comprehension and identification of the constraints and factors which involve the problem and on the direction that the project must take. The design step defines the actions and the procedures that will take place in the project. The implementation stage, as its name points, consists in carrying the actions defined in the previous phase. Finally the evaluation phase is focused on testing the obtained product and detecting which problems and issues are still not solved and which aspects can be improved. The cycle is repeated until the evaluation obtained is completely satisfactory.

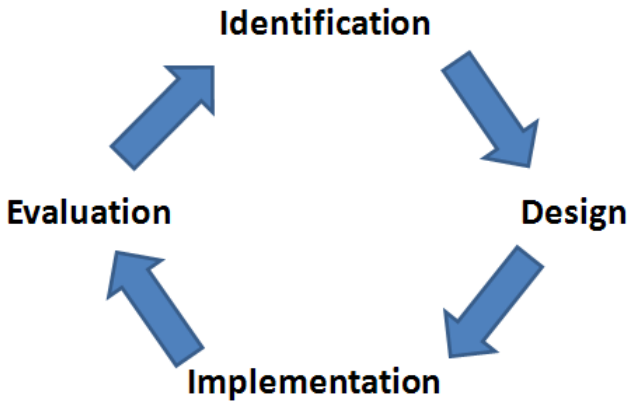

Figure 1.1: Project life-cycle schema.

1.4 Thesis Outline

The master thesis is structured in the following chapters:

- **Chapter 2 - Background:** The basic concepts needed to understand the work presented on this thesis are provided in this chapter. First we provide a brief workflow vocabulary so the reader can be familiarized with the concepts discussed on the rest of chapters; then a the basic Petri net notation and terminology is introduced in order to facilitate the comprehension of the monitoring and modeling methods presented; finally the fundamentals of case-based reasoning are shown.

- **Chapter 3 - State of the art:** This chapter presents different works and papers concerning the research exposed in this master thesis. It is divided in four different sections. The first one, workflow modeling, shows different methods which can be used to model workflow activities and applications where they have been used. The second section work related to workflow monitoring is exposed. The third section exposes research about discovering unknown workflows given a set of data while the fourth section concerns the application of data mining techniques to known workflows in order to adapt or optimize them.

- **Chapter 4 - Methodology:** This chapter explains the solutions we adopted to solve the problem presented in the first chapter. Firstly

we propose an extension for the Petri nets in order to incorporate the available resources in a system to its notation: *resource aware Petri nets*. The following section explains how to monitor a system using resource aware Petri nets and proposes a method for detecting and predicting delays in workflow executions. The use of case based reasoning for repairing workflows is proposed to avoid and reduce the predicted delay impacts.

- **Chapter 5 - Experimentation and results:** Here the results of the master thesis are presented simulating workflows corresponding to common activities in the medical device maintenance industry. Firstly some examples of process modeling are shown and then the concurrent execution of these workflows is simulated in different scenarios, showing how the monitoring and the delay detection is done.

- **Chapter 6 - Conclusions and further work:** The last chapter of the thesis states the conclusions obtained from the research. Moreover a proposal to continue this work as a PhD is formulated.

Chapter 2

Background

This chapter describes the basic concepts required for understanding the work presented in this thesis. Firstly, some terminology regarding workflows is introduced, then Petri nets are accurately described and some of the most common Petri net extensions are commented, finally, we briefly describe the fundamentals of case-based reasoning and its different steps.

2.1 Workflow Concepts

In this section some workflow concepts which are treated along the document are defined in an informal way. The goal of this section is to acquaint the reader with the workflow nomenclature so the thesis can be easily understood.

- **Business process:** A business process is a set of related tasks which must be carried out to produce a particular service or product. It can be visualized as a diagram composed by a sequence of activities. E.g. Figure 2.1a shows the business process of a device reparation.
- **Task:** A task is a concrete activity with an identifiable start and end which must be accomplished within a defined period of time. A task can be composed by other tasks. In Figure 2.1*Device diagnosis* or *device reparation* are examples of tasks.
- **Workflow:** A workflow consists in a graph of interconnected actions which represents the tasks and interactions to be realized by a mech-

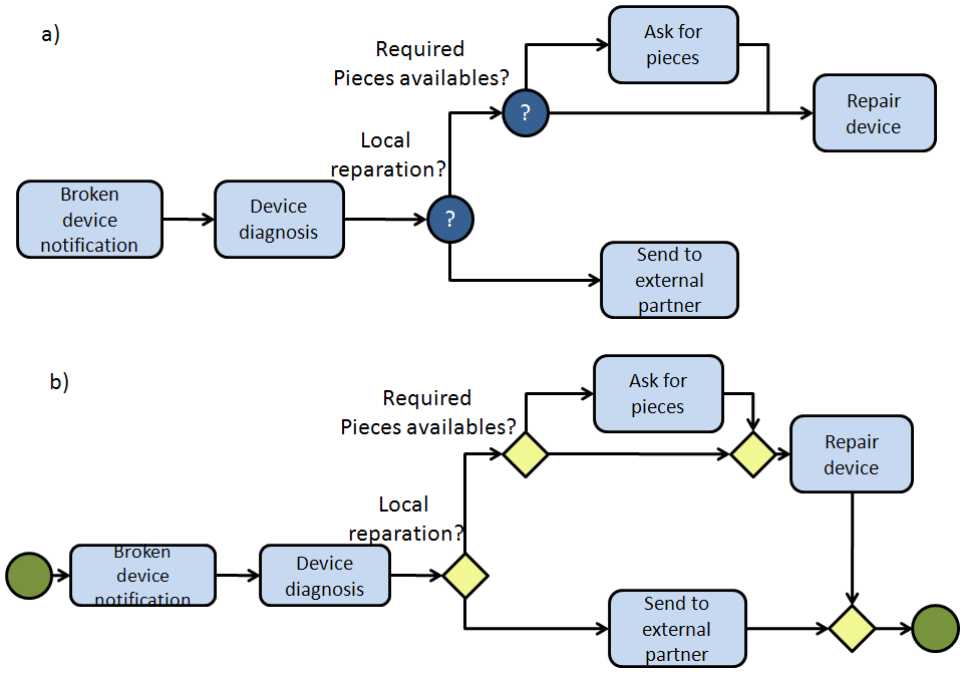

Figure 2.1: a)Business process of device reparation. b)Workflow modeling the device reparation business process using UML.

anism, a person, a staff, an organization, etc. Workflows can model not only business process but also exchange of messages, software procedures or information flows. E.g. Figure 2.1b shows the workflow which models the device reparation business process.

- **Workflow language:** A workflow language is a visual or grammatical representation used to define a business process or a workflow. Examples of workflow languages are *business processes management notation* (BPMn) of flow charts.

- **Workflow pattern:** A workflow pattern is a workflow which define specifically recurrent problems and provides a solution to the development of workflow designs (e.g. iterations, concurrent task executions or choices which are represented by rhombus in Figure 2.1b)

- **Workflow instance:** A workflow instance is a workflow which is being

executed in a concrete time instant. Every time a workflow starts, a workflow instance is created following a given pattern.

- **Workflow marking:** A workflow marking is the status (the step which is being executed, the data generated, the resources used, etc.) of a workflow instance in a concrete time instant.

- **Workflow environment:** A workflow environment is a set of workflows which coexists in a common framework (e.g. an organization, an industry, a server, etc.). They can share resources, actors or information.

- **Workflow environment marking:** A workflow environment marking is the status of a workflow environment. It stores the different workflow markings plus the resource information, the priorities, the workflows which are waiting for resource to be released, etc.

- **Workflow management system (WMS):** The workflow management system manages and monitors the different tasks which take place inside an organization or a workflow environment. Is responsible for monitoring the status of the different workflow instantiations and to store in a log the different events related to the workflow developement.

2.2 Petri Nets

A Petri net is a mathematical modeling language for describing action flows. Like other standards such as UML or BPMN Petri nets offers a graphical notation for stepwise processes and can represent actions such as choices, iterations, splits, joins, concurrent execution, etc. However, unlike almost all the industry standards, Petri nets have a mathematical definition for their execution and representation with a well-foundamentend mathematical theory. Moreover, there exists different variants and modification of Petri nets which facilitate the task of representing workflows and can be simplified to the mathematical Petri nets notation.

A Petri Net (PN) is a particular vision of a bipartite graph. A PN is composed by directed arcs, tokens and two kind of nodes: transitions or bars and places or circles. It is defined 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*
- $P \cap T = \emptyset$
- A *is the set of arcs which connect* P *with* T *and viceversa* $A : (P \times T) \cup (T \times P)$

In Petri nets directed arcs always connect transitions with places or vice versa but never connect to nodes of the same kind. A token is an entity which is traveling along a Petri net following the direct arcs and can represent a workflow execution state. Places store the tokens until they advance to a new node; in workflow terms places would represent an action which is being developed or a waiting interval. Transitions are responsible to make tokens advance; when a transition is enabled, if the previous place contains a token, consumes the token and places the token at the end of the output arcs. This action is known as firing and the equivalence in workflow semantics would be to start an action. For a more elaborate introduction to Petri nets see [35, 16].

The usage of Petri nets for workflow representation has been widely studied by authors like Russell[52], Van der Aaalst[65], Masuthe[40], Sheng[55] or Tick[60]. As mentioned above, in order to include different domain particularities such as time, labels or priorities, Petri nets where enriched with extensions which represented theses different domain particularities. This new kind of nets were called *high level Petri nets* (Figure 2.2).

2.2.1 Colored Petri Nets

In the basic 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 basic representation, the only way to discern between both tokens/process is to duplicate the Petri net and to put each token to the different nets, which presents a problem for real

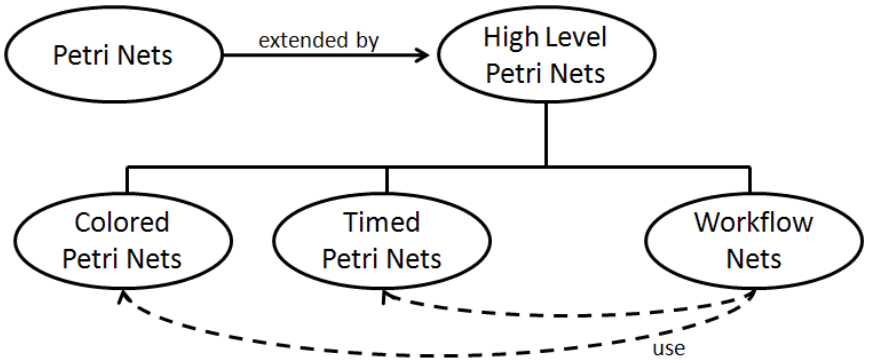

Figure 2.2: High level Petri nets extend the basic Petri net notation for including new features such as token differentiation or time.

process modeling since 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[61]. Colored Petri nets assign a type or an identifier to each token so the confusion between tokens disappears.

2.2.2 Timed Petri Nets

Another common extension for the basic Petri nets is the inclusion of time which can be handled 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 before its transition fires. Finally, *token-timed* PN (or *dense-timed* PN)[3] 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 work, when timed Petri nets are mentioned is referring to this token-timed PN extension.

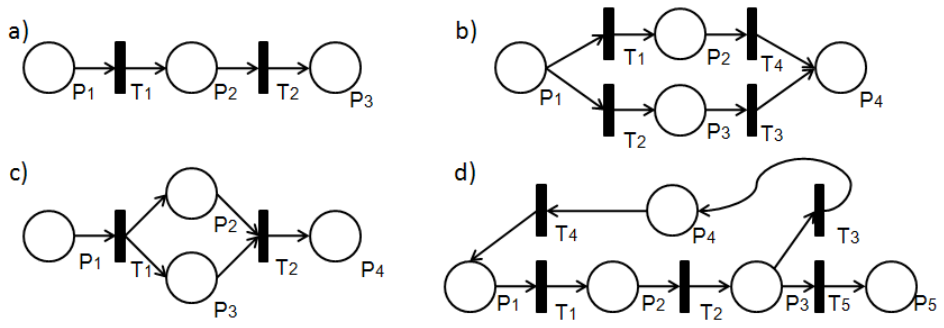

Figure 2.3: a)Petri net routing sequence. b)Petri net choice. c)Petri net parallel execution. d)Petri net iteration.

