RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes*

Cristina Cabanillas, Manuel Resinas, and Antonio Ruiz-Cortés

Universidad de Sevilla, Spain
{cristinacabanillas,resinas,aruiz}@us.es

Abstract. An important task of business process design is the definition of what and how members of an organization are involved in the activities of the business processes developed within it. In this paper we analyse the capabilities of BPMN 2.0, the de-facto standard for business process modelling, in this regard. The conclusion is that, although it provides some mechanisms to assign resources to business process activities, they present several drawbacks. On the one hand, it does not provide a clear way to relate the assignment of resources with a model of the structure of the organization. On the other hand, it relies on XPath as the default language to assign resources to activities. The consequence is that it has limitations regarding the expressiveness of resource assignment expressions. Furthermore, it makes resource assignment not easy to learn and use since XPath has not been designed for that purpose. To overcome these drawbacks we introduce RAL (Resource Assignment Language), a DSL based on a well-known organizational metamodel that can be used together with BPMN 2.0. RAL provides more expressiveness to the resource assignments and it uses a high-level sintaxis defined to be used by technically unskilled users.

Keywords: resource-aware business process design, resource assignment, RAL, BPMN, workflow resource pattern.

1 Introduction

Business processes and the organization in which they are developed are closely related, since the human resources[1] of the company (i.e., its members) play an indispensable role both as supervisors of the execution of automatic activities and as performers of software-aided and/or manual activities. Consequently, an important task in business process design is the definition of which members of an organization are involved in each of the activities of the business processes developed within it.

* This work has been partially supported by the European Commission (FEDER), Spanish Government under the CICYT project SETI (TIN2009-07366); and projects THEOS (TIC-5906) and ISABEL (P07-TIC-2533) funded by the Andalusian Local Government.

[1] From now on we will use the term *resource* to refer to *human resources*.

Nowadays, most business process modelling languages provide some mechanism to carry out such a task. In this work we focus on BPMN 2.0 because it is the current standard notation for business process modelling. We have studied its capabilities to manage resources in business process models and we have realized that, although the graphical representation of resource assignments is not possible in BPMN, it does provide a textual way to assign resources to the activities of the process models. Specifically, it provides two different methods, one focused on selecting resources of a concrete type (e.g. a role or a group) and applying filters over that type to decide the potential performers of the activity, and another open to allow free assignments on any basis. In both cases, it relies on XPath[2] as the default language to either define filters or assignments. However, these methods present several drawbacks regarding expressiveness, relation with the organizational structure and ease of use.

As far as expressiveness is concerned, sometimes the assignment of the resources that can do a certain activity is quite straightforward, e.g., "Activity *Design process* must be performed by a business process analyst". However, it is not hard to find assignments that are more complex to express. For instance, "Activity *Supervise Code* must be performed by an expertised technician (with at least three years of experience) or by a consultant". In this regard, Russell et al. have described a set of *workflow resource patterns* that intend to capture the various ways in which resources are represented and utilised in workflows [1]. In particular, the *creation patterns* focus on different ways resources can be assigned to activities and constitute the main set of workflow resource patterns expressing things configurable at the level of process models, such as for instance "Activity *Deploy Application* must be undertaken by someone that reports work to the *Project Manager*, preferably the person that carried out activity *Supervise Code*". Unfortunately, the use of XPath as the default language limits the expressiveness to specify resource assignment expressions, as detailed in Section 3.

As can be seen from the previous examples, relating the organizational structure with the process models is necessary in order to be able to deal with some of these patterns. Besides being unable to express such type of constraints, the lack of consideration of the organizational structure regarding resource assignment may cause execution problems such as delays and/or blocks. For instance, two parallel activities could be associated with the same role, meaning that persons playing that role must perform them at run time. If only one person of the organization has that role, there may be delays in the process execution. This problem could be solved with different resource management. However, if the process model is not explicitly related to a model of the structure of the organization, which is the case of BPMN, it is much harder to analyse and detect this kind of situations.

