# A Generic Workflow Metamodel to Support Resource-aware Decision Making

Ravi Ramdoyal[1], Christophe Ponsard[1], Myriam-Amina Derbali[2], Gabriel Schwanen[2], Isabelle Linden[2] and Jean-Marie Jacquet[2]
[1]*CETIC Research Center, Charleroi, Belgium*
[2]*University of Namur, Namur, Belgium*

Keywords: Business Modelling, Business Process Management, Workflow Modelling, Exception Management, Decision Making, Resources.

Abstract: When dealing with workflows, either at design-time or run-time, it is very likely to have to take resources into account at some points. Many kinds of requirements on workflows can involve resources: they can constraints the execution of specific tasks, require global optimization, allow some flexibility or not, etc. However, resource is seldom expressed as "first class citizen" in many workflow definition languages. Hence it is difficult to design rich reasoning abilities on top of them and consequently this does not ease the development of powerful resource-aware decision support. In this paper, we propose an enhanced workflow metamodel capturing the resource dimension within both design-time and run-time dimensions. Based on this metamodel, we illustrate some interesting usage scenarios coping with design-time aspects (e.g. potential bottlenecks) and, most importantly, run-time aspects (e.g. strategies from intelligent recovery, degraded mode). The model was elicited and validated from an industrial case study which is illustrated in a simplified way.

## 1 INTRODUCTION

As business processes are getting increasingly complex, more and more organizations now turn to modelling and workflow management systems to better understand and master all the different tasks and actors involved in the implementation of their processes. At the time being, various languages and tools allow, on the one hand, to express the structuration and execution of the various tasks, as well as the constraints related to the participants. On the other hand, they provide assistance to decision-making in situations with several acceptable alternatives.

However, these solutions are still limited, especially in terms of resource management in the broadest sense, as well as for the management of unexpected events for which there is no pre-procedure defined. Indeed, the management of resources in the workflows is traditionally mainly considered under two approaches. On the one hand, flow-oriented modelling languages only take into account human resources involved as actors in the workflows. On the other hand, resource-oriented proposals neglect flow aspects of the processes. In this work, we hence propose a metamodel that incorporates a refined notion of resources and their availability in a stream-oriented model.

In order to illustrate the metamodel and its application, we use an airport management case study from the ongoing BEM project (BEM, 2013). It focuses on the landing procedure of approaching airplanes until they are fully parked, and includes alternative scenarios such as redirection to other airports and incident/accident management. Key resources to consider are related to the plane (such as fuel, schedule) and the airport (such as number of tracks, waiting levels, controllers).

The remainder of the paper is structured as follows. Section 2 delineates the research context and describes the related works in the field of workflow modelling. Section 3 presents the resource-aware workflow metamodel that was defined to overcome the limitations of existing approaches. Section 4 subsequently discusses this proposition and its application, notably with respect to the airport management example. Section 5 finally concludes this paper.

Ramdoyal R., Ponsard C., Derbali M., Schwanen G., Linden I. and Jacquet J..
A Generic Workflow Metamodel to Support Resource-aware Decision Making.
DOI: 10.5220/0004417002430250
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 243-250
ISBN: 978-989-8565-61-7
Copyright © 2013 SCITEPRESS (Science and Technology Publications, Lda.)

## 2 RESOURCE MODELLING IN WORKFLOW MODELLING LANGUAGES (WML)

Today, there are various available WMLs, that differ in the way they support the different aspects of processes (e.g, activities, roles, resource, data, etc.). Among WMLs, the most commonly used are *control-flow driven*, which focus on the activities and their flow. The resources required to actually perform the activities at run-time, when mentioned, are usually only subject to a basic description. Other languages, called *resource-driven*, consider resources as first class citizens and provide more refined description of resources and related constraints. These languages are more declarative and less commonly used by industries. In this section, we present and discuss the resource integration in representative WMLs of both families.