2.2.3 Workflow Nets

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. As Figure 2.3 shows, Petri nets can represent typical workflow behaviors like sequential routings, parallel executions, choices, iterations, etc.

When using Petri nets to model workflows, we talk about workflow nets (WN). In WN places represent states or conditions, transitions represent tasks, services, decisions or events and tokens correspond to cases (workflow instances). The main restrictions for modeling a workflow with WN are the following: a workflow net must have 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

In a workflow, the most common type of transition is the one which represents tasks (e.g. *send_message* in Figure 2.4 right), it is fired just at the moment when the task starts. When transitions represent the making of a decision they do not start any new task, they just chose if a path must

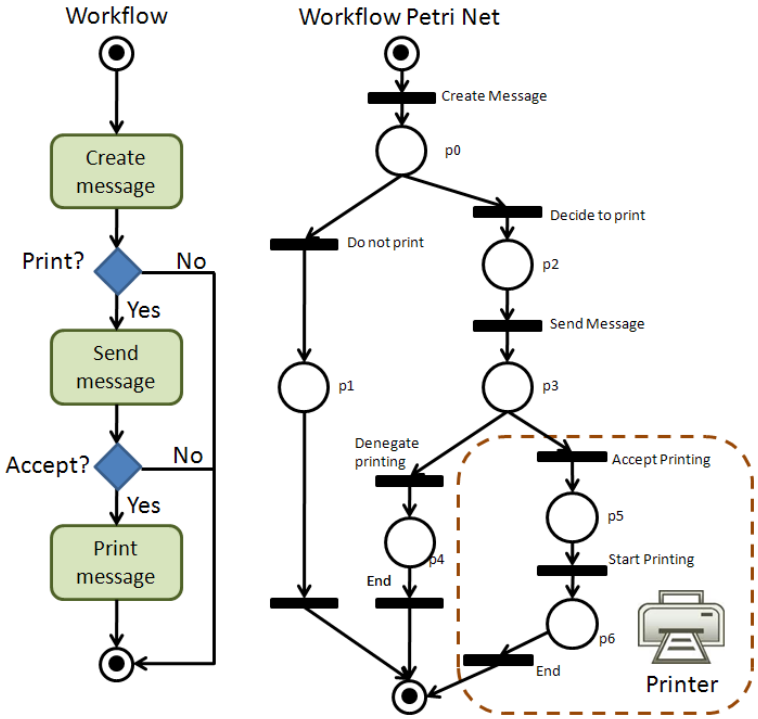

Figure 2.4: Transformation of a common workflow representation (left) to a workflow Petri net (right).

be followed or not and they are fired by the system (or by the actor which takes the decision). An example of this kind of transitions in Figure 2.4 is *accept_printing* and another one is *denegate_printing*. External events (e.g. user inserts a coin into the 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 task type one; decision type transitions never come alone, there must be at least to 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

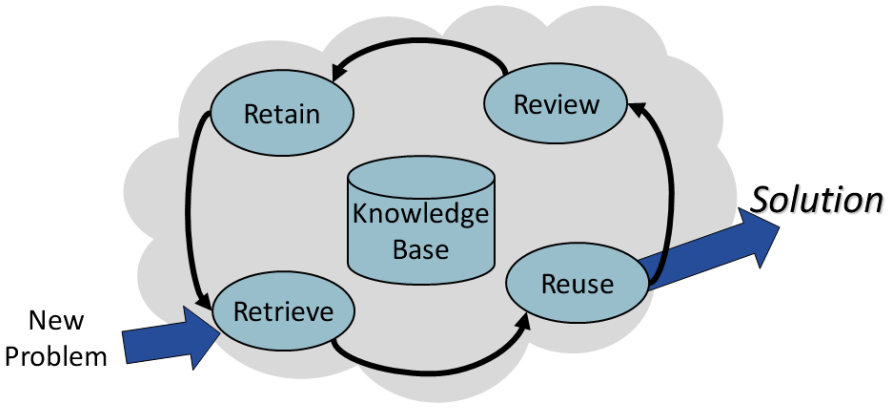

Figure 2.5: Case-based reasoning schema.

pre-condition of its input transition and a place p is the post-condition of its output transition. E.g. in Figure 2.4 place p_3 indicates than a message has been send but it still have not been printed.

Finally, in WF nets, tokens (which are colored) represent cases. Every time a token appears in the input place i means that a bussines new process bp has started inside the workflow. Tokens are colored so the processes can be distinguished between them. In some workflow nets the time is included as dense-time Petri nets so the workflow can be represented more accurately.

2.3 Case-Based Reasoning

Case Based Reasoning (CBR) is a knowledge-based technique which allows classifying or predicting a subject based on past experiences[2]. Reasoning by reusing past experiences is a powerful and common way to solve problems by humans. When facing a problem, humans tend to search for similar past situations and to adapt the used solution to solve the current issue they are dealing with. Case-based reasoning tries to transfer this behavior to artificial intelligence. CBR seeks similar cases to the current one and analyzes which decision or classification was taken in order to reuse it in the present solution.

All CBR shares a set of common tasks: identify the problem it is dealing with, find a similar past case, use the old case to propose a solution to the new one, evaluate the suggested solution and updating the knowledge

base with the new experience. As Figure 2.5 shows, CBR is divided in four different stages which are repeated for every new case: Retrieve, Reuse, Revise and Retain. First step, retrieve, searches for similar past situations to the new one; Reuse stage uses the retrieved situations to propose a suitable classification or solution to the problem; the third step, revise, consists in supervising and validating the proposed classification (this task is often carried by a human expert); in the last stage, Retain, it must be decided if the treated case should be included or not in the knowledge database in order to help in future situations.

CBR tools have been used in many different fields such as fault detection in batch processes[10], medical diagnosis[47], device fault detection[44], etc. Each domain has its own particularities and different techniques are required to provide a proper solution. These singularities concern the four stages of the CBR process. In the retrieve stage the way the data is provided is extremely important as the style the data is stored (plain or structured) conditions the metrics to follow while comparing different cases; moreover the type of the attributes (categorical, numerical, etc.), the attributes weighs, how to deal with missing data, etc. are also important aspects to take into account. The reuse of the retrieved case solution in the context of the new case focuses on two aspects: the differences among the past and the current case and what part of a retrieved case can be transferred to the new case (directly coping the solution, using the same methodology, etc.). Depending on the domain where CBR is applied these can mean to directly copy the solution of the retrieved case or its adaptation. The revise and retain stages are less affected by the application domain; the main difference between different CBR fields is the kind of expert which is in charge of deciding if the adopted solution was correct and if it must be retained into the knowledge base in order to increase system's awareness.

Chapter 3

Related Work

Chapter 3 presents different works and papers concerning the research exposed in this master thesis. It is divided in four different sections. The first one, workflow modeling, shows different methods which can be used to model workflow activities and applications where they have been used. In the second section, work related to workflow monitoring is exposed. The third section, workflow discovering and adaptation, presents a state of the art concerning workflow mining to handle data about workflows. It is divided in two parts: the first one exposes research about discovering unknown workflows given a set of data while the last one concerns the application of data mining techniques to known workflows in order to adapt or optimize them.

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 3.1). Some authors use other fields' representation models such as UML activity diagrams[32] or different types of petri nets (called WF-nets)[49, 68]. On the other hand other researchers have chosen to develop specific languages for workflow representation.

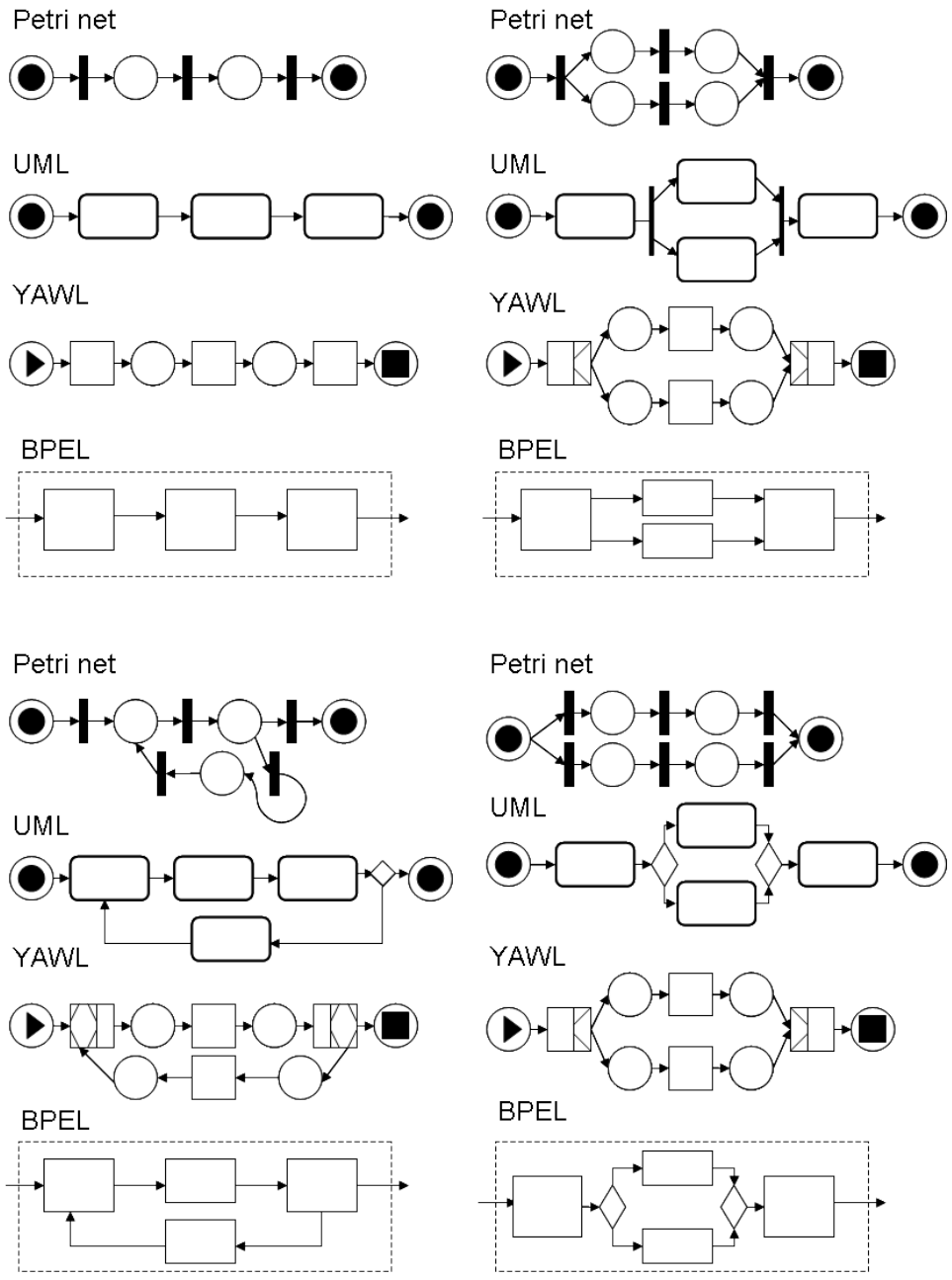

Figure 3.1: Workflow pattern examples in different modeling languages. Top left: sequential rouging. Top right: concurrent execution. Bottom left: iterative routing. Bottom right: exclusive or choice.