Finally, one of the goals of BPMN is to provide a notation that is understandable by non-technical users, allowing to reduce the gap between business and IT. However, XPath is a language oriented exclusively to technical users and it has a very different purpose than to assign resources to activities. This makes

[2] `http://www.w3.org/TR/xpath20/`

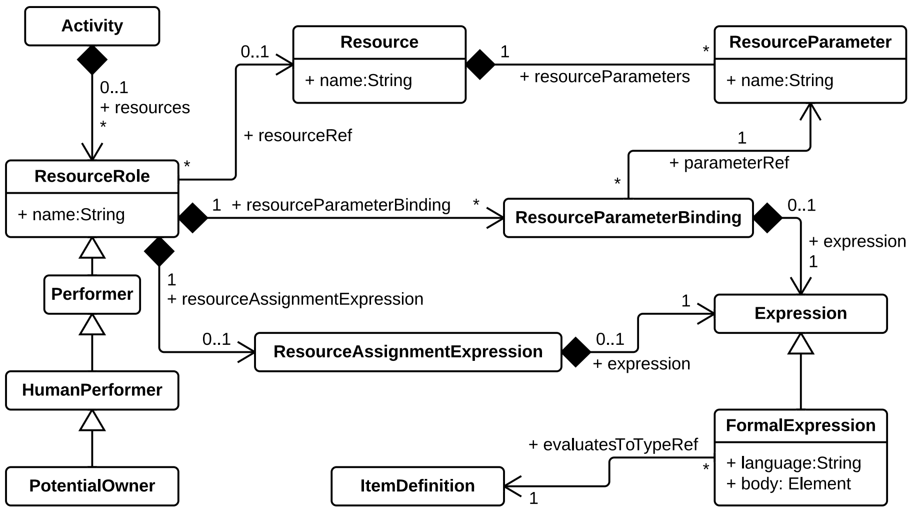


Fig. 1. Excerpt of the BPMN 2.0 metamodel regarding resource assignment [2]

it certainly hard for a non-technical user to understand such a type of resource assignment.

We have defined a Domain Specific Language (DSL) called RAL (Resource Assignment Language) with the aim of easing the way resources are assigned in BPMN, while providing high expressiveness due to its basis on a well-known organizational metamodel [1]. In this paper we will explain what can be expressed with RAL and how it can be used inside of BPMN 2.0.

Section 2 contains a detailed explanation of how BPMN 2.0 allows resource assignment in business process models. In Section 3 we present RAL. Section 4 shows the expressiveness of RAL by applying it to some creation patterns and to an example use case. Some related work can be found in Section 5, and a set of conclusions are presented in Section 6.

2 Resource Management Capabilities of BPMN 2.0

BPMN is the de-facto standard for business process modelling. It has been improved in its current version (2.0) as for the assignment of resources to activities of a business process [2]. However, the definition of *resource* BPMN 2.0 provides and the use of this term are still a little imprecise and hard to use. On the one hand, it allows the definition of elements of type *Resource*, but resource types are not set (so a resource can be anything, from a person to an organization), no relationships can be established between them and there is not a metamodel supporting them. On the other hand, the procedure proposed by BPMN is not oriented to modellers without technical skills, since the default language to define resource assignment expressions is XPath, which is far from easy-to-use.

Figure 1 shows an excerpt of the BPMN 2.0 metamodel regarding the assignment of resources to activities [2]. Each activity can have zero or more instances of *ResourceRole* assigned, which can be seen as potential performers or potential resources responsible for the activity at run time (class *PotentialOwner*). The metamodel contains two alternatives to assign the so-called potential owners.

Queries over a Specific Resource Type. As stated in [2], "a *Resource* can be *Human Resources* as well as any other resource assigned to activities during process execution time. The definition of a resource is *abstract* [...]". The BPMN specification indicates that *the name* of the resource type we want to assign to an activity must be set in class *Resource*, e.g., a specific role. We can then configure the assignment giving values to the resource parameters, such as country or age, by means of class *ResourceParameterBinding*. This class will contain an *Expression* that defines constraints on the values of the parameters to reduce the number of potential owners. Class *ResourceParameterBinding* can only be used if in conjunction with *Resource*.
In order to define the filtering expression BPMN proposes by default the use of XPath. The language has been extended to provide functions that ease some tasks such as reading information from data objects connected to the activities of the process. A brief example of this resource assignment method is shown in [2]. As can be deduced from the XML code of the example, expressing queries this way may become quite complicated and, besides, although the name of the resource type is textually specified, the process actually knows nothing about what type of resource it is (i.e., it could be a role, a group, etcetera), so the actual resource type is something transparent to the process.