### 2.1 Business Process Modelling Notation (BPMN)

Currently maintained by the Object Management Group, BPMN was developed to provide a notation describing business processes understandable for both analysts and developers, while bridging the gap between their design and implementation (BPMN 1.2, 2009). The BPMN 2.0 specification (BPMN 2.0, 2011) gathers the core elements used to describe the process model, which consist in four categories. *Flow objects* define the behaviour of business process (events, activities and gateways). *Connecting Objects* (sequence flows, message flows and associations) specify the connection of the flow objects to each other or with other information. *Swimlanes and Pools* distinguish different (types of) participants in a process, either being a specific business entity (e.g. a company, a department or a team) or a more general business role. *Artefacts* allow the introduction of additional information about the process, including *data object*, *group* and *annotation*. The modelling of resources is outside the main scope of BPMN (Wohed et al., 2006), but the concepts of lanes and pools reveal the need to support at least human-resources. In addition, BPMN offers a specific resource attribute at the activity level that allows to specify who will perform (or be responsible for) the task (a person, a group or organizational unit or position).

### 2.2 Petri and Workflow Nets

First introduced in (Petri, 1962), *Petri Nets* were originally designed as a tool for modelling and analysing concurrent (distributed) systems. Their simple graphical syntax makes them very easy to use while their clearly defined formal state-based semantics supports a large number of useful and efficient analysis techniques (Murata, 1989). Such characteristics make them particularly well suited to model workflow processes (van Der Aalst, 1998). The *Workflow Nets* proposed in (van Der Aalst, 1996) are a sub-class of Petri Nets specifically defined to represent workflow processes and that simplify the modelling of workflows (van Der Aalst and van Hee, 2004). They provide dedicated graphical operators to represent common control flow structures (as the synchronization or the sequencing of activities).

Despite Petri Nets intrinsic qualities, it is very difficult, even sometimes impossible, to express some control flow structures frequently used in workflow management, namely those involving multiple instances of the same activity and those implying a *wait and see* behaviour. Likewise, the fact that transitions are only local in Petri Nets makes impossible the specification of activities that cancel others in the workflow (cancellation activities). The use of *High-level Petri Nets*, that extends Petri Nets (e.g. with colour or hierarchy), allows to overcome some of these (control-flow related) drawbacks but not all at once (van Der Aalst and van Hee, 2004).

Regarding the resources modelling, Petri nets, and thus Workflow Nets too, greatly lack syntactic support to expressivity. Indeed, a set of $n$ equivalent resources can easily be modelled by the presence of $n$ tokens in a place used as input of all the transitions requiring this type of resource. However, as soon as the resources description involves multiple attributes, availability agenda, structuration in sub-categories and so on, increasing complexity makes the model unmanageable and enhances the inadequation of the notation.

### 2.3 Yet another Workflow Language

In the mid-2000's, an international academic joint effort delivered a new language to overcome the failure of *High-level Petri Nets* to solve all the problems seen before at once (van der Aalst and ter Hofstede, 2003). The so-called YAWL (*Yet Another Workflow Language*) supports the 20 control flow patterns identified by the Workflow Patterns initiative in (van Der Aalst et al., 2003). Although inspired by Petri Nets and its extensions, YAWL is a graphical and executable language with its own syntax and semantics, and extends the class of Workflow Nets with multiple instances, composite tasks, OR-joins gateways and removal of tokens, *i.e.* cancellation (ter Hofstede and van der Aalst, 2005; ter Hofstede et al., 2010). The YAWL System built around the YAWL language is designed as a services-oriented application that offers an editor to model workflows in YAWL and an execution engine to verify and execute them. The YAWL language implements only the control flow perspective while the YAWL system supports partially those related to data and resources. Resources are linked during the modelling step to an organizational model, which specifies information about the participants, such as roles, capabilities and privileges. Initially, participants can be system actors or human resources related to activity (task) level. The data layer also contains the execution data, which includes case data and execution logs. Resources are managed during execution through the *Resource Service* provided by the YAWL system. The recent 2.3 version (The YAWL Foundation, 2012) introduces support for non human resources. In that version, similar resources can be associated in (sub)categories and availability calendars can be associated to each resource. This recent evolution enhances the need for developing resource specifications and manage events in control flow driven models. However, except the calendar, description of resources stays limited and their dynamic evolution (creation, modification, destruction) is not considered.

## 2.4 Event-driven Process Chains (EPCs)

EPCs provide a graphical business process description language introduced within the framework of ARIS (Scheer, 2000) and based on the concepts of stochastic networks and Petri nets. The control flow of the process is presented as an event-driven process chain consisting of the following elements. *Functions* model the activities of the workflow through transformations from an initial state to a resulting state. *Events* link functions together by describing the situation before and/or after their execution (pre and post conditions). *Logical connectors* connect function and events through the logical operators AND, XOR and OR. EPCs has been extended with data and organizational elements called extended event-driven process chains (eEPCs) to support data and resource aspects (Scheer, 2000). For each function, an organizational role is specified to perform this function. These roles are defined in an organizational chart of the company and are mainly dedicated to model human resources.

