Designing Business Processes with History-Aware Resource Assignments*

Cristina Cabanillas, Manuel Resinas, and Antonio Ruiz-Cortés

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

Abstract. Human resources are actively involved in business process management (BPM), due to their participation in the execution of the work developed within business process (BP) activities. They, thus, constitute a crucial aspect in BP design. Different approaches have been recently introduced aiming at extending existing BP modelling notations to improve their capabilities for human resource management. However, the scope of the proposals is usually quite limited, and most of them provide ad-hoc solutions for specific scenarios. Resource Assignment Language (RAL) was developed just to overcome such shortcomings, being independent of the modelling notation in which it is used, and providing interesting resource analysis mechanisms. Still, RAL is currently focused on a single BP instance and, thus, resource assignments cannot contain constraints between two process instances. In this paper, we introduce a complete (i.e. syntactical and semantical) extension for RAL to provide it with history-aware expressions. These expressions will, in turn, be able to be automatically resolved and analysed along with the other RAL expressions, thanks to RAL's semantics based on Description Logics (DLs).

Keywords: Human resource management, history-based distribution, RAL, resource-aware business process design, design-time business process analysis.

1 Introduction

Human resource management is one of the key aspects to consider when designing a BP, since the participation of people drives the execution of the processes carried out in an organization. Therefore, if we want everything to work properly and efficiently at run time, the design of an appropriate distribution of the work among the members of an organization is crucial.

As a proof of that, lots of approaches dealing with issues related to human resource management have been introduced in the recent years, and current Business Process Management Systems (BPMSs) are increasingly concerned about improving the support they provide for that purpose. Several proposals address

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

M. La Rosa and P. Soffer (Eds.): BPM 2012 Workshops, LNBIP 132, pp. 101–112, 2013.

the extension of Business Process Modelling Notation (BPMN) to enhance the resource perspective [1,2]; others extend Unified Modelling Language (UML) to specify and check well-known rules such as Segregation of Duties (SoD) and Binding of Duties (BoD) [3]. New resource assignment languages that can be used in combination with different process modelling notations have also been introduced [4,5]. However, in spite of the large amount of solutions, most of them are ad-hoc, their scope is quite limited, or they are just incomplete, meaning that they focus on specific aspects and leave others aside.

From the existing approaches, we are going to focus on RAL, a Domain Specific Language (DSL) to define resource assignment expressions in BP activities, aimed at exceeding the scope of similar approaches [5]. It has notable advantages, to be named: (i) it offers a wide collection of resource assignment expressions, (ii) it is not designed for a concrete BP modelling notation, and (iii) it provides automated analysis capabilities derived from its formal semantics based on DLs. Section 2 contains background on RAL. Nonetheless, the current scope of the language is restricted to a single BP instance, without considering past information in the resource assignments. In particular, RAL covers ten out the eleven Creation Patterns belonging to the Workflow Resource Patterns (WRPs) described by Russell et al. [6], but support for the following pattern referred to history information is currently missing:

History-Based Distribution: The ability to distribute work items to resources on the basis of their previous execution history.

We consider this an important limitation of the language, since this pattern is already supported by some other proposals, and there are constraints such as the aforementioned SoD and BoD that sometimes need information from previous BP executions (e.g. static SoD [3]). RAL's specification and semantics should, thus, be extended to include this pattern. In this paper, we introduce such an extension. As a result, we provide RAL with the required elements to specify that: (i) an activity has to be performed by the person that executed a certain activity in a previous instance of the BP; and (ii) an activity cannot be undertaken by somebody that has participated in the process in instances run during the last week. We wanted to keep RAL expressive enough to enable the specification of assignments that are likely to be used in organizations, so we have introduced history-aware assignment expressions both at activity level and at process level (cf. Section 3 for details). Besides that, in Section 4 we describe the DL-based semantics of the new RAL expressions, as well as the mechanism to automatically resolve such expressions to obtain the set of potential performers of an activity. Only one of the candidates might be later allocated to the task at run time. Finally, some related work is briefly introduced in Section 5, and the paper ends up with some conclusions and future work in Section 6.