Free Resource Assignment. BPMN allows less restrictive resource assignment as well, permitting to write any XPath expression to define the assignment by means of class *ResourceAssignmentExpression*. In this case, the XPath expression does not have to be stuck to a previously fixed resource type. This total freedom may be positive because no constraints are set beforehand but, at the same time, it makes it difficult for users not familiarized with XPath to define complex resource assignments in an easy and high-level way. We remind the reader that the main goal of BPMN is to allow non-technical users to design or, at least understand business process models. From this perspective, we believe the current resource assignment language provided by BPMN is not the best option.

It is important to stress that the two methods are incompatible with each other, i.e., the selection of potential owners is made either with the mechanism based on *Resource* or with a *ResourceAssignmentExpression*. Our proposal constitutes an alternative to XPath that must be used in the second resource assignment method aforementioned.

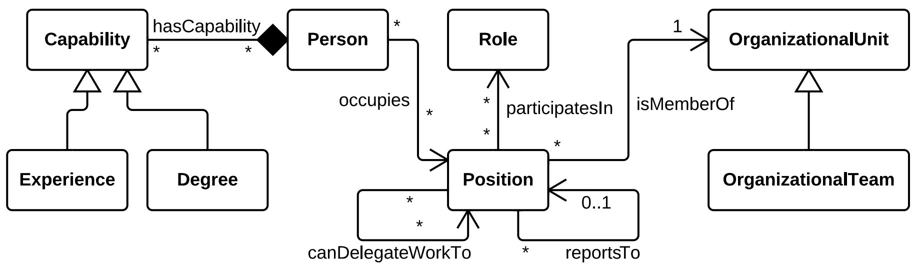


Fig. 2. Excerpt of the organizational metamodel described by Russel et al. [1]

3 RAL (Resource Assignment Language)

RAL is a DSL developed to ease the assignment of resources to the business process activities. It uses the entities and relationships defined by Russell et al. in the organizational metamodel shown in Figure 2 to define the way assignment expressions can be built. As depicted in the figure, the organizational metamodel basically consists of persons, positions, roles and organizational units. A person can have a set of capabilities, such as his/her professional experience. The metamodel is extensible to include new capabilities. Each person occupies one or more positions within an organization. In turn, a position can participate in several roles and belong to an organizational unit, which can be, for instance, an organizational team. Some relationships between positions are also established.

RAL expressions should be placed in class *FormalExpression* of the BPMN metamodel (cf. Figure 1), setting attribute *language* to RAL and writing the RAL expression in attribute *body*. As described below, RAL allows expressing from simple assignments based on a specific person or role to assignments as complex as desired by means of the compound expressions. Its EBNF notation is shown in Language 1. We next explain RAL expressions, using the term *group resource* to refer to anything but persons, i.e., positions, roles and organizational units. Persons are sometimes called *individual resources*.

Expression *IS PersonConstraint* allows expressing that an activity must be performed by someone indicated in a *PersonConstraint*: (i) a specific person; (ii) the person who performed another activity; or (iii) the person indicated in a data field.

HAS GroupResourceType groupResourceName allows assigning an activity to a given group resource, or to one read from a field of a data object.

SHARES Amount GroupResourceType WITH PersonConstraint is used to assign persons that share some or all position(s), role(s) or organizational unit(s) with the person indicated in a *PersonConstraint*.

Expression *HAS CAPABILITY CapabilityConstraint* allows expressing constraints based on personal capabilities, such as *years of experience* or *reputation*[3]. These constraints may consist of the existence of certain capability

[3] We can also consider issues such as *age* or *origin* capabilities.