## 2.5 Resource-driven Framework

The alternative framework proposed in (Kumar and Wang, 2010) supports resource-driven workflow modellings, which is typically useful when multiple resources are involved in a process and may be subject to usage conflicts. This model is based on an analysis of resource flows between tasks and checks that each task will obtain its input resources. In addition, it can be used to generate a preliminary design for ad hoc-workflows. Processes modelled according this framework, on one hand, are very flexible and can be changed instantly by changing constraints. On the other hand, the framework is difficult to use in an industrial context because the model loses the expressivity of the control flow and makes it difficult to get an intuition of the system's behaviour.

## 2.6 Evaluation

Table 1 summarizes this survey of workflow modelling languages focused mainly on their resource modelling capabilities. The control-flow aspect is presented in most WMLs but lacks in the resource-driven framework because it focuses only on the dependencies between the necessary resources to execute the activities. However, each language differs in the way to describe tasks, gateways, connectors, etc. Besides, WMLs partially provide the resource aspect, except for the resource-driven framework. All other WMLs of table 1 focus on the human resources as workflow participant and include the role concept at activity level. However, such simple resource management and the recent progress of Yawl argue in favour of a framework supporting more sophisticated resources management even within control-flow driven models.

Table 1: Evaluation of WMLs.

| | **BPMN** | **Petri net** | **YAWL** | **EPC** | **Resource-driven Framework** |
|---|---|---|---|---|---|
| **Control flow** | + | + | + | + | - |
| **Data** | + | +/- | + | - | + |
| **Human resource** | + | +/- | + | +/- | + |
| **Non-human resource** | - | - | +/- | - | + |

This led us to opt for a generic workflow metamodel to deal to the observed limitations. The solution developed in this paper covers all workflow aspects (control flow, all resource type, data,...). Among other qualities, this integrated view offers efficient support to dynamic management and improve the flexibility of the workflow.

## 3 PROPOSAL: A RESOURCE-AWARE WORKFLOW METAMODEL

To answer the previously mentioned limitations, we propose a *resource-awareness* workflow metamodel presented according to two dimensions, namely (i) *Design-time versus Run-time* and (ii) *Control-flow versus Resources*. The *design-time* concerns the upstream definition of the workflows for the given environment, which notably implies defining, classifying and associating concepts such as processes, activities, events, exceptions, resources and roles. In contrast, the *run-time* relates to the downstream instantiation, execution and monitoring of these workflows. The *control-flow* aspect focuses on the structuring and succession of the activities, as well as the management of exceptions, in terms of (i) occurrence and (ii) handling. As for the *resources*, they embrace the means that can be used to achieve these activities. The proposed metamodel is subsequently described according to these two dimensions, using the wide-spectrum Generic Entity-Relationship (GER) model (Hainaut, 2005), which is typically used to describe conceptual (meta)models in various abstraction levels and paradigms. The metamodel provides the vocabulary to express both design time checks but also, and more importantly, dynamic run-time recovery strategies which we address in section 4.

### 3.1 Design-time Control-flow

Deriving from traditional approaches, modelling design-time control-flow requires to be able to specify the processes of the application domain and their decomposition into activities, as described in Figure 1(a). This aspect is typically covered during the analysis of the application domain, to understand how the given organizations are supposed to operate while managing well-known or predictable deviations from the normal modes of operation. A *process* can be composed of any number of *activities*, which can in turn be decomposed into sub-activities. An activity may be mono-instance or multi-instance. It is preceded by a junction point (Join) and followed by a separation point (Split) into any number of activities. These joins and splits may be subject to specific constraints regulating the start and the termination of the activities. The processes and activities may encounter known *exceptions*, which may result from the occurrence of a known *event*. Identifying such events can be done using standard approaches and techniques, such as the risk analysis methodologies described in (Tixier et al., 2002). For each of these events, the nature of the exception is crucial. In particular, we focus on how the resources are impacted. Specific recovery processes can be defined to handle documented exceptions, and in turn be decomposed and described.