2 Background

The work we present in this paper is the continuation of previous work on the improvement of resource management in BPs. A study we carried out about the

features provided by BPMN (and other BP modelling notations) as for resource management, revealed that the capabilities offered by current notations are not sufficient to cater for all the resource management needs of an organization. This impelled us to develop a language that allowed an easier definition of resource assignments in BP models, while keeping expressiveness in the collection of assignments that could be specified [5]. The result was RAL, a DSL specifically developed to express resource assignments in BP activities overcoming some drawbacks present in other existing approaches [1,2,3]. RAL expressions cover from simple assignments of activities to specific individuals of the company, to complex assignments containing (access-control) constraints between activities, as well as compound expressions. RAL's syntax, quite similar to natural language, increases its understandability, as shown in the following examples:

```
RAL 1: IS Samuel
RAL 2: NOT (IS PERSON WHO DID ACTIVITY CreateResolutionProposal)
RAL 3:(HAS ROLE DocumentWriter) OR (HAS POSITION ACDocumentSigner)
```

Besides, the language was equipped with formal semantics based on DLs, which enabled the design-time analysis of resource assignment expressions by using DL reasoners existing in the market [7]. This allowed us to infer information automatically from *RAL-aware BP models*, i.e. models with RAL expressions associated with the BP activities, such as (i) the potential performers of each activity; or (ii) the potential set of activities each person of an organization can be allocated at run time. As a proof of concept, we developed RAL Solver, a plug-in for Oryx [8] that emerged both to test the use of RAL expressions in BP models, and the benefits of its DL-based semantics to analyse RAL-aware BP models at design-time [7,9].

Some benefits of design-time resource-related analysis are that it informs the company about the possible workload of its employees, and warns about potential allocation problems that may arise at run time. Furthermore, it eases the detection of inconsistencies between the resource assignments associated to the activities of a BP model and the structure of the organization where it is used, e.g. non-existent roles or persons.

In the rest of this paper, we introduce an extension for RAL to deal with history information in resource assignment expressions. In particular, we have added some new expressions to the specification of the language, and we have mapped them into DLs in order to be able to automatically resolve them, and so take them into consideration along with the rest of RAL expressions.

3 Extending RAL's Specification to Support History-Based Distribution

We have extended RAL expression `IS PERSON WHO DID ACTIVITY activityName` (line 11 in Language 1), which stated that an activity had to be allocated to the same person that had performed another activity (assuming the same instance

Language 1. EBNF specification for RAL's new expressions

```
1  Expression := IS PersonConstraint
2       | HAS GroupResourceType GroupResourceConstraint
3       | SHARES Amount GroupResourceType WITH PersonConstraint
4       | HAS CAPABILITY CapabilityConstraint
5       | IS ASSIGNMENT IN ACTIVITY activityName
6       | RelationshipExpression
7       | CompoundExpression
8
9  PersonConstraint := personName
10      | PERSON IN DATA FIELD dataObject.fieldName
11      | PERSON WHO DID ACTIVITY activityName [HistoryExpression]
12      | PERSON WHO HAS PARTICIPATED IN BPHistoryExpression
13
14 HistoryExpression := IN CURRENT INSTANCE
15      | IN ANY INSTANCE
16      | IN ANOTHER INSTANCE
17      | FROM startDate TO endDate
18
19 BPHistoryExpression := CURRENT PROCESS INSTANCE
20      | ANY PROCESS INSTANCE
21      | ANOTHER PROCESS INSTANCE
22      | A PROCESS INSTANCE BETWEEN startDate AND endDate
```

of the BP), to deal with the history-based distribution pattern (lines 14 to 17). The extension consists of spreading the scope to other BP instances, allowing the definition of constraints about the instance in which the referenced activity was executed, specifically:

- *Line 14.* The same process instance currently running. This is the option selected by default in case no *HistoryExpression* is specified.
- *Line 15.* Any instance of the process (including the ongoing one).
- *Line 16.* Any previous process instance (excluding the ongoing one).
- *Line 17.* Those process instances in which the activity has been completed between two given dates (regardless of whether the process instance itself is over or not).