Language 1. Expression assignment EBNF language definition

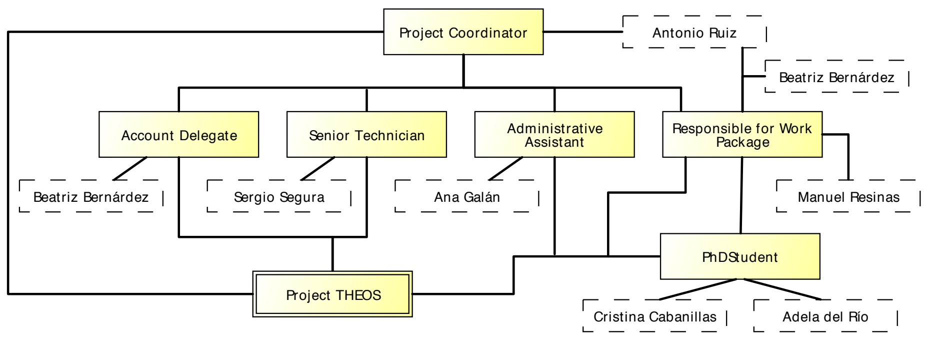
```
Expression := IS PersonConstraint
     | HAS GroupResourceType GroupResourceConstraint
     | SHARES Amount GroupResourceType WITH PersonConstraint
     | HAS CAPABILITY CapabilityConstraint
     | IS ASSIGNMENT IN ACTIVITY activityName
     | RelationshipExpression
     | CompoundExpression

RelationshipExpression := ReportExpression
             | DelegateExpression

ReportExpression := REPORTS TO PositionConstraint Depth
           | IS Depth REPORTED BY PositionConstraint

DelegateExpression := CAN DELEGATE WORK TO PositionConstraint
            | CAN HAVE WORK DELEGATED BY PositionConstraint

CompoundExpression := NOT (Expression)
             | (Expression) OR (Expression)
             | (Expression) AND (Expression)
             | (Expression) AND IF POSSIBLE (Expression)

PersonConstraint := personName
          | PERSON IN DATA FIELD dataObject.fieldName
          | PERSON WHO DID ACTIVITY activityName

GroupResourceConstraint := groupResourceName
              | IN DATA FIELD dataObject.fieldName

CapabilityConstraint := capabilityName
               | CapabilityRestriction

PositionConstraint := POSITION namePosition
             | POSITION OF PersonConstraint

Amount := SOME         GroupResourceType := POSITION
     | ALL                         | ROLE
                                 | UNIT
Depth := DIRECTLY
        | λ
```

or of the holding of certain condition on the value of a capability. We are not detailing the *CapabilityRestriction* for space reasons, since it is based on mathematical and logical operators and its use is easily understandable.

Expression *IS ASSIGNMENT IN ACTIVITY activityName* is used to indicate that an activity has the same RAL expression as another activity. This avoids having to re-write several times the same assignment, at the same time as it helps saving time and effort and prevents typing errors.

RelationshipExpression is set to allow expressing constraints such as "Activity *Fill Travel Authorization* must be performed by someone that reports to the *Project Coordinator*", according to the relationships between positions depicted in Figure 2.

CompoundExpression allows expressing combination and negation of the aforementioned expressions. Furthermore, the conditional expression *AND IF POSSIBLE* has been included to let the modeller express preferences/priorities. For instance, by stating that, *if possible*, an activity has to be carried



Position	Role
Project Coordinator	Responsible
	Account Administrator
	Resource Manager
	Doctoral Thesis Advisor
	Researcher
Responsible for Work Package	Responsible
	Researcher
	Doctoral Thesis Advisor

Position	Role
PhD Student	Research Assistant
Senior Technician	Responsible
Account Delegate	Account Administrator
Administrative Assistant	Clerk

Fig. 3. Excerpt of the organizational model of ISA Group from a project perspective

out by certain role, we are meaning that that is the first assignment we have to try when actually allocating the activity to an individual resource (at run time). In case preferences are not fulfilled, they are just ignored.