Unified Modeling Language[51] (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.[32], propose to use UML or to extend UML[7, 75] in order to model workflow while others have discussed its suitability. Dumas et al.[17] analyzed the UML notation by representing the workflow patterns defined in [52] and concluded that, unlike alternative commercial languages, UML provides support for waiting and processing states and decomposition tools while they syntax and semantics 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[42] (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[30], 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)[45], but also in scientific processes[73] or to model grid computing interactions[73]. Moreover, another interesting point of BPEL, is that can be easily transformed to other languages such as YAWL[13], BPMN (mentioned above) or to Petri net struchtures[28] as Hinz et al. and Brogi et al. show on their works.

Petri nets[46](PN) are an established tool for modeling and analyzing processes[64]. On the one hand, Petri nets can be used as a design lan-

guage 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[15] which allows a more accurate representation of the workflow representation. For example, Eshuis et al.[19] 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[19]. Other examples of extended Petri nets applied to workflow modeling is the one presented in [26] 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 [8] or by Alt et al. who use HPN to represent grid workflows[5].

In [67], 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[52]. 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 [52] 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[53].

Regarding the resources representation in workflows, despite some languages such as the mentioned UML or BPEL provide tools for represent-

ing 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 [36] 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 [29] 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 [33] 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.[66] 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 [50] 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.[21] 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[61].

Other previous related works regarding to workflows monitoring come from the multi-agent community. For example, in [72] 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 [76] an agent based system is also proposed to deal with coordination and management of workflows between virtual enterprises.

3.3 Workflow Discovering

Workflow discovering it is a reverse enginery method used to learn the workflow that defines a process from the practical experience. An event log is build by executing the process several times. This log is later used to construct a process specification which appropriately models the registered behavior. By assuming that each generated event refers to a single task, that each event belongs to a single case and that the events are ordered[68] a workflow equivalent to the process that generated the events can be found. As logs are extracted from real processes, they can contain data errors such as noise causing the events to be unordered, incomplete or mislabeled. Authors have defined different approaches to discover workflows according to different situations.

The first approach is the simpler and it assumes ideal logs with no noise and complete information (the log registered all the cases). It uses workflow nets as representation model. To *rediscover* the workflow which models the process the α *algorithm* [63, 62] is used. As there exists an infinite number of Petri nets which model a workflow, the α *algorithm* finds the simpler one by assuming that for many WF-nets two tasks are connected if and only if their causality can be detected by inspecting the log. However, this algorithm presents some limitations when dealing with certain kind of loops and different tasks with a common name cannot be detected as they are understood as AND or OR splits. Moreover, as workflow nets are based on Petri nets they present the same limitations, in consequence, α *algorithm* presents problems when dealing with non-free-choice constructs[16]. Some programs that use this approach to model workflows are EMiT[63] (Enchanced Mining Tool) and MiMo[62] (Mining Modeling), both from the

Eindhoven Technische Hogeschool.

In order to overcome the limitations of noise and completeness information heuristic approaches were developed. Noise in log registries can alter the order of the events making dependencies difficult to relate and making consecutive events look like parallel or complementary executions. *Little Thumbs*[74] is a tool which uses the α *algorithm* but includes heuristic techniques in order to gain robustness. First of all *Little Thumbs* builds a frequency table counting the times an event is followed by another one; then applies a set of rules in order to decide if the events which appear on the frequency table are consecutive actions, concurrent tasks, choices, etc. Finally, based on the results of the previous state, reconstructs the workflow.

In [57] Silva et. al present a probabilistic method for workflow learning. In it, they propose and alternative to the α *algorithm*: the *Learn-OrderedWorkflow*. Given a process event log, the algorithm detects which tasks can be executed in parallel and which ones are dependent among them by analyzing the task execution. The workflow returned is modeled using a graph notation and it is not necessarily a complete workflow (can have multiple ending points). To deal with the event log noise they perform the χ test and parametrize the algorithm according to the obtained value. To test their system, a simulated study based on a theoretical workflow that models the annual process of writing final reports at Clairvoyance Corporation is performed. A similar work can be find in [22], where Gaaloul et. al present a similar procedure but, in addition, they try to match the obtained dependency relations with the workflow patterns presented in [52] by Van der Aalst.

An alternative to deal with noise and incompleteness is the use of *InWolve*[27]. This software is a tool developed by the University of Vienna which, apart from dealing with noise, can work with event logs where events can refer to more than one task in a workflow and where to different task can use the same label. In order to achieve that, *InWolve* implements the *splitPar*, which given a data log builds a workflow w, and the *SAGtoADL*[27], which checks w synchronicity, algorithms. Another difference between this approach and the ones mentioned before is that *InWolve* uses ADONIS[20] as workflow representation language.

Process Miner[54] is a completely different approach from the ones mentioned above. While the previous commented tools are graph-based tech-

niques, the algorithms used by that *Process Miner* are rewriting methods and they are specially tailored towards mining block-structured workflows. Block-structured workflow models are modeled in a top-down way by setting one operator as starting point of the workflow and nesting other operators as long as the desired flow structured is obtained. In other words, a block-structured workflow is a tree whose leafs are always operands. To reach the block-structure that defines a process, *Process miner* reads the event log and builds a trace for each process instance; then, a time-forward algorithm constructs a process schema from the trace groups that is in the disjunctive normal form. In the following step, *Process Miner* tries to find relations between task orders and to identify the tasks that do not have a real precedence relation between them. In the final step an induction is performed for each decision point of the schema; decision trees are transformed into rules and then they are attached to the particular alternative operators.

3.4 Workflow Adaptation

Given a concrete workflow and an historic of the workflow executions a workflow can be refactored in order to simplify its representation, to obtain a more accurate representation for the modeled process or to solve design issues as incompleteness. Workflow mining can be also used to identify conflictive points, conflicts, and bottle necks in manufacturing processes so they can be optimized in order to improve the efficiency of the process.

In [71] a tool for automatic graph refactoring and completion are presented: the tool allows different improvements in real time as completing partial workflow graphs or speeding up of OR-join workflows. Moreover it provides local termination detection and refactoring for incomplete workflow graphs and multi ending workflow graphs in runtime. In order to achieve they use two different approaches which they call the *Normal*[70] and the *Refined*[69] *Process Structures* which are based on transforming workflow graphs into tree structure models.

Ho et. al present the *intelligent production workflow mining system* (IPWMS)[25], a method to analyze the quality of a manufacturing process and to optimize it. Using the information obtained from the execution of the process, the authors analyze the quality of the workflow using fuzzy logic rule sets. When an anomaly or a poor quality is discovered by its tool, an

external expert redefines the workflow and the goodness of the new workflow is predicted using artificial neuronal networks which have been trained with the previous workflows.

In [48] a supporting tool for workflow modification is proposed. The goal of this tool is to help in the modification process once the status of the workflow is known.

Subramaniam et. al consider that as workflows are manually designed, they entail assumptions and errors which leads to inaccurate workflow models. In [59] they expose that the critical points which slow a workflow and increase the uncertainty are the decision points (also known as XOR choices) and that, by positioning this choices to the earliest stages can improve process efficiency by decreasing their uncertainties and identifying redundant activities. They present techniques to analyze the event logs generated by the workflow management system in order to notice if XOR choices can be moved to earlier points of the workflow so the efficiency of the process increase.

In [11] workflow mining is used to distinguish tasks which apparently are identical but which consume sensibly different time amounts. Studing workflow event logs represented as temporally-annotated sequences (TAS)[24] they classify tasks according to duration. This information can be later used to study and to optimize the modeled process since situations as lack of resources or excessive load charges can be detected. This is done by giving a higher importance to the timing of the tasks than to the order in which they appear on the event logs[23].

Another approach to workflow adaptation is the application of case-based reasoning (CBR). CBR[2] has been proposed as a natural approach to the recall, reuse and adaptation of workflows and knowledge associated with their systems structure. Minor[41] et. al present a case-based approach to representation and index-based retrieval of past workflows in order to in order to adapt recent workflow instances to system modifications and innovations. The knowledge database they use is composed by UML workflows and they compare the different workflows using the metrics defined in [14]. Moreover, the AI group at the University of Greenwich also has applied CBR to workflow monitoring[34].

Another example of the CBR application to the workflow domain is CAKE[9]. Given a set of requisites and conditions, CAKE seeks for sim-

		Language	Formal Language	PN based	Pattern Compatibility	Resource Definition
Workflow Modeling	UML [51]	Standard	No	No	Low	Yes
	BPMN [42]	WF Specific	No	No	Low	Yes
	YAWL [67]	WF Specific	Yes	Yes	High	No
	Petri Nets [64]	Standard	Yes	Yes	High	No
	RAPN (our approach)	WF Specific	Yes	Yes	High	Yes

Figure 3.2: Summary of workflow modeling state of the art.

		Supervision	Logs	Prediction	Workflow Triggering	WF adaptation	Multi Agent
Workflow Monitoring	Hong et. Al [29]	Yes	Yes	No	Yes	No	No
	Kanana et. Al [33]	No	No	No	Yes	No	No
	Frankowiak et. Al [21]	No	No	No	Yes	No	No
	Wanget. Al [72]	No	No	No	No	Yes	Yes
	Our approach	Yes	Yes	Yes	No	Yes	No

Figure 3.3: Summary of workflow monitoring state of the art.

ilar requirements in a past cases database and models a workflow model according to the requirements; It is used in different domains such as fire extinction, software engineering, microchip manufacturing or medicine.

3.5 Summary

In this section we resume the work studied in this chapter according to different parameters. The parameters used to qualify workflow modeling tools are the following:

- **Language** describes if a language is a standard or if a language was expressly created for workflow designing.
- **Formal Language** describes if the modeling language is supported by a formal notation.
- **PN Based** indicates if the modeling language is based in Petri nets.
- **Patter Compatibility** describes the number of Van der Aalst[52] patterns which can be represented by the workflow language
- **Resource Definition** shows if the language allows to include resources into its representation.

	WF Discovering	Representation	WF Refactoring	WF Adapting	Quality Analysis	Time consideration	CBR
EMiT[63]	Yes	Graph Based	No	No	No	No	No
MiMo[62]	Yes	Graph Based	No	No	No	No	No
Little Thumbs [74]	Noise Robust	Graph Based	No	No	No	No	No
Silva et. Al [57]	Noise Robust	Graph Based	No	No	No	No	No
InWolve[27]	Noise Robust	Graph Based	No	No	No	No	No
Process Miner [54]	Noise Robust	Block Structured	No	No	No	No	No
Vanhatalo et. Al[69]	No	Graph Based	Yes	Completion	No	No	No
Reichert et. Al[47]	No	Graph Based	No	Modification	No	No	No
IPWMS [25]	No	Graph Based	Yes	No	Yes	No	No
Subramaniam et. Al [59]	No	Graph Based	No	Modification	Optimization	No	No
Berlingerio et. Al [11]	No	Graph Based	No	Completion	No	Yes	No
Minor et. Al [41]	No	Graph Based	Yes	Modification	No	No	No
CAKE [9]	Yes	Graph Based	No	No	No	No	Yes
Our approach	No	Graph Based	Yes	Modification	No	No	Yes

Figure 3.4: Summary of workflow mining state of the art.

Regarding the workflow monitoring the used parameters are listed below:

- **Supervision** shows if the software captures the state of the workflow at each variation.
- **Log** indicates if an event log is generated every time the workflow advances.
- **Prediction** describes if the tool includes delay prediction techniques.
- **Workflow Triggering** indicates if the software is in charge of starting the workflow actions.
- **WF Adaptation** shows the software is automaticly adapted when the workflow changes.
- **Multi Agent** describes the software uses multi agent technology in the monitoring procedure.