Furthermore, based on the same constraints, we have introduced a new expression in the language (line 12). In it, we do not specify the activity whose performer is referenced, but the BP instance in which he/she has participated, i.e. he/she has undertaken some activity in that process instance. It is a more generic expression but can be useful not to limit so much the scope of the constraint. This way, lines 12 and 19-22 state that an activity has to be performed by somebody that executed *some* activity in (i) the ongoing process instance, (ii) any process instance, (iii) a previous process instance, and (iv) any process instance, provided that the activity was completed between two given dates (similarly to expression in line 17, it is not required the whole process instance being over by that moment).

Besides, we remind that RAL has a negation operator (NOT) we could use to define the opposite expressions, e.g. to state that an activity cannot be performed by the person that undertook another activity at any time in the past.

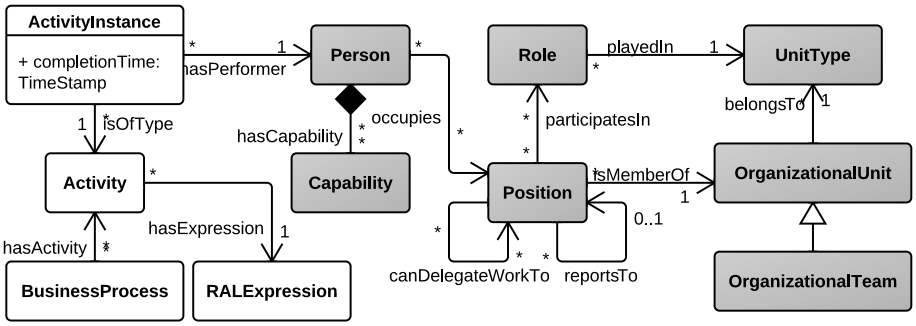


Fig. 1. Meta model with the required elements for history-aware RAL expressions

The organizational meta model on which RAL is based does not need to be modified in order to apply this extension. However, we have to provide a way to store the information required for the history-aware expressions. In Figure 1, classes related to organizational aspects are shown in gray. They correspond to the excerpt of the organizational meta model described by Russell et al. [10] RAL has always used. New elements to deal with history information are introduced in white. These elements contain the execution information necessary to resolve the new expressions. Specifically, a *BusinessProcess* has a set of *Activities*, which have associated a *RALExpression* indicating the resources allowed to perform the task at run time (i.e. potential performers of the activity). For each BP instance, zero, one or more instances of its activities can be executed. Each *ActivityInstance* (a.k.a. *work item*) can have a different *actual performer*, as long as that person meets the conditions stated by the resource assignment expression. The completion time of each activity instance is recorded, so that RAL expressions in lines 17 and 22 can be resolved. The set of work items undertaken by a single person constitute his/her execution history. The meta model shown in Figure 1, thus, contains all the information required to use the previous version of RAL and the new history-aware expressions.

3.1 Application Example

We are going to use the same use case we used when we first introduced RAL to exemplify the use of the new expressions [5].

The BP represented by the BPMN model in Figure 2 illustrates a simplified version of the process to manage the trip a conference (according to the rules of the University of Seville). It starts with the submission of the Camera Ready version of a paper accepted for publication at the conference. Then, one of the authors fills in a form requesting for authorization both to travel to the venue place and to take the funds from some funding source. This document must be approved by some person allowed to authorize the applicant to attend the conference and take funds from the project specified in the authorization form, e.g. the project coordinator. The signed document is sent for revision to an

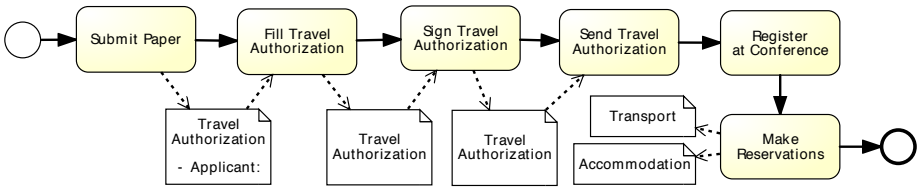
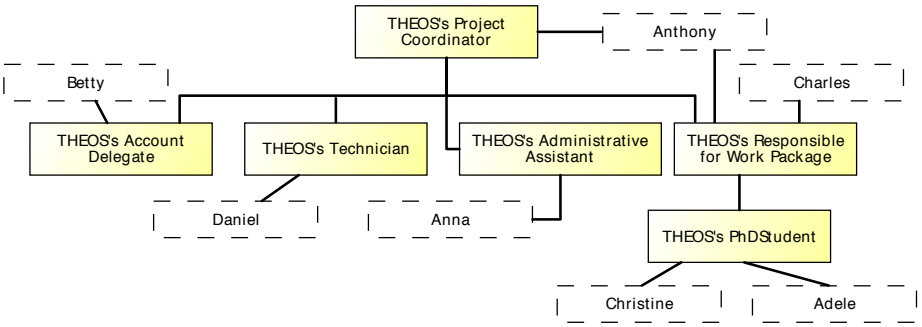