Note that some of these expressions could be analysed at design time and a set of potential owners would be obtained (cf. Section 2), from which the actual owner/performer and, thus, the person in charge of the activity, would be selected at run time. However, sometimes the allocation has to be directly deferred until run time because some running information is required and it is missing at design time, e.g. those assignments depending on data field values.

It is important to notice that we have restricted RAL to expressions involving *a single instance* of a business process. The history of resource allocations and past process executions are not considered for now. Some specific examples of the language usage are described in Section 4 with the help of a use case.

4 Application of RAL. Examples

Imagine we belong to an organization with the structure shown in Figure 3. This figure contains an instantiation of the organizational metamodel described in Section 3. Specifically, it is an excerpt of the *ISA Research Group* of the University of Seville from a research project perspective. There are six positions (Project Coordinator, Account Delegate, Senior Technician, Administrative Assistant, Responsible for Work Package and PhD Student) that are members of one organizational unit (Project THEOS), and seven persons occupying these positions. Each position of the model can delegate work to any inferior position

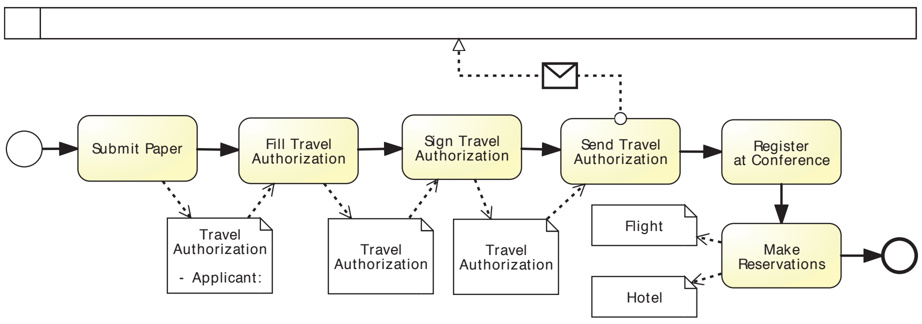


Fig. 4. Simplified process for Conference Travel Management

and report work to its immediately upper position. The relationship *participatesIn* of the metamodel is summarized in a table. For instance, individual *Beatriz Bernárdez* belongs to positions *Responsible for Work Package* and *Account delegate*. As a responsible for work package she has three roles: *Responsible*, *Researcher* and *Doctoral Thesis Advisor*. On the other hand, her other position gives her the role *Account Administrator*. Both positions are in turn linked to the *Project THEOS*, which is an organizational unit. A table with the *hasCapability* relationship should also be specified.

The business process in the BPMN model of Figure 4 may represent some work developed in our organization. The figure illustrates a simplified version of the process to manage the trip to a conference (according to the rules of the University of Seville), from the submission of the final version of an accepted paper to the booking of the transport tickets and the accommodation. It starts with the submission of the Camera Ready version of a paper, and it continues when one of the authors fills up a form requesting for authorization both to travel to the venue place and to take the funds from some funding source. This authorization must be approved by some person in charge of account management related to the applicant. The travel authorization is sent for revision to an external entity, where someone might sign the document. Then, the attendant must register at the conference and make the appropriate reservations.

We are going to show examples of resource assignments to the activities of the business process in Figure 4 using RAL language. We are using as example some workflow resource patterns. Specifically, the patterns we are most interested in are the creation patterns, as they are mainly focused on information that must/can be specified at design time, as is the case of RAL.

Direct Allocation: The ability to specify at design time the identity of the resource that will execute a task. For instance, the *Sign Travel Authorization* task must only be undertaken by *Antonio*:

```
Sign Travel Authorization: IS Antonio
```

Role-Based Allocation: The ability to specify that a task can only be executed by resources with a given role. For instance, instances of the *Fill Travel Authorization* task must be executed by a *Research Assistant*:

```
Fill Travel Authorization: HAS ROLE ResearchAssistant
```

Deferred Allocation: The ability to defer specifying the identity of the performer of a task until run time. For instance, during execution of the process, instances of the *Send Travel Authorization* task will be executed by the person named in the resource field *Applicant* of data object *Travel Authorization*:

```
Send Travel Authorization:
    IS PERSON IN DATA FIELD TravelAuthorization.Applicant
```