By analyzing the state of the art concerning workflow modeling we have noticed that, as can be seen in Figure 3.2, there is no workflow specific modeling languages which include resources inside its formal definition. Our challenge is to fill this gap with the resource-aware Petri nets. Regarding the workflow monitoring (Figure 3.3), any of the works analyzed on this chapter provide prediction tools for an early detection of the workflow delays. We pretend to add this feature to the ones which are covered by other tools (supervision and data storage).

Figure 3.3 describes the workflow adaptation and discovering state of the art in the following therms:

- **WF Discovering** shows if the tool can rebuild a workflow given an event log.
- **Representation** indicates how workflows are represented.
- **WF Refactoring** describes if the software can modify the structure of the graph in order to simplify it.
- **WF Adapting** shows how a workflow can be adapted by a modification or by completing it.
- **Quality Analysis** describes if the tool evaluates the performance or the workflow and if it can be optimized.
- **Time Consideration** points if the software takes into account the workflow deadlines when refactoring or modifying the workflow.
- **CBR** indicates if the approach uses case-based reasoning techniques.

Analyzing the mentioned aspects, we detected that there is not many research involving at the same time case-based reasoning with workflow modification and refactoring. Moreover, time constraints are not considered in almost any of the articles analyzed. In this thesis we focus in the combination of this two aspects: Time and case-based reasoning for workflow modification.

Chapter 4

Workflow Management System

As Figure 4.1 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; to prevent this delays the WMS uses case-based techniques in order to readapt successful past solutions so delays can be avoided or minimized; moreover, to diagnose the cause of a delay, a complex event processing system analyzes the event log generated by the WMS so delay reasons can be diagnosed.

In this chapter we first introduce the WMS, then we present a Petri net extension which combines timed Petri nets, colored Petri nets and adds the concept of resource and subpath: resource aware Petri nets (RAPN). To notice the delays we propose to monitor the workflow process on the task level and heuristically calculate the retards instead of monitoring the whole process. Finally we propose the foundations for refactoring and adapting ongoing workflows using cased based reasoning when a delay is detected.

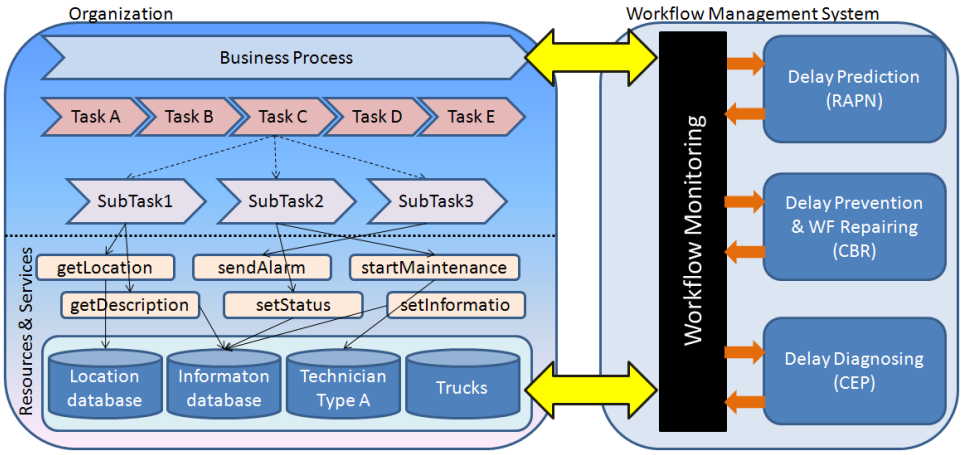

Figure 4.1: Overview of the whole system.

4.1 Workflow Management System

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

The workflow management system (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 4.2. 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:

1. Receives a request for a business process Bp_i from the system.

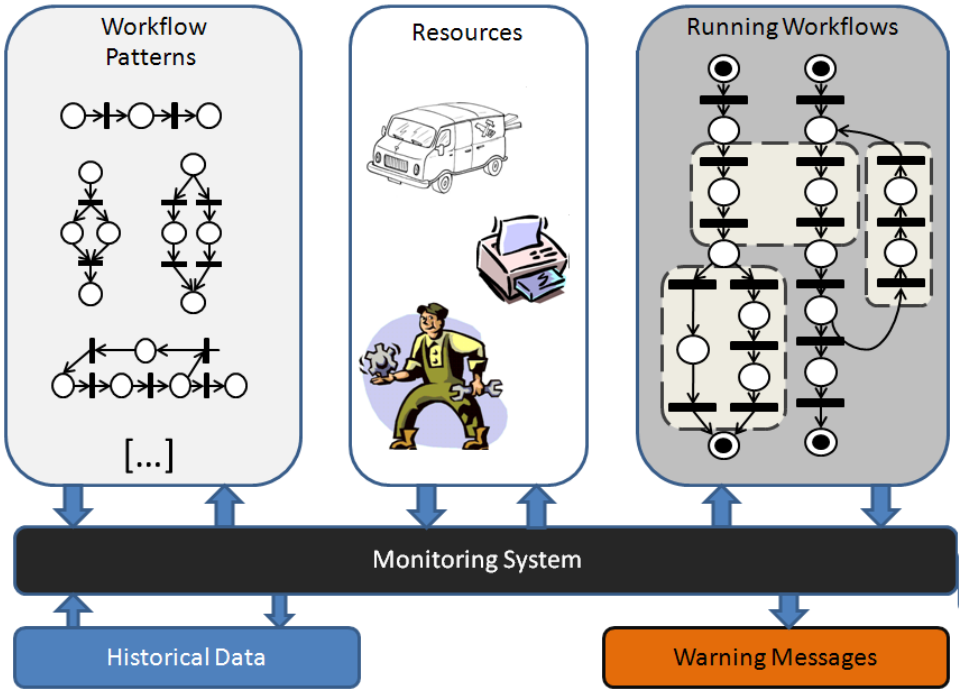

Figure 4.2: The workflow management system is responsible for modeling and monitoring the workflow and sends warnings when a possible delay is detected.

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 4.2), WMS estimates durations of activities so delays, lack of resources or devia-

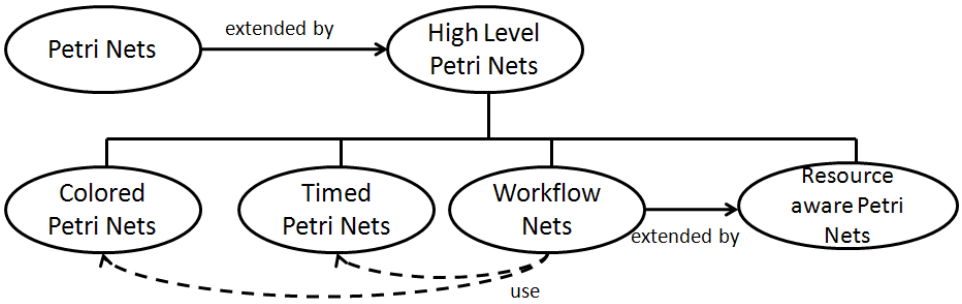

Figure 4.3: Resource-aware Petri nets extend the workflow net notation including resources.

tions 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.

4.2 Workflow Modeling: Resource Aware Petri Nets

Workflow modeling using high level Petri nets has been broadly studied during the last years[64, 19]. As our work is specially focused on delays prediction, 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 (Definition 2) with a new *resource* element (Figure 4.3). We called this extension *resource-aware Petri nets* (RAPN) (Definition 6). RAPN incorporate resources to high level Petri nets[3]. Resources (Definition 3) are related with sets of consecutive transitions (forming subpaths, Definition 4) 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 Workflow net is a 4-tuple* $\langle P, T, A, TO \rangle$ *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*
- *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

Definition 3. *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 4. *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,* $TS = \langle t_s, t_e \rangle$.

Definition 5. *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 6. *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 viceversa* $A : (P \times T) \cup (T \times P)$
- *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*

4.3 Delay Prediction with Resource Aware Petri Nets

The workflow management system (WMS) is responsible for monitoring the development of a workflow case. 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. Then, for each workflow the mean time and the standard deviation can be learned from past executions.

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

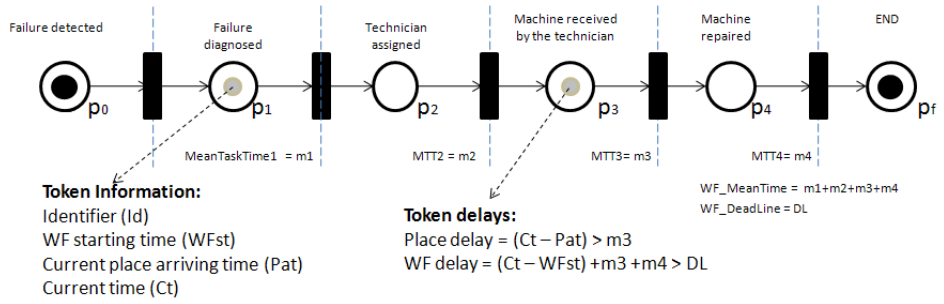

Figure 4.4: Graphical representation about how the delays are estimated based on the token information, the transitions mean times and the workflow state.

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. Usually, 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. There, in addition to the mean time of a workflow we also learn from past executions its estimated time duration.

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.

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 (e.g. Figure 4.4) as we can notice when a task is exceeding it's normal execution time (Definition 11), sending *task delay* warning alarms. 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 (Definition 13) then a *workflow delay* alarm is triggered. Thus, the method we propose to monitor business process is prerformed both at the workflow and at the task level.

4.3.1 Definitions

Definition 7. *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 8. *Workflow mean time $wf\mu_{time}(WF)$ is the mean of the time spent by tokens in a workflow WF*

Definition 9. *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 10. *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 11. *A Task Delay associated to a token TO_i, $SD(TO_i)$, 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 12. *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 13. *A workflow instance represented by a token TO_i has a delay, $WD(TO_i)$, if its estimated duration exceeds the workflow mean time, $WD(TO_i) = EDW(TO_i) > wf\mu_{time}(WF)$.*