Fig. 2. Business process for conference trip management



Position	Role
THEOS's Project Coordinator	Project Coordinator
	Project's Account Administrator
	Project's Resource Manager
THEOS's Responsible for Work Package	Project's Responsible for Work Package
	Project's Researcher
THEOS's PhD Student	Project's PhD Student
…	…

Fig. 3. Excerpt of the organizational model of ISA Group for project THEOS

external entity, where someone evaluates the request. If it is approved, one author of the paper must register at the conference and make the reservations required.

Imagine that the previous BP is used in an organization with the structure shown in Figure 3. It is actually an excerpt of the structure of the ISA research group of the University of Seville with regard to a research project called *THEOS* (class *OrganizationalUnit* of the meta model in Figure 1). It has six positions and seven persons occupying them. Each position of the model can delegate work to any inferior position and report work to its immediately upper position. The relationship *participatesIn* of the organizational meta model is summarized in the table attached to the figure.

Figure 4 shows possible RAL assignments for some activities of the process in Figure 2. The DL definitions of the expressions necessary to automate their resolution can also be seen in the figure (cf. Section 4.4 for details about the mapping to DLs).

Submit Paper: A *Project's PhD Student* is in charge of submitting the paper.
RAL: `HAS ROLE ProjectsPhDStudent`
DL: $RALSubmitPaper \equiv \exists occupies.(\exists participatesIn.\{ProjectsPhDStudent\})$

Fill Travel Authorization: The performer of task *Submit Paper* in the ongoing instance must fill in the authorization form.
RAL: `IS PERSON WHO DID ACTIVITY SubmitPaper IN CURRENT INSTANCE`
DL: $RALFillTA \equiv \exists hasPerformer^{-}.(\exists hasActivity^{-}.\{tm_1\} \sqcap SubmitPaper)$

Make Reservations: Anybody but the performer of task *Sign Travel Authorization* can make the reservations required.
RAL: `NOT (IS PERSON WHO DID ACTIVITY SignTA IN ANY INSTANCE)`
DL: $RALMakeReservations \equiv Person \sqcap \neg(\exists hasPerformer^{-}.(\exists hasActivity^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagement)) \sqcap SignTA))$

Fig. 4. Resource assignments to activities of the process in Figure 2

4 Extending RAL's Semantics to Support History-Based Distribution

As explained in [7], RAL's semantics is defined in DLs [11], which provides the language with analysis capabilities that can be exploited with operations implemented in current DL reasoners, e.g. Pellet and HermiT. The goal now is to define formal semantics for the history-aware RAL expressions. For them to be accurate, it is necessary to store run-time information that is necessary to know the actual performer of any activity already executed. Therefore, ontological elements relating to the performers of the activities of each instance of a BP have to be added to the previous Web Ontology Language (OWL) ontology defined for RAL as detailed in the following sections.

4.1 New OWL Upper Ontology

Previous RAL's semantics was provided from design-time perspective for a single instance of the BP. Only organizational aspects were thus considered in the OWL ontology generated to resolve the assignments. However, in order to store and use information about who has performed each activity of a BP in the different execution instances, new *classes* and *properties* have to added to the TBox of the OWL upper ontology. In particular, we have added one OWL class for *some* of the classes related to BPs (white boxes in the meta model of Figure 1), specifically for *Activity*, and *BusinessProcess*. Class *RALExpression* can be considered equivalent to current class *Person*, since a RAL expression represents a sub-set of the members of an organization [5]. Therefore, it is not inserted as a new OWL class in the upper ontology, using *Person* for the same purpose. In addition, although class *ActivityInstance* has been added to the meta model in Figure 1 to show the real link between the organizational meta model and the BP meta model, it actually represents an *instance* of an activity, so it is