Authorization: The ability to specify the range of resources that are authorized to execute a task. For instance, only a *Researcher* and a *Research Assistant* are authorized to execute instances of the *Submit Paper* task:

```
Submit Paper:
    (HAS ROLE Researcher) OR (HAS ROLE ResearchAssistant)
```

Separation of Duties: The ability to specify that two tasks must be allocated to different resources in a given workflow case. For instance, instances of the *Sign Travel Authorization* task must be allocated to a different person from that who executed the *Fill Travel Authorization* task:

```
Fill Travel Authorization:
    NOT (IS PERSON WHO DID ACTIVITY SignTravelAuthorization)
Sign Travel Authorization:
    NOT (IS PERSON WHO DID ACTIVITY FillTravelAuthorization)
```

In this case, we assume at design time we do not know the real execution order of the activities and, thus, we set the constraint in both of them.

Case Handling: The ability to allocate the activities within a given workflow case to the same resource. For instance, all tasks assigned to position *PhD Student* are allocated to the same person.

```
Assigned to some activities: (HAS POSITION PhDStudent) AND
    (IS PERSON WHO DID ACTIVITY FillTravelAuthorization)
```

The second part of the composition is not necessary for the first task that has been assigned the position *PhdStudent*. Please, note that the example exposed is this case is fictitious and will not be considered later in this paper.

Retain Familiar: Where several resources are available to undertake an activity, the ability to allocate an activity within a given workflow case to the same resource that undertook a preceding activity. For instance, any *PhD Student* available can undertake the *Register at Conference* task, but it should be allocated to the same person that undertook the *Submit Paper* task.

```
Register at Conference: (HAS POSITION PhDStudent)
    AND IF POSSIBLE (IS PERSON WHO DID ACTIVITY SubmitPaper)
```

Capability-based Allocation: The ability to offer or allocate instances of a task to resources based on their specific capabilities. For instance, instances of the *Submit Paper* task must be allocated to someone with a degree:

```
Submit Paper: HAS CAPABILITY Degree
```

Organizational Allocation: The ability to offer or allocate instances of a task to resources based on their position within the organization and their relationship with other resources. For instance, the *Sign Travel Authorization* task must be allocated to someone that is reported by (the position of) the person that undertook the *Fill Travel Authorization* task:

```
Sign Travel Authorization: IS REPORTED BY POSITION OF
    PERSON WHO DID ACTIVITY FillTravelAuthorization
```

Please note that we have not included workflow resource pattern *history-based allocation* because we are focused on a single business process instance and disregard previous executions, as aforementioned. Pattern *automatic execution* is not included either because no resource assignment is required in this case.

The final resource assignment of every activity of the business process in Figure 4 are those depicted in Figure 5. Note that the last assignment does not belong to the previous examples and has been specified here to show how Language 1 allows expressing quite complex constraints.

5 Related Work

The need of including organizational aspects in business process design can be seen in [3], where Künzle et al. present a set of challenges that should be addressed to make business processes both data-aware and resource-aware.

In 1999, Bertino et al. defined a language to express constraints in role-based and user-based assignments to the tasks of a workflow [4]. They got to check whether the configured assignments were possible at runtime and to plan possible resource allocation based on the assignments. They considered also dynamic aspects for these checks. The language was based on functions and was more complex and hard to use than RAL, since its goal was wider.