4.3.2 Predicting Method

In order to make predictions the WMS stores the following 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. Every time a task or a decision is taken in any of the ongoing workflows, a transition is fired in its corresponding resource-aware net. 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
```

During the monitoring two kind of warning can be send: *Task Delay*(Definition 11) and *Workflow Delay*(Definition 13). 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. 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.

4.4 Workflow Repairing: Case-Based Reasoning

In this section we propose the use of case-based reasoning (CBR) as a solution for workflow repairing when delays are predicted. Organizations and business companies tend to have preventive and procedures for solving problems, avoiding delays etc. This actions can have different nature: changing workflow priorities, reallocating resources, modifying business process applying escalation rules, etc. Nevertheless, the continuous changes and modifications on workflow environments and the introduction of new business

processes can produce new situations in which the existing procedures are not effective or even causing the absence of defined preventive actions. In this sense, CBR offers way of dealing with the comparison of the environment marking at the time a delay is predicted with previous environment markings, such that a preventive procedure can be obtained by the merging of previous successful preventive procedures.

This section summarizes which aspects should be taken into account to design a CBR tool for refactoring workflows when a delay is predicted in order to minimize its impact. Firstly, we discuss about how a case should be defined and compared; then the appropriate kind of reuse is studied; finally we expose an example about how this CBR tool should work.

4.4.1 Case Representation

Before discussing which retrieving techniques it is important to define the data contained by a case. In the problem we are facing a case is a workflow environment marking in a concrete time instant and the solution applied to it. Therefore the information contained in a case would be the problem (workflow status: on time, delay predicted, delayed), the context (the total number and the type of system resources, the resources available and its status, the marking of the instantiated workflows which includes the petri net modeling, token position, the workflow deadline, the time the workflow started and the resources used by the WF and the instantiated workflow status) and the solution (the collection of preventive procedures used to refactor the workflows).

Consequently to this case definition, a case would be defined as structured data since neither the number or the kind of workflows and resources would be static. In other words, one case could contain information about just one workflow type and one resource type while another one could store information about hundreds of them, making the plain representation completely inviable. In Figure 4.5 a possible representation of a case is shown.

4.4.2 Retrieve

Regarding the comparison between cases, it should combine different kinds of metrics since the attributes which define a case are of different type. There is a group of attributes (number of resources, number of workflows, etc.) which

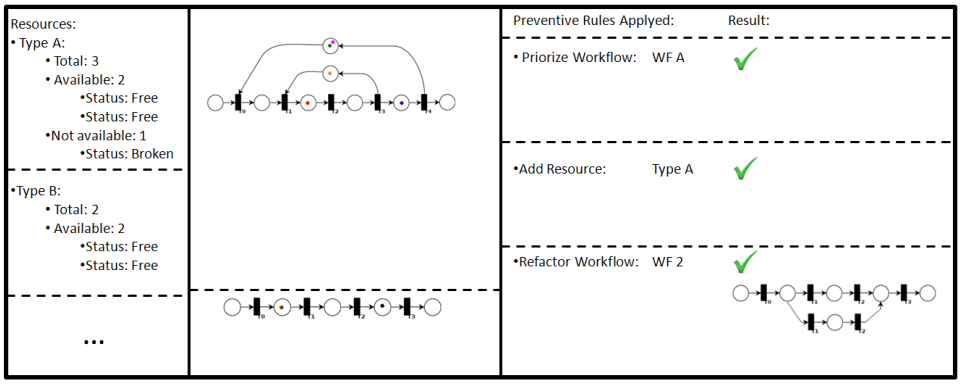

Figure 4.5: Possible representation of a workflow environment case.

can be easily compared using geometrical metrics. The structured data, such as the resource status or the instances inside each workflow, could be represented as an structured language (e.g. XML) file so a suitable method for comparing this part of the cases could be an adaptation of [56]. The similitude between the workflow models should be calculated using graph metrics. Graph comparing is a computational complex task so the process of comparing would require good equipment and some time, therefore we recommend to use the metrics presented in [14] should be considered. The final step of the comparing process is the integration of the three kinds of metrics.

4.4.3 Reuse

The reuse stage is also an important step for the workflow delay prevention using case based reasoning. Conversely to other domains where CBR as used to solve problems, in this application the solution used in a past case cannot be directly used as a solution since is highly improbable that the workflow environment is exactly the same in two different cases. In consequence, new case reuse for adapting the solution of the most similar cases should be explored.

For preventing a delay in a workflow environment the four most common kind of solutions to be considered are the following:

- Increasing the priority of the workflows susceptible to be delayed. This makes that a possibly delayed workflow waiting for a resource, uses it as soon as it is released by another workflow.

- Increasing the amount of resources in the system. This solution can imply economic costs are resources are expensive, but, in some emergencies, the cost of including a new resource on the system can be lower than the cost of having a delay in certain business process.

- Modifying and extending the workflow tasks deadlines. This method can be used when the workflow susceptible to be delayed has a low priority. In some situations the assumption of a delay in a secondary workflow can avoid problems in the primary ones. This solution is often combined with the first.

- Refactoring workflows. Some workflows can be altered by switching tasks, realizing activities concurrently or by adding an alternative procedure. This one can reduce or avoid the delay but its application is not trivial as requires an analysis of the modification impact into the system and new deadlines and mean time tasks must be taken into account.

According to the options presented before, the CBR should not only provide the best kind of solution to face the delay but also provide the parameters and the adaptation needed to fit the solution to the current case. In the adaptation process the solution can be the adjustment of a single solution, the merging of two solutions, or a mixing of different parts of many solutions. Regarding the first three type of possible solutions, the adaptation of the retrieved solution concerns variables of the system (resources, deadlines and priorities). A good starting point would be to study and adapt the reuse techniques used by Cynthia Marling et al. in [38] where a CBR calculates the insulin dose required by a diabetic patient. On the other hand, the refactoring of workflows shares a lot of similarities with planification problems as the solution to provide is not a value for a variable but an structured graph or structured data. Consequently, the use of akin strategies as the proposed in [39], where CBR is used to plan the behavior of small football robots, or the used in [78] where, in a case-based planing problem,

they convert into constrains the different pieces of the plans retrieved and they build a new planing based on this constrains.

4.4.4 Revise and Retain

The revise and the retain phases required for this CBR do not present any special particularity as they do not have special needs. An expert supervises the taken decisions and if they can contribute to the case data base with new knowledge, they are added to it.

4.5 Workflow Delay Diagnose: Complex Event Processing

Despite the prediction methods presented above, it is important not to forget the workflow diagnose as predictions sometimes can fail and delays can be produced anyway. An automatic analysis of the delays detected can help to apply corrective measures and to improve the system in future. The execution of the workflow management system can produce events as a consequence of its execution, this events are highly related to our Petri net extensions: transition triggering, resource allocation, resource releasing, etc. Since its analysis can be useful and can supply information about the two main delay reasons (task delay or no available resources) we propose to carry the workflow diagnosing using complex event processing[58].

4.5.1 Events on Workflow Nets

An event is an object that records a activity in a system. The event represents the activity and it may be related to other events [37].

To meet our needs we have created the representation of six events which correspond to the most basic activities that a workflow can perform [43, 64]:

E1 Transition activation $\Rightarrow act(W^T, T_j, t_i)$

E2 Resource allocation $\Rightarrow alloc(W^T, r^k, P_j, t_i)$

E3 Resource liberation $\Rightarrow free(W^T, r^k, P_j, t_i)$

E4 Advance token $\Rightarrow go(W^T, T_j, P_j, t_i)$

E5 Start workflow $\Rightarrow st(W^T, t_i)$

E6 End workflow $\Rightarrow end(W^T, t_i)$

Where W^T is a type T workflow, T_j stands for the j-th transition, r^k is the k-th resource, P_j represents the j-th place, t_i is the i-th token and all the event have an associated creation time stamp.

4.5.2 Complex Events on Workflows

A complex event is an event abstraction that signifies a set of events and their relationships over a time interval. There are three relationships between events that causes the most common event abstractions: time (event A happened before/after event B), cause (event B happened because of event A) and aggregation (event A signifies events B_1, ... B_n).

In order to create a starting point for our study case we have created six complex events that represent example situations we can find during a workflow execution. They are the following

C1 Lack of resource: When an E1 but no E4 is detected means that the token can not advance due a lack of some resource.

C2 Activity delay: If an E4 event is detected, but no E4 event is noticed in the following $[ts(E4), ts(E4) + duration(E4.P_j)]$ time interval, it means that the estimated time to finish the activity has been exceeded and hence there is a delay.