### 3.2 Design-time Resources

Modelling design-time resources requires implies specifying all the means necessary to achieve the various processes and activities described in Section 3.1. We here use the term *resource* as defined in (Kumar and Wang, 2010), which implies that they are not limited to (human) *participants*, but also covers tools, machines, consumables and so on. Exhaustive examples of ways resources can be used in workflows are presented in (Russell et al., 2005). Resources can therefore be classified into resource types for which different properties can be defined (Figure 1(b)). The expected availability and unavailability of these resources may also be defined, as well as the possible *conflicts* between them (Figure 1(e)). Each resource may be owned by one or several *organizations* (Figure 1(d)). For human participants (users), personal and contact informations may be provided. The hierarchical structure of the organizations may be specified in terms of *groups* and *positions*. The expected availability, unavailability and reachability and the groups may also be defined. An activity may require different *roles* for its execution (Figure 2). Each role may necessitate specific prerequisite capabilities and be assigned to a specific participant or position. Roles can be organized into a given hierarchy, and an assignment order can be defined. Roles can also be forbidden to specific participants or positions.

An activity may also require the *occupation* of specific resources or types of resources (Figure 1(c)). The occupation rate and the shareability of the (types of) resources may be specified. An assignment order can be specified for the occupation and specific (types of) resources can also be forbidden. Besides, activities can impact or modify specific types of resources during their execution. Finally, the metamodel enables to express the link between *events* and *resources* at design time. This implies defining how an event may impact a resource type or modify some of its properties (similarly as the execution of an activity), but also expressing how the unavailability of resources for specific roles and occupations can translate into specific events.