In 2007, Russell et al. described a set of workflow resource patterns aimed at explaining the requirements for resource management in workflow environments [1]. They analysed the support provided by some workflow tools, BPMN 1.0 among others, but they did not provide a specific way to assign resources to workflow activities. These patterns were used by Grosskopt to analyse the ability of BPMN 1.0 again and to propose solutions to extend the number of patterns addressed by the standard [5]. However, he did not consider nor included organizational information in the process models and, hence, he could not establish assignments on the basis of the resource capabilities or their relationships.

```
Submit Paper:
((HAS ROLE Researcher) OR (HAS ROLE ResearchAssistant))
AND (HAS CAPABILITY degree)

Fill Travel Authorization:
(HAS ROLE ResearchAssistant) AND
(NOT (IS PERSON WHO DID ACTIVITY SignTravelAuthorization))

Sign Travel Authorization:
(IS Antonio) AND ((NOT(IS PERSON WHO DID ACTIVITY
FillTravelAuthorization)) AND (IS REPORTED BY POSITION OF PERSON
WHO DID ACTIVITY FillTravelAuthorization))

Send Travel Authorization:
IS PERSON IN DATA FIELD TravelAuthorization.Applicant

Register at Conference:
(HAS POSITION PhDStudent) AND IF POSSIBLE (IS PERSON WHO DID
ACTIVITY SubmitPaper)

Make Reservations:
(NOT (IS Antonio)) AND ((SHARES SOME ROLE WITH Antonio)
OR (HAS ROLE ResearchAssistant))
```

Fig. 5. Resource assignments of the process activities in Figure 4

During 2008, Meyer worked on the extension of BPMN 1.1 to manage resource allocation in business process models and he presented the results in his Master's Thesis [6]. He revised the metamodel and task lifecycle of BPMN and proposed a formal representation of the resource perspective, together with a prototypical implementation for Oryx[4].

In 2009, Awad et al. used the workflow resource patterns again as a reference framework to study the resource management in BPMN 1.2 and proposed a metamodel extension [7]. They focused on the creation patterns but, unlike our approach, they played with swimlanes by giving specific meaning to lanes, so the process models grew as more roles were involved in the processes. Furthermore, they did not consider the organizational structure and proposed OCL[5] as constraints language. They extended Oryx with a prototype that included graphical representation for the creation patterns, but we believe defining new constraints is very complex due to the use of OCL.

To the best of our knowledge, there is not yet an approach that tries to improve resource management in BPMN 2.0 without changing its metamodel and oriented to users technically unskilled.

[4] `http://bpt.hpi.uni-potsdam.de/Oryx/`
[5] `http://www.omg.org/spec/OCL/`

6 Conclusions and Future Work

In this paper we have explained the current mechanism BPMN 2.0 proposes to assign resources to the activities of a business process, concluding that:

– It allows expressing quite a lot of constraints regarding resources, but the use of XPath makes it problematic the expression of constraints containing conjunctions, disjunctions and/or negations referring to resource types.
– In the current approach the process model is always kept out of the organizational structure of the company, so it does not know about roles, positions or persons, and, hence, assignments considering relationships between the potential owners cannot be made. That may be the reason why most of the tools for business process execution (e.g., jBPM, Activiti) use only resource assignments based on individual resources or groups.
– Its basis on XPath also makes it difficult for users with no technical knowledge about coding to learn how to work with resource assignments in BPMN model activities. A higher-level user-oriented language would be useful.

We have intended to overcome these three drawbacks of BPMN with RAL, by providing a notation close to natural language, and expressive enough to build complex assignments considering both the business process and the organizational model. In the future we plan to define a graphical notation for RAL and we will explain how we have managed to analyse resource assignments and extract useful information from resource-aware business process models by mapping RAL into an OWL ontology.

References

1. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
2. "Bpmn 2.0," recommendation, OMG (2011)
3. Künzle, V., Reichert, M.: Integrating Users in Object-Aware Process Management Systems: Issues and Challenges. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 29–41. Springer, Heidelberg (2010)
4. Bertino, E., Ferrari, E., Atluri, V.: The specification and enforcement of authorization constraints in workflow management systems. ACM Trans. Inf. Syst. Secur. 2, 65–104 (1999)
5. Grosskopf, A.: An extended resource information layer for bpmn. tech. rep., BPT (2007)
6. Meyer, A.: Resource perspective in bpmn - extending bpmn to support resource management and planning. Master's thesis, Hasso Plattner Institute, Potsdam, Germany (2009)
7. Awad, A., Grosskopf, A., Meyer, A., Weske, M.: Enabling resource assignment constraints in bpmn. tech. rep., BPT (2009)