C3 Lack of resource delay: If a complex event C1 occurs and then a complex event C2 occurs for the same process, we can conclude that there is a delay in the process because of the lack of a resource.

C4 Transition delay: If a complex event C2 and then no E1 event is detected, it implies that something in the process is exceeding its estimated time.

C5 Workflow delay: If an E5 event enters the system but no E6 is detected in the following interval $[ts(E5), ts(E5) + duration(E5.W^T)]$ then a delay has appeared at some point during the execution delaying the whole process.

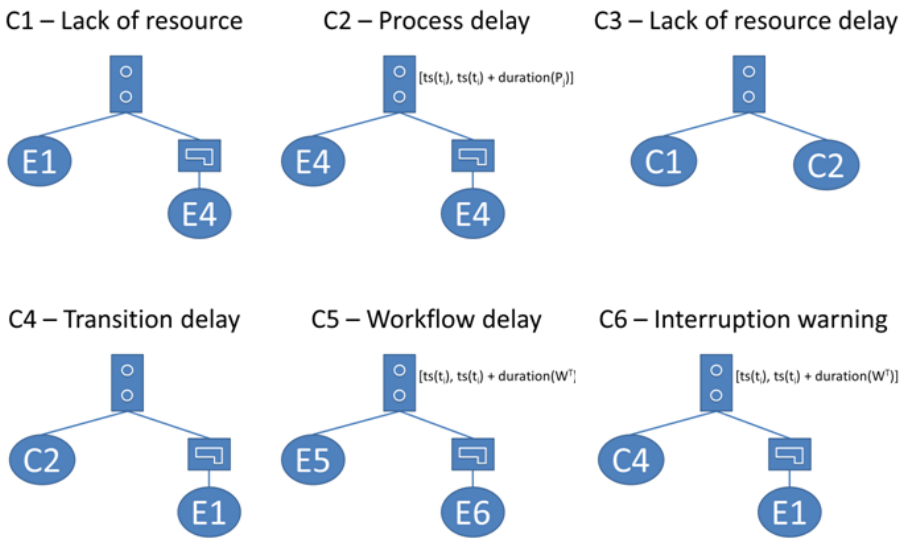

Figure 4.6: Graphical representation of the complex events.

C6 Interruption warning: When a C4 is detected but no E1 arrives in the following $[ts(C4), ts(C4) + duration(C4.W^T)]$ it means that the process has been stuck for a long time in the same place and something that was not expected is happening.

Where functions $ts()$ and $duration()$ returns the time stamp and the estimated time it should take respectively. A graphical representation can be observed in Figure 4.6 where the colon symbol represents an ordered sequence of elements, the negation symbol "¬" represents the non-existence of the elements bellow and the values inside the braces are the time interval in which the the events (circles) must occur.

4.5.3 Rules on Complex Events

Using those complex events, a set of rules can be developed to infer operations that should be carried out by workflows to avoid sever delays. These rules are like the following:

R1 IF C1 THEN check resources status

R2 IF C3 THEN prioritize

Such kind of rules are similar to any rule based system, and are domain dependent according to the application field.

4.6 Summary

To sum up, our proposal for workflow monitoring, delay prediction and diagnosing is a workflow management system (WMS) based on a new Petri net extension which introduce the resource concept into the Petri net notation: resource-aware Petri nets (RAPN). In our WMS the business processes are modeled as RAPN in such a way that the resources are not considered from a single workflow point of view but from the whole workflow environment. In this thesis we presented a monitoring system which, by monitoring the business process from the task level and by considering the resource information provide by RAPN, can estimate which workflow are susceptible to be delayed so preventive actions can be taken avoiding or minimizing the delay. The technique proposed to chose preventive actions is the use of case-based reasoning since the analysis of the data log provided by the WMS combined with the knowledge of which preventive actions have been taken in past can provide solutions to new problems. Moreover, we endowed the WMS with a complex event processing system which, studying the event log generated by the WMS, allows to diagnose the causes of the unexpected and unpredicted delays.

Chapter 5

Experimentation and Results

In this chapter we test the work previously presented in this thesis, concretely we focus our experiments on the workflow monitoring and on the delay prediction process. To test the performance of our system we modeled and simulated a set of workflows extracted and adapted from the AIMES project [4]. The chapter is divided in three sections, the first one presents the tested scenarios, the second one shows the results obtained and, finally, in the last one, the value of the results is discussed.

5.1 Experimental Setup

Our first prototype consists in 3 modules (Figure 5.1): a workflow framework, a workflow simulation engine and the workflow management system (see appendix A). The workflow framework can load Petri nets defined by XML or load Petri nets designed with the PM Editeur[77] 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 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

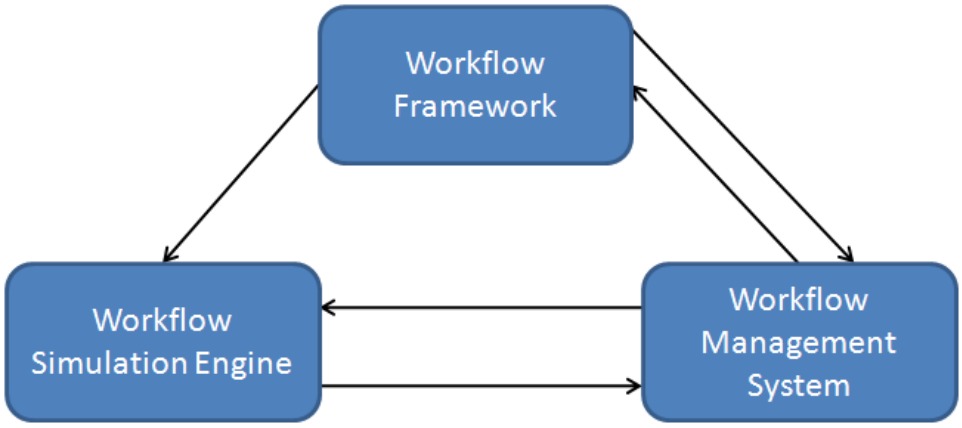

Figure 5.1: Prototype architecture.

delays and to ask the workflow motor to fire the transitions.

5.1.1 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.

- **Reactive Maintenance Interventions - RMI** (Figure 5.2 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 5.2 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

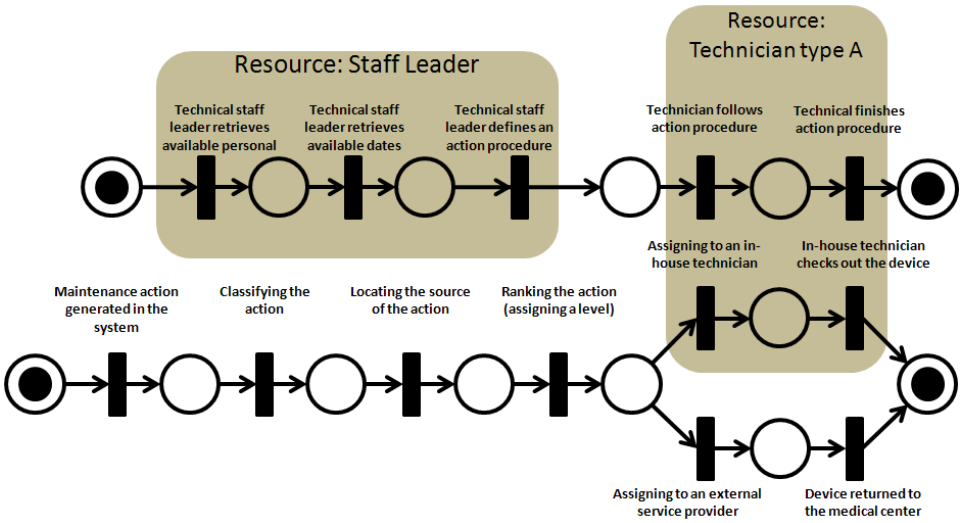

Figure 5.2: Top: Maintenance event escalation management. Bottom: Reactive maintenance intervention.

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 5.3): 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 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 5.4: This workflow is an extension of the *Reactive Maintenance Intervention* (RMI) workflow presented before. As the previous one,

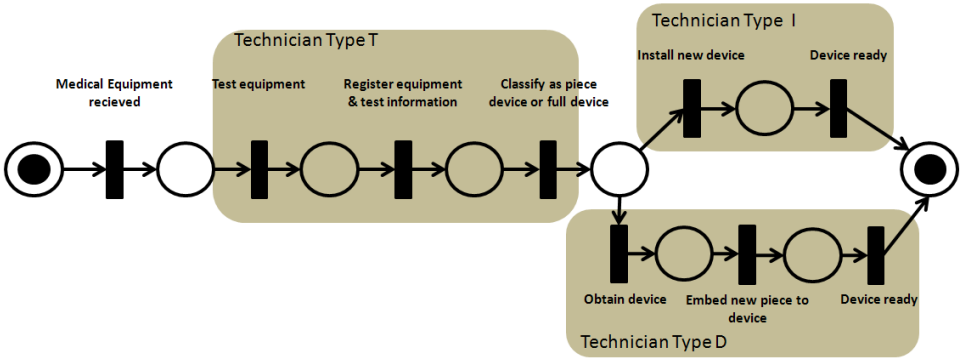

Figure 5.3: Inventory and Installation of New Equipment.

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.

5.1.2 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

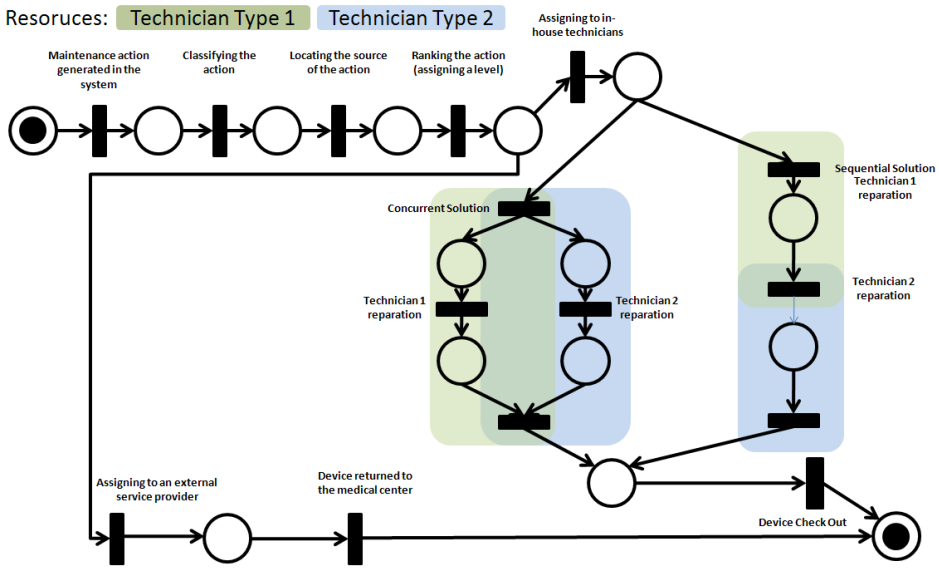

Figure 5.4: Multiple Reactive Maintenance Intervention.

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 manage-*

ment . 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 case created to analyze the behavior of the system when many workflows share the same resource. Figure 5.5 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.

5.2 Results

Figures 5.6, 5.7, 5.8, 5.9, and 5.10 shows the obtained results 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

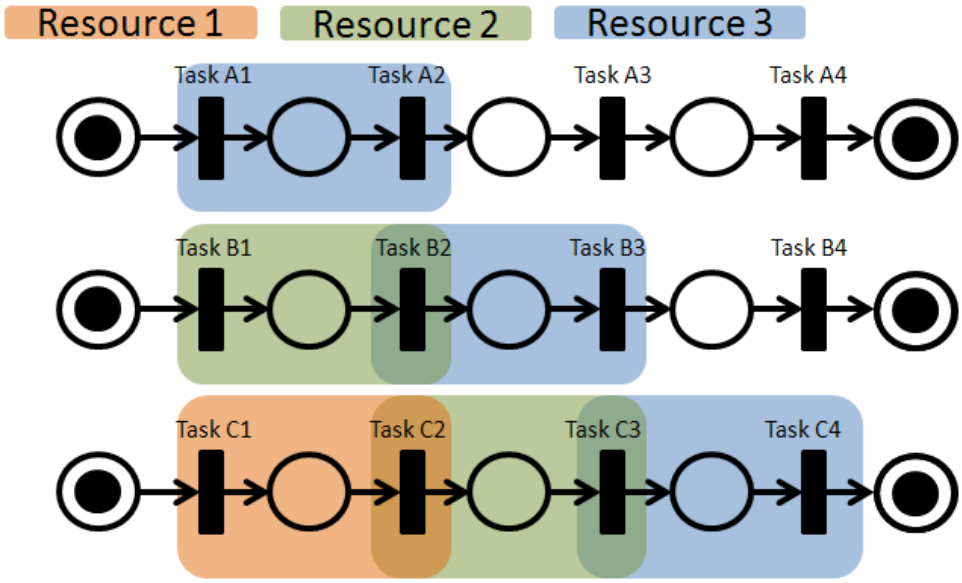

Figure 5.5: Scenario 5 graphical description.

instants in which our tool predicted a delay for the workflow are marked with a vertical line.

Figure 5.6 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 5.7 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, 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.

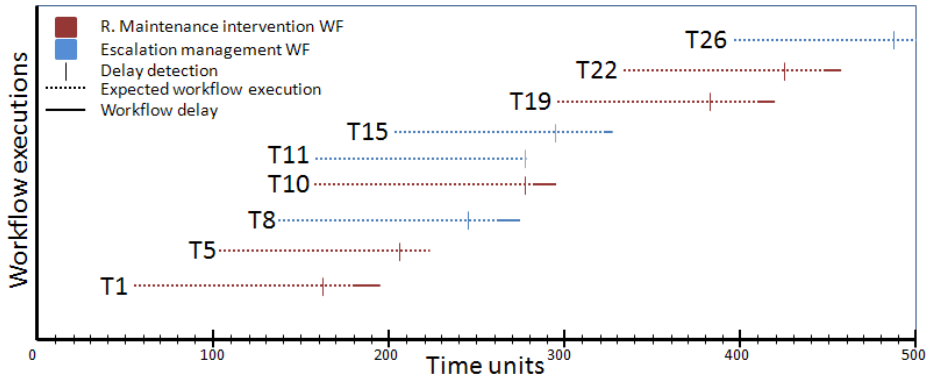

Figure 5.6: Result of the first scenario where the system resources are 4 type A technicians and a 1 technician staff leader.

Figure 5.8 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.

In Figure 5.9 the output of the fourth scenario is shown. The simulation of the Multiple reactive maintenance intervention generated 97 workflow instances but only 5 have been predicted as delayed workflows (T1, T5, T43, T77 and T78), nevertheless two of them are false positive as T5 and T43 ended inside the stipulated deadline. The low number of delays produced is probably caused by the fact that only one workflow is monitored in this scenario.

Finally, Figure 5.10 shows a synthetic example which shows the behav-

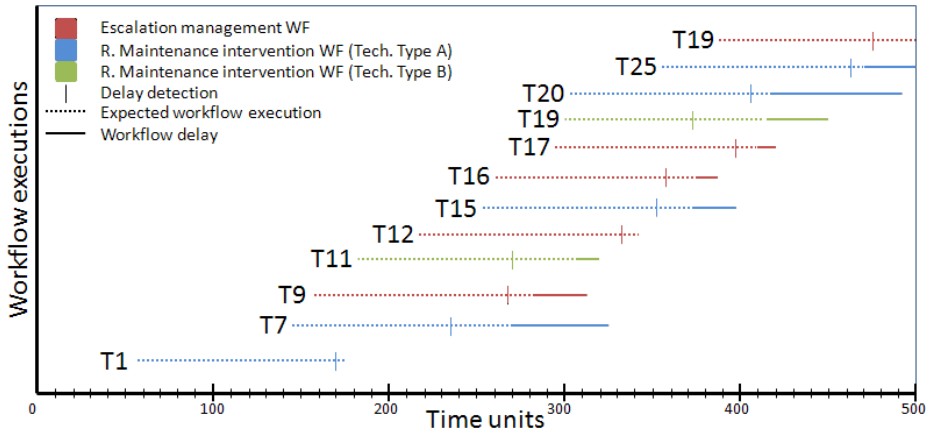

Figure 5.7: Result of the second scenario where the system resources are 3 type A technicians, 1 type B technician and a 1 technician staff leader.

ior of an organization where a resource is shared by many workflows. 42 workflows where generated among the simulation, 13 of them classified as susceptible of suffering a delay (T5, T9, T10, T13, T14, T16, T17, T27, T28, T32, T34, T37 and T38). 10 where correctly tagged while T9, T17 and T27 had an erroneous classification. Regarding the typology of the predicted workflows, 8 used 3 different types of resources, 2 needed two kind of resources and 2 required just 1 resource. The results of this scenario are quite similar to the third experiment as in both a high number of resources is used.

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 7(scenario2), token 22(scenario3) or token 16(scenario5) 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 service workflow, especially in long duration workflows as medical device maintenance operations (which can have long term deadlines) or manufacturing processes (with midterm deadlines).

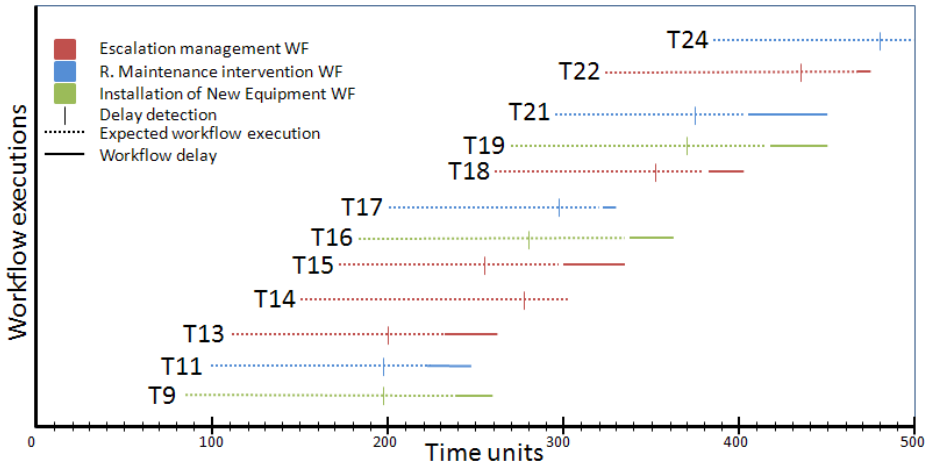

Figure 5.8: Result of the third scenario where the system IINE is added to the workflow environment.

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.

If we compare the third scenario with the fourth we can realize that the number of delays is much higher in scenario 3 than in scenario 4. This is caused by the higher number of workflows in the organization since, despite the fact that the *multiple reactive maintenance intervention* is a complex workflow it is not affected by other procedures and all the workflows instantiations have a similar behavior. Still focusing in this two scenarios, we can see that the mismatched classifications is higher in scenario 4. Probably, this points out that our workflow management system provides a more accurate delay prediction when different types of workflow are interacting and allocating resources.

The fifth scenario was created in order to analyze the behavior of the system when several workflows share the same resource and when a work-

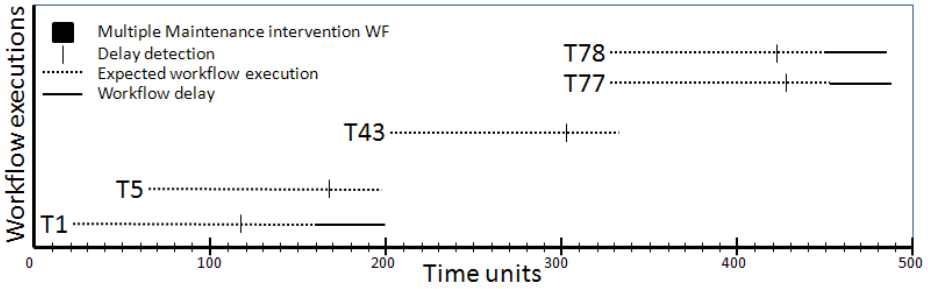

Figure 5.9: Result of the fourth scenario where only the multiple reactive maintenance intervention is simulated.

flow uses many resources. In this case in an environment with 3 different resources one of them is used by all the workflows inside the organization and with one workflow using all the resource types. As expected, the workflow involving all resources types is the one which suffers a higher number of delays and the biggest ones. The workflow management system have been able to detect this fact and all the delays produced in the *Workflow C* where detected with, at least, 30 time units which a reasonable margin for starting a preventive action. Nevertheless, it is important to remark than almost all the false positive classifications in this scenario have been done in the *Workflow C* which shows than the algorithm tends to point delays in the workflows which use more resources.