### 3.3 Run-time Control-flow

Modelling run-time control-flow focusses on logging

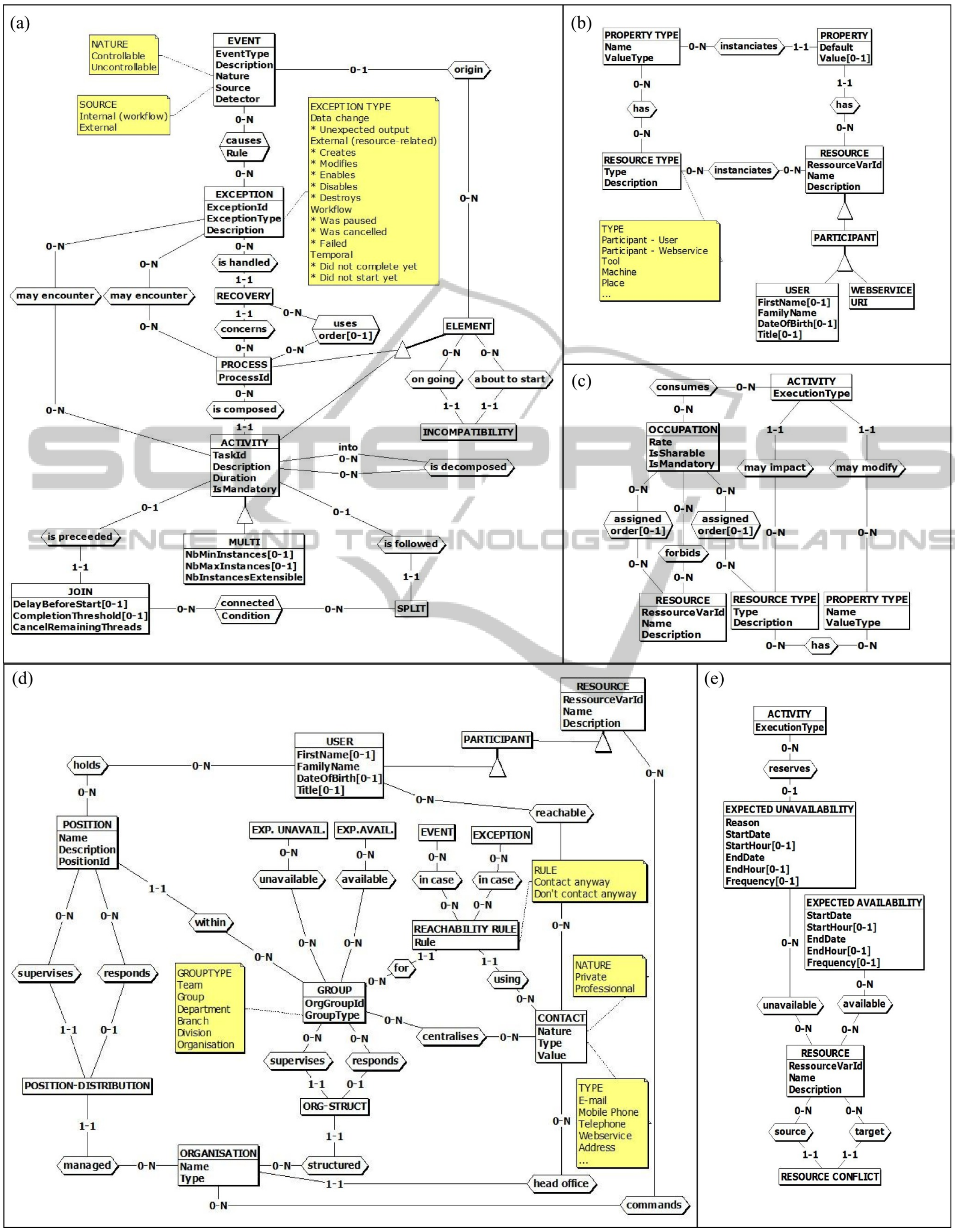

Figure 1: Metamodelling of (a) design-time control-flow, as well as resources (b) properties, (c) occupation, (d) organizations and (e) availability/conflicts.

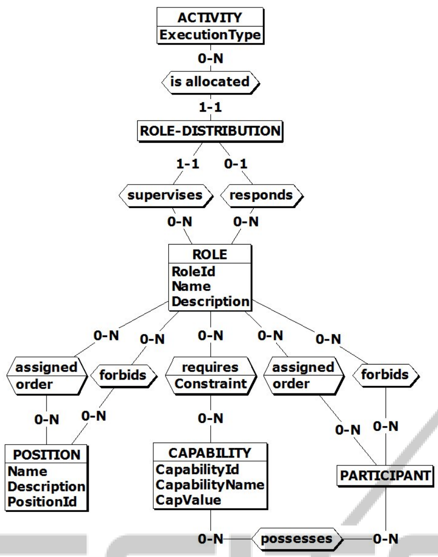

Figure 2: Design-time resources modelling in the metamodel: roles.

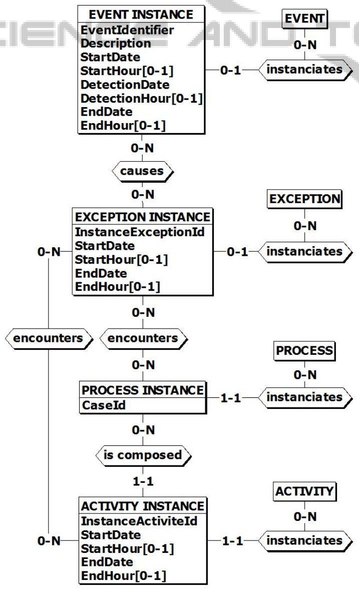

Figure 3: Run-time control-flow modelling in the metamodel.

the instances of processes, activities, events and exceptions defined in Section 3.1, as described in Figure 3. Note that it is however possible to encounter events or exceptions that have not been predicted upstream.

## 3.4 Run-time Resources

Finally, modelling run-time resources focusses on logging the real-time usage of resources specified in Section3.2 by the instances of processes and activities logged in Section 3.3, as described in Figure 4.

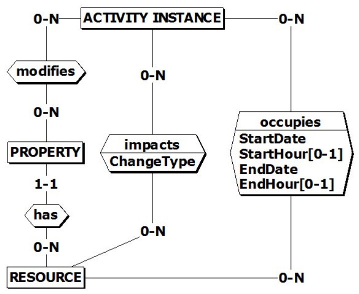

Figure 4: Run-time resources modelling in the metamodel.

## 4 DISCUSSION: USING THE METAMODEL FOR RUN-TIME DECISION MAKING

The resource perspective provided by the metamodel opens new reasoning possibilities for the system to manage processes over time and circumstances, e.g. in the context of our airport management case study. The resource characteristics captured by the model include the expected/observed availability/unavailability, the activities within workflow that relies on it or can impact it, specific constraints (exclusions, ordering, amount,...), etc. Our high-level framework for decision support (Figure 5) integrates these information and enables several resource-related reasonings such as: (i) statically analyze resource bottleneck and monitor their usage to anticipate potential risks; (ii) plan and dynamically adapt processes based on the evolving demand and offer of involved resources; (iii) capture unexpected events and reactively overcome them using non-explicitly described processes; (iv) define complex processes to manage documented exceptions for which recovery processes have not been defined (typically because the exceptions occur extremely rarely, or the recovery processes are not part of the operational activities of the organisation). This expert system relies on the detection of abnormal situations and the application of adequate corrective actions based on domain-specific and/or more generic decision rules, that can typically be expressed through the *conditions* and *rules* represented in Figure 1(a). For the sake of simplicity, we here consider basic rules of the following form:

```
IF <deviation condition>
                THEN <corrective action>
```