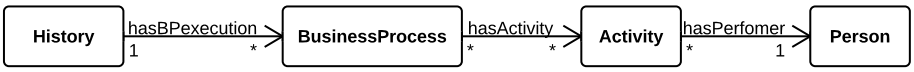


Fig. 5. OWL upper ontology extension to deal with history information

not part of the upper ontology, either. However, we have had to add some extra information required to deal with the negation form (i.e. operator NOT) of the history-aware RAL expressions. Specifically, class *History* has been created to represent the overall history of the executions of the BPs of the organization.

Object properties *hasBPexecution*, *hasActivity* and *hasPerformer* have been added respectively to associate *History* with *BusinessProcess*, *BusinessProcess* with *Activity*, and *Activity* with *Person*, as depicted in Figure 5. Inverse properties have been created to ease the use of the ontology[1]. Furthermore, class *Activity* has a *data property* called *wasCompleted* of standard type *xsd:dateTime*, to save the completion date of the activity.

4.2 Refining the Ontology for Each Business Process

The upper ontology in the TBox will be refined for each BP used in the organization, by introducing sub-classes of the classes previously defined, together with specific configuration necessary to make it work properly. We are taking as example activity *Submit Paper* of the BP in Figure 2 to show this refinement.

First of all, a sub-class of *BusinessProcess* has to be defined to represent the BP for conference trip management. Then, a new sub-class of *Activity* is introduced for each activity *type* of the process. Similarly, classes representing the RAL expressions of the activities (e.g. $RALSubmitPaper$) are added, together with one axiom for each activity to indicate that it has *exactly* one performer that belongs to the subset of $Person$ defined by the RAL expression. Finally, two axioms are added to state that the process is composed of all of its activities and to indicate that all activities are disjoint. The definition in DL is done as follows:

$TripManagement \sqsubseteq BusinessProcess$
$SubmitPaper \sqsubseteq Activity$
$SubmitPaper \sqsubseteq = 1 hasPerformer.RALSubmitPaper$
$TripManagement \sqsubseteq \exists hasActivity.(SubmitPaper \sqcup ... \sqcup MakeReservations)$
$SubmitPaper \sqsubseteq \neg\{FillTravelAuthorization, ..., MakeReservations\}$

4.3 Instantiation of the BP-related Elements

Regarding instantiation of the classes, we will have one and only one instance of class *History*, which will be related to the specific elements stored in the ontology. Then, every time a new process instance is started, OWL instances (or individuals) of the corresponding classes have to be added to the ontology, and the appropriate property associations have to be configured also at this level.

[1] For the sake of understanding, we will use syntax $property^{-}$ to refer to them.

$History(hist)$
$TripManagement(tm_1)$
$hasBPexecution(hist, tm_1)$
$SubmitPaper(sp^1_{tm_1})$
$hasActivity(tm_1, sp^1_{tm_1})$

Once a person is selected from the set of potential performers (i.e. the activity might be allocated to that person at run time), the individual must be added as performer of *that* instance of the activity. Similarly, when the activity instance is complete, the completion time is set on the activity.

$hasPerformer(sp^1_{tm_1}, person_X)$
$wasCompleted(sp^1_{tm_1}, timestamp_1)$

Finally, some technical details are required to complete the ontology. First, all instances are set as different from each other, since DL does not assume it. Second, to avoid unintuitive effects of the open world assumption in DL [11], each time a new execution is added to the ontology (e.g. a new activity $sp^i_{tm_j}$ of type *SubmitPaper* is started in process tm_j), the instance tm_j is updated to indicate that it has exactly i activities of type *SubmitPaper*. The same applies to new processes and the history instance:

$tm_1 \neq tm_2 \neq ... \neq tm_n, sp^1_{tm_1} \neq sp^1_{tm_2}, \neq ... \neq, sp^1_{tm_n}$
$tm_j \in= ihasActivity.SubmitPaper$
$hist \in= jhasBPexecution.TripManagement$