Table 5.1 shows the confusion matrix of the two presented experiments. If we analyze the results we can observe that in any of the performed experiments appear false negative classifications (delayed workflows classified as on time workflows). This is a remarkable detail as it means that no delayed workflows are ignorated. 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. Regarding the false positives we can see that they also appear in all the tables, representing around the 6% of the classified workflows on tables a, b, c and e. The impact of the false positives is directly proportional of the cost of applying a preventive action, in the workflow domain we consider than 6% is a reasonable per-

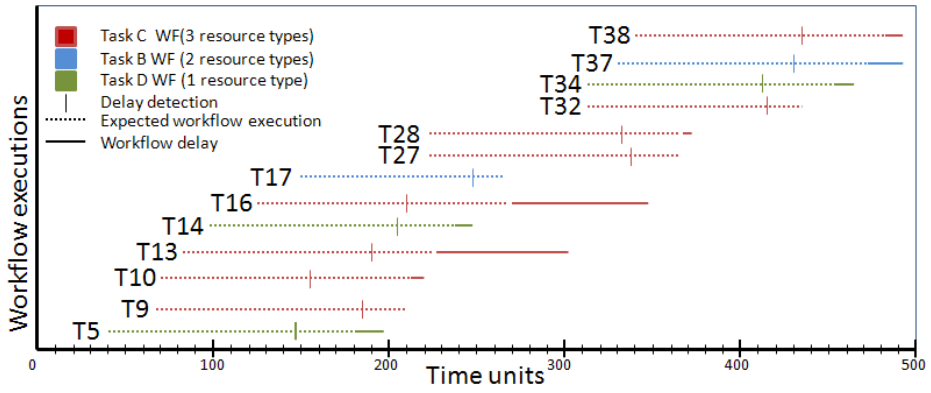

Figure 5.10: Result of the fifth scenario where many workflow instances share the same resource.

centage. However, in table d, despite the fact that the percentatge of false positives is lower (2.01%) we can see that means that those 2 false positives represente a 40% of the workflow classified as delayed, probably this fact is cause by the concurrent usage of workflows and it would be a good idea to study this case in detail in future so the prediction algorithm can be more precisely tuned.

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.

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)

PC\RC	Delay	On Time
Delay	3	2
On Time	0	92

(d)

PC\RC	Delay	On Time
Delay	10	3
On Time	0	39

(e)

Table 5.1: These tables show the confusion matrix from the different scenarios (a)Scenario 1 confusion matrix. (b)Scenario 2 confusion matrix. (c)Scenario 3 confusion matrix. (d)Scenario 4 confusion matrix. (e)Scenario 5 confusion matrix.

Chapter 6

Conclusions and further work

This final chapter presents the conclusions obtained from the research presented in this thesis. The chapter is divided in three sections. The first one exposes the conclusions, the second points the contributions of this thesis to artificial intelligence while the last one proposes some options to continue this work.

6.1 Conclusions

In this thesis we have presented a new approach to cover a need that nowadays is becoming a key point in business processing, the workflow monitoring.

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 services are not enough to represent all the variables that can affect their efficiency, as the available resources. That is, Petri nets, a common modeling tool, allow to represent the resources needed by an activity, but neither the total amount of resources available in the system nor their dependencies.

This work has faced different problems regarding workflow monitoring: how to model workflows including information about the resources needed

to its execution; how to monitor a workflow for predicting possible delays in its execution; how to decide which preventive actions should be taken when a delay is detected; and how to diagnose the reason of a delay so can be avoided in future.

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 for 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. According to this predictive capacity, we have discussed how a preventive action to avoid or minimize delays can be chosen; we proposed the fundamentals for a case-based reasoning tool to exploit the information stored by the workflow management systems. For this purpose we present a possible case definition of a workflow environment and we discussed the kind of reuse techniques which should be used.

Moreover, taking advantage of the research carried out by other partners of the *eXiT research group*, we include a complex event processing engine so unpredicted delays it can be diagnosed by analyzing the event log generated by the workflow management system (although this is not the focus of this thesis).

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 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.

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.

6.2 Contributions

The main contribution of this thesis to the artificial intelligence community are the creation of a high-level Petri net which includes in the flow modeling the resources available inside a workflow environment. This extension, which is based in dense-time Petri nets and color Petri nets, is *resource aware Petri nets*. This fills a gap in the workflow representation field since there exist some modeling languages (e.g BPEL) which include system resources but none of them offer a formal notation and the same compatibility with the Van der Aalst patterns[52] as Petri nets.

Moreover, we provide a workflow management technique which by monitoring workflows in the task-level (in the transition level when talking about Petri nets) can predict unexpected behaviors in the workflow execution such as delays.

The research work concerning the workflow modeling, the delay prediction and the workflow monitoring exposed in this master thesis has been submitted to the following conferences:

- **Albert Plà**,Beatriz López, Pablo Gay, and Joaquim Meléndez. Resource aware Petri nets for Service Workflow Monitoring and Delay Prediction. 8^{th} *International Conference on Service Oriented Computing (ICSOC)*. 2010, San Francisco, California. (Submitted)

- Pablo Gay, **Albert Plà**, Beatriz López, Joaquim Meléndez and Regina Munier. Service workflow monitoring through complex event processing. 15^{th} *IEEE International Conference on Emerging Techonologies and Factory Automation (ETFA)*. 2010, Bilbao, Spain. (Accepted)

Regarding the work involving complex event processing, which is not the main theme of this thesis but it is also treated, it has been published at the following congress:

- Pablo Gay, Beatriz López, **Albert Plà** and Joaquim Meléndez. Complex event processing for public-cycling transport supervision. 13^{th} *Congrés Internacional de l'Associació Catalana d'Intel·ligència Artificial (CCIA)*. 2010, Espluga de Francolí (Tarragona), Spain. (Accepted)

6.3 Further Work

The work started in this MSc Thesis will be continued as a part of a PhD program. A good starting point for continuing the work stated in the previous chapters would be the implementation of the case based reasoning module and its integration to the workflow management system; also, taking advantages of optimization tools from the constraint community, as the work done in [6] could be a good contribution.

The complex event processing module presented in Chapter 4.5 considers a reduced set of events and a small collection of rules. Its extension and a wider analysis of them could result in a better diagnosing of the unpredicted delays.

This work has been tested using a simulation based on a medical device maintenance organization, its validation into a real environments such as service oriented architectures or industrial manufacturing process would provide an important feedback of the points which could be improved and into the localization of weaker aspects.

Finally, the incorporation of agents into the monitoring stage, following the work started by Blake[12] and Ehrler et al.[18] could provide a more efficient monitoring and an interesting discussion about the resource-focused or the workflow-focused approach.

Appendix A

Implementation notes

A.1 The workflow editor

The workflow editor is the tool we created for defining and modeling workflows. For this purpose we reused an existing application: *The PM editeur*[77] (PME). PME provides a graphical editor for Petri net creation and edition, it allows the user to create Petri nets in a simpler way and to save them in its own file format. However, our proposed solution for workflow modeling is the usage of RAPN. In consequence, we decided to create a parser to transform the PME file format to a simpler XML file so it can be easy understood. We implemented an editor where resources can be created and, given a set of PME files, resources can be related with the Petri net transitions so they become RAPN. The designed workflows are stored on the workflow library.

A.2 Workflow management system

The workflow management system is also implemented in Java. It is composed by the following different classes which implement the resource aware Petri nets: arch, place, token, transition, resource and RAPN, the class which unifies all of them. WMS implementation is highly related with the WfSi java code as the WMS is checking the state of the WfSi in order to analyze the behavior of the workflows and to check that the resource and the tasks are evolving in the proper way.

Bibliography

[1] Bpel project. http://www.eclipse.org/bpel/ *Accessed June 10th, 2010.*

[2] A. Aamodt and E. Plaza. Case-based reasoning; foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

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

[4] AIMESproject. Deliverable 1.3: requirements specification, 2008-2010.

[5] M. Alt, S. Gorlatch, A. Hoheisel, and H.-W. Pohl. 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*, page 13, Washington, DC, USA, 2006. IEEE Computer Society.

[6] R. Bartak, J. Little, O. Manzano, and C. Sheahan. From enterprise models to scheduling models: bridging the gap. *Journal of Intelligent Manufacturing*, 21:121–132, 2010.

[7] R. Bastos, D. Dubugras, and A. Ruiz. 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*, page 291, Washington, DC, USA, 2002. IEEE Computer Society.

[8] B. Benatallah, P. Chrz;stowski-Wachtel, R. Hamadi, M. O?Dell, and A. Susanto. Hiword: A petri net-based hierarchical workflow designer.

Application of Concurrency to System Design, International Conference on, 0:235, 2003.

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

[10] X. Berjaga, A. Pallarés, and J. Meléndez. A framework for case-based diagnosis of batch processes in the principal components space. In *ETFA'09: Proceedings of the 14th IEEE international conference on Emerging technologies & factory automation*, pages 745–753, Piscataway, NJ, USA, 2009. IEEE Press.

[11] M. Berlingerio, F. Pinelli, M. Nanni, and F. Giannotti. Temporal mining for interactive workflow data analysis. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–118, New York, NY, USA, 2009. ACM.

[12] B. Blake.

[13] A. Brogi, R. Popescu, A. Brogi, and R. Popescu. Bpel2yawl: Translating bpel processes into yawl workflows, 2006.