The airport case study is resource-limited in several ways (Figure 6), e.g. planes have a limited amount of fuel which translates into the limited time available before requiring emergency landing. Airports also have only limited infrastructure to host the planes in terms of tracks and separations

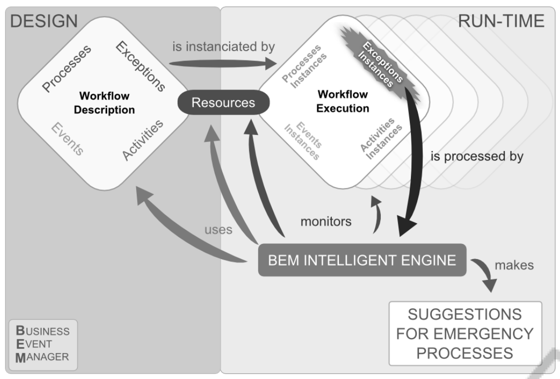

Figure 5: Overview of a Business Event Manager.

levels, which translates into a limited acceptance rate for incoming flights. Managing resources is hence important, e.g. to ensure safety (critical for the case study), minimize operation costs (hence optimizing resource allocation) or allocate the work load according to the available manpower. Rules regarding the management of runways and plane rerouting can be stated as follows:

(i) `IF <the traffic grows AND the separation level on allocated runways exceeds a given threshold AND there exists an unallocated runway> THEN <allocate extra runway>`

(ii) `IF <there is no free runway for the plane AND the plane has low fuel AND another airport is available> THEN <reroute the plane to the other airport>`

(iii) `IF <the number of used runways is smaller that a given threshold AND the traffic is not growing> THEN <close unnecessary runways>`

Such rules, balancing operational conditions and costs optimisation, can be generalised to many industrial application domains.

Regarding the management of unexpected events, it must be pointed out that in-depth design-time modelling is expensive and may lead complex and hard-to-maintain workflows, while risk analysis methodologies have their limits. In this context, the proposed approach provides dynamic mechanisms to directly tackle unexpected events with feasible resources-wise solutions. Unrealisable solutions are discarded either by default (if defined as such) or whenever available resources are incompatible with the current global state of the system (active workflow instances, allocated resources, environment characteristics...). The previously stated rules coped with dynamic variation of the related resources, and can hence be extended to deal with adverse events. For instance, regarding the management of weather conditions and rerouting:

(i) `IF <there is adverse wind on a runway> THEN <close that runway>`

(ii) `IF <there is no nearby airport available> THEN <open all available runways>`

An important discussion point relates to what extend the envisioned actions can be applied automatically given the unexpected character of what is happening. Again, we could have identified the above conditions at design time and have foreseen these in the workflow. Adding them at run-time can be done either manually or using some automatic instantiation of generic rules. For example, replacing an unavailable resource by an equivalent available resource. Looser strategies can also be designed to allow using only partly compatible resources whenever the impact is acceptable (e.g. identified degraded mode). Applying such strategies requires the greatest care as it can impact the stability of the system. This can be supported by specific analysis techniques such as model-checking. However this kind of strategy update and analysis should be seen as an intermediate and flexible adaptative maintenance on the system, hence taking place on another time scale than the system operation. It must also take into account and arbitrate the needs for pure immediate automatic system reactions versus the low-term evolution of the system workflows.

## 5 CONCLUSIONS AND PERSPECTIVES

In this paper, we presented a *resource-aware workflow metamodel* coping with the limitations of existing WMLs. Taking into accounts two critical dimensions (*design-time versus run-time* and *control-flow versus resources*), it enables to capture and express advanced characteristics of the resources that can typically be used to improve decision support. In particular, it enables to distinguish human participants from other types of resources such as consumables and to explicit the close interactions between activities and resources, as well as between events and resources. We also showed how the meta-model provides a rich semantic framework supporting rules to enable reasoning on the system evolution, beyond the classical static resource bottleneck analysis. It should be noted that a tool-supported methodology is currently being developed to exploit the metamodel. It relies on a guided questionnaire that progressively explores the various aspects of the meta-model with the stakeholders, and provides an extension to the YAWL software environment to capture the resulting specifications. The next steps of this work will consist in pushing further the evaluation of the meta-model and

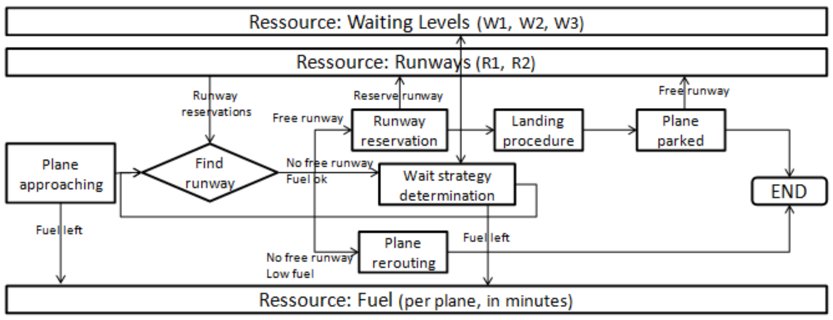

Figure 6: Airport management case study.

polishing its methodology and tool-support in order to improve the quality of the resulting specifications and tune the metamodel accordingly. On the run-time side, a run-time expert system connected to a workflow monitoring and enacting framework is currently being designed: the rule language sketched in the paper is being refined and a library of resource management patterns is being elaborated.

## ACKNOWLEDGEMENTS

This work was carried as part of the BEM project, supported by the Région Wallonne and its Plan Marshall funding (convention number 6306). We also thanks the partners of the BEM project for providing the air traffic management case study.

## REFERENCES

BEM (2013). The Business Event Manager Project (BEM) project. http://www.logisticsinwallonia.be/en/bem.

BPMN 1.2 (2009). Business Process Modeling Notation (BPMN) Version 1.2. Technical report, Object Management Group (OMG).

BPMN 2.0 (2011). Business Process Modeling Notation (BPMN) Version 2.0. Technical report, Object Management Group (OMG).

Hainaut, J.-L. (2005). The Transformational Approach to Database Engineering. In *GTTSE 2005*, volume 4143 of *LNCS*, pages 95–143. Springer.

Kumar, A. and Wang, J. (2010). A framework for designing resource-driven workflows. In *Handbook on Business Process Management 1*, International Handbooks on Information Systems, pages 419–440. Springer Berlin Heidelberg.

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. In *Proc. of the IEEE*, pages 541–580.

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

Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M., and Edmond, D. (2005). Workflow resource patterns: Identification, representation and tool support. In *CAiSE*, pages 216–232.

Scheer, A.-W. (2000). *Aris-Business Process Modeling*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition.

ter Hofstede, A. H. M. and van der Aalst, W. M. P. (2005). YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275.

ter Hofstede, A. H. M., van der Aalst, W. M. P., Adams, M., and Russell, N., editors (2010). *Modern Business Process Automation - YAWL and its Support Environment*. Springer.

The YAWL Foundation (2012). YAWL User Manual - version 2.3. Technical report, The YAWL Foundation.

Tixier, J., Dusserre, G., Salvi, O., and Gaston, D. (2002). Review of 62 Risk Analysis Methodologies of Industrial Plants. *Journal of Loss Prevention in the Process Industries*, 15(4):291 – 303.

van der Aalst, W. and ter Hofstede, A. H. M. (2003). YAWL: Yet Another Workflow Language . Technical report, Queensland University of Technology.

van Der Aalst, W. M. P. (1996). Structural Characterization of Sound Workflow Nets. Technical Report 23, Eindhoven University of Technology.

van Der Aalst, W. M. P. (1998). The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems and Computers*, 08(01):21–66.

van Der Aalst, W. M. P., Ter Hofstede, A. H. M., Kiepuszewski, B., and Barros, A. P. (2003). Workflow Patterns. *Distrib. Parallel Databases*, 14(1):5–51.

van Der Aalst, W. M. P. and van Hee, K. (2004). *Workflow Management: Models, Methods, and Systems*. The MIT Press.

Wohed, P., Dumas, M., Ter Hofstede, A. H. M., and Russell, N. (2006). On the Suitability of BPMN for Business Process Modelling. In *Proc. of BPM 2006, LNCS*, pages 161–176. Springer.