[14] H. Bunke and B. T. Messmer. Similarity measures for structured representations. In *EWCBR '93: Selected papers from the First European Workshop on Topics in Case-Based Reasoning*, pages 106–118, London, UK, 1994. Springer-Verlag.

[15] M. Buscemi and V. Sassone. High-level petri nets as type theories in the join calculus. In *In Proceedings of 4th FOSSACS, volume 2030 of LNCS*, pages 104–120. Springer, 2001.

[16] J. Desel and J. Esparza. *Free choice Petri nets*. Cambridge University Press, New York, NY, USA, 1995.

[17] M. Dumas and A. H. ter Hofstede. Uml activity diagrams as a workflow specification language. pages 76–90. Springer Verlag, 2001.

[18] L. Ehrler, M. Fleurke, M. Purvis, B. Tony, and R. Savarimuthu. Agent-based workflow management systems(wfmss): Jbees - a distributed and adaptive wfms with monitoring and controlling capabilities. In *Journal of Information Systems and e-Business Management, Volume 4, Issue 1*, pages 5–23. Springer-Verlag, 2005.

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

[20] H.-G. Fill. Uml statechart diagrams on the adonis metamodeling platform. *Electron. Notes Theor. Comput. Sci.*, 127(1):27–36, 2005.

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

[22] w. Gaaloul, K. Bana, and C. Godart. Towards mining structural workflow patterns. 3588/2005:24–33, 2005.

[23] F. Giannotti and M. Nanni. Efficient mining of temporally annotated sequences. In *In Proc. SIAM Conference on Data Mining*, 2006.

[24] F. Giannotti and M. Nanni. Mining sequences with temporal annotations. In *In Proc. SIAM Conference on Data Mining*, pages 346–357. SIAM, 2006.

[25] C. L. A. I. G.T.S. Ho, H.C.W. Lau and K. Pun. An intelligent production workflow mining system for continual quality enhancement. *The International Journal of Advanced Manufacturing Technology*, 28(28):792–809, 2006.

[26] S. Ha and H.-W. Suh. A timed colored petri nets modeling for dynamic workflow in product development process. *Comput. Ind.*, 59(2-3):193–209, 2008.

[27] J. Herbst and D. Karagiannis. Workflow mining with inwolve. *Comput. Ind.*, 53(3):245–264, 2004.

[28] S. Hinz, K. Schmidt, and C. Stahl. Transforming bpel to petri nets. In W. M. P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Proceedings of the 3rd Int'l Conference on Business Process Management (BPM 2005)*, pages 220–235, Nancy, France, 2005. Springer Verlag.

[29] H.-S. Hong, B.-S. Lee, K.-H. Kim, and S.-K. Paik. 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*, page 166, Washington, DC, USA, 2000. IEEE Computer Society.

[30] J. R. M. Indulska. How good is bpmn really? insights from theory and practice. In *14th European Conference on Information Systems*, 2006.

[31] R. D. Jarrard. *Scientific Methods*. Online book http://www.emotionalcompetency.com/sci/booktoc.html, 2001.

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

[33] E. Kanana and M. Farhi. Monitoring information and data flows using triggers in a dynamic workflow environment. In *Proceedings of the 5th European Conference on Knowledge Managemen*, pages 175–179, 2004.

[34] S. Kapenakis, M. Petridis, J. Ma, and L. Bacon. Workflow monitoring and diagnosis using case based reasoning on incomplete temporal log data. In *Proceedings of the Workshop on Uncertainty, Knowledge Discovery, and Similarity in Case Based Reasoning UKDS, in Workshop proceedings of the 8th International Conference on Case Based Reasoning*, 2009.

[35] T. M. Koulopoulos. *The Workflow Imperative: Building Real World Business Solutions*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

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

[37] D. C. Luckham. The power of events: An introduction to complex event processing in distributed enterprise systems. Addison-Wesley Longman Publishing Co., Inc., 2001.

[38] C. Marling and J. Shubrook. Case-based decision support for patients with type 1 diabetes on insulin pump therapy.

[39] C. Marling, M. Tomko, M. Gillen, D. Alex, and D. Chelberg. Case-based reasoning for planning and world modeling in the robocup small sized league. In *In IJCAI Workshop on Issues in*, 2003.

[40] P. Massuthe and K. Wolf. Operating Guidelines for Services. *Petri Net Newsletter*, 70:9–14, Apr. 2006.

[41] M. Minor, A. Tartakovski, and R. Bergmann. Representation and structure-based similarity assessment for agile workflows. In *ICCBR '07: Proceedings of the 7th international conference on Case-Based Reasoning*, pages 224–238, Berlin, Heidelberg, 2007. Springer-Verlag.

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

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

[44] E. Olsson and P. Funk. Diagnosis of industrial equipment using case-based reasoning and sound comparison. In J. Malek, editor, *AILS2004*, page 8, April 2004.

[45] K. Pant. *Business Process Driven SOA using BPMN and BPEL: From Business Process Modeling to Orchestration and Service Oriented Architecture*. Packt Publishing, August 2008.

[46] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.

[47] C. Pous, P. Gay, A. Pla, J. Brunet, J. Sanz, T. R. Cajal, and B. López. Modeling reuse on case-based reasoning with application to breast cancer diagnosis. In *AIMSA '08: Proceedings of the 13th international*

conference on Artificial Intelligence, pages 322–332, Berlin, Heidelberg, 2008. Springer-Verlag.

[48] M. Reichert and P. Dadam. Adeptflex supporting dynamic changes of workflows without losing control. *Journal. Intelligent Information Systems*, 10:93?129, 1998.

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

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

[51] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Boston, MA, 2. edition, 2005.

[52] N. Russell, Arthur, W. M. P. van der Aalst, and N. Mulyar. Workflow control-flow patterns: A revised view. Technical report, BPMcenter.org, 2006.

[53] N. Russell, Arthur, W. M. P. van der Aalst, and N. Mulyar. Workflow control-flow patterns: A revised view. Technical report, BPMcenter.org, 2006.

[54] G. Schimm. Process miner - a tool for mining process schemes from event-based data. In *JELIA '02: Proceedings of the European Conference on Logics in Artificial Intelligence*, pages 525–528, London, UK, 2002. Springer-Verlag.

[55] L. Sheng, F. Yushun, and L. Huiping. Dwelling time probability density distribution of instances in a workflow model. *Comput. Ind. Eng.*, 57(3):874–879, 2009.

[56] H. Shimazu. A textual case-based reasoning system using xml on the world-wide web. In *Proceedings of the European Workshop on Advances in Case-Based Reasoning*, pages 274–285, London, UK, 200. Springer-Verlag.

[57] R. Silva, J. Zhang, and J. G. Shanahan. Probabilistic workflow mining. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 275–284, New York, NY, USA, 2005. ACM.

[58] D. Sottara, A. Manservisi, P. Mello, G. Colombini, and L. Luccarini. A CEP-based SOA for the management of WasteWater treatment plants. In *2009 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems, EESMS 2009 - Proceedings*, pages 58–65, 2009.

[59] S. Subramaniam, V. Kalogeraki, D. Gunopulos, F. Casati, M. Castellanos, U. Dayal, and M. Sayal. Improving process models by discovering decision points. *Information Systems*, 32(7):1037 – 1055, 2007. Special Issue on Intelligent Information Processing.

[60] J. Tick. Workflow model representation concepts. *Nemzetkzi Szimpziuma 7 th International Symposium of Hungarian Researchers on Computational Intelligence Workflow Model Representation Concepts*, 7, 2002.

[61] W. van der Aalst. Interval timed coloured petri nets and their analysis, 1993.

[62] W. van der Aalst and B. F. V. Dongen. Discovering workflow performance models from timed logs. In *International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002), volume 2480 of Lecture Notes in Computer Science*, pages 45–63. Springer-Verlag, 2002.

[63] W. van der Aalst, A. Weijters, and L. Maruster. Workflow mining: Which processes can be rediscovered? In *Eindhoven University of Technology*, pages 1–25, 2002.

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

[65] W. M. P. Van Der Aalst. Inheritance of interorganizational workflows: How to agree to disagree without loosing control? *Inf. Technol. and Management*, 4(4):345–389, 2003.

[66] W. M. P. van der Aalst and M. Pesic. Specifying, discovering, and monitoring service flows making web services process-aware. *BPM Center Report BPM-06-09, BPM Center*, 2006.

[67] W. M. P. van der Aalst and Ter. Yawl: yet another workflow language. *Information Systems*, 30(4):245–275, June 2005.

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

[69] J. Vanhatalo, H. Völzer, and J. Koehler. The refined process structure tree. *Data Knowl. Eng.*, 68(9):793–818, 2009.

[70] J. Vanhatalo, H. Völzer, and F. Leymann. Faster and more focused control-flow analysis for business process models through sese decomposition. In *ICSOC '07: Proceedings of the 5th international conference on Service-Oriented Computing*, pages 43–55, Berlin, Heidelberg, 2007. Springer-Verlag.

[71] J. Vanhatalo, H. Völzer, F. Leymann, and S. Moser. Automatic workflow graph refactoring and completion. pages 100–115. 2008.

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

[73] B. Wassermann, W. Emmerich, B. Butchart, N. Cameron, L. Chen, and J. Patel. Sedna: A bpel-based environment for visual scientific workflow modelling. In *In Workflows for eScience - Scientific Workflows for Grids*. Springer Verlag, 2007.

[74] A. Weijters and W. van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10:2003, 2001.

[75] G. Wirtz, M. Weske, and H. Giese. Extending uml with workflow modeling capabilities. In *CooplS '02: Proceedings of the 7th International Conference on Cooperative Information Systems*, pages 30–41, London, UK, 2000. Springer-Verlag.

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

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

[78] H. Zhuo, Q. Yang, and L. Li. Constraint-based case-based planning using weighted max-sat. In *ICCBR*, pages 374–388, 2009.