4.4 Mapping History-Aware RAL Expressions into DLs

Finally, we have to map every RAL expression of the BP into DLs, e.g. the assignment defined in class *RALSubmitPaper*. In order to do so, each kind of RAL expression must have an accurate and well-defined semantics. In Table 1, we present the DL definition of the history-aware RAL expressions. As execution information is sometimes needed, to be able to provide accurate semantics we assume we are running the process and, thus, we know the identifier of the current process instance (in this case tm_1). Activity *Submit Paper* is once again used as example. If operator NOT is used, e.g. `NOT (IS PERSON WHO DID ACTIVITY SubmitPaper IN ANOTHER INSTANCE)`, the mapping of such expressions is $Person \sqcap \neg(map_p(expr))$, where $map_p(expr)$ is the corresponding DL expression in Table 1.

Notice that the semantics we have defined implies that the expressions are resolved at run time, since they require run-time information. To enable design-time analysis, design-time semantics of the history-aware RAL expressions is provided in Table 2. As for the negation, all the expressions in the table are mapped to $Person$. Obviously, this mapping just provides an approximation of the real semantics (cf. Table 1), useful to work at design-time, when run-time data is missing.

4.5 Automated Resolution of History-Based RAL Expressions

As explained in [7], as RAL's semantics is defined in DLs, we can use operations already implemented in existing DL reasoners to perform analysis operations

Table 1. Accurate mapping of history-aware RAL expressions into DL *concepts*

History-Based RAL Expression (*expr*)	**DL Mapping** ($map_p(expr)$)
IS PERSON WHO DID ACTIVITY SubmitPaper	
IN CURRENT INSTANCE	$\exists hasPerformer^{-}.(\exists hasActivity^{-}.\{tm_1\} \sqcap SubmitPaper)$
IN ANY INSTANCE	$\exists hasPerformer^{-}.(\exists hasActivity^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagement) \sqcap SubmitPaper))$
IN ANOTHER INSTANCE	$\exists hasPerformer^{-}.(\exists hasActivity^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagement) \sqcap \neg\{tm_1\}) \sqcap SubmitPaper)$
FROM startDate TO endDate	$\exists hasPerformer^{-}.(\exists hasActivity^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagement) \sqcap SubmitPaper \sqcap \exists(wasCompleted \geq startDate) \sqcap \exists(wasCompleted \leq endDate)))$
IS PERSON WHO HAS PARTICIPATED IN	
CURRENT INSTANCE INSTANCE	$\exists hasPerformer^{-}.(\exists hasActivity^{-}.\{tm_1\})$
ANY PROCESS INSTANCE	$\exists hasPerformer^{-}.(\exists hasActivity^{-}.(\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagement))$
ANOTHER PROCESS INSTANCE	$\exists hasPerformer^{-}.(\exists hasActivity^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagement) \sqcap \neg\{tm_1\})$
A PROCESS INSTANCE BETWEEN startDate AND endDate	$\exists hasPerformer^{-}.(\exists hasActivity^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagement) \sqcap \exists(wasCompleted \geq startDate) \sqcap \exists(wasCompleted \leq endDate)))$

Table 2. Approximate mapping of history-aware RAL expressions into DL *concepts*. The mappings missing are done exactly like in Table 1.

History-Based RAL Expression (*expr*)	**DL Mapping** ($map_p(expr)$)
IS PERSON WHO DID ACTIVITY SubmitPaper	
IN CURRENT INSTANCE IN ANY INSTANCE IN ANOTHER INSTANCE	$\exists hasPerformer^{-}.SubmitPaper$
IS PERSON WHO HAS PARTICIPATED IN	
CURRENT INSTANCE INSTANCE ANY PROCESS INSTANCE ANOTHER PROCESS INSTANCE	$\exists hasPerformer^{-}.(\exists hasActivity^{-}.TripManagement)$

over the RAL expressions of a BP and, so, infer information about how resources are being managed in the process. Specifically, to obtain the potential performers of an activity we must execute the operation *individuals* on the RAL expression, e.g. for activity *Submit Paper* it is $individuals(\exists hasPerformer^{-}.SubmitPaper)$.

5 Related Work

The importance of considering resources as part of BPM is a well-known concern [12]. Approaches have typically focused on one single process instance, disregarding what has happened in the past [13]. However, more and more history-based allocation is coming on stage, and we can find several proposals dealing with the introduction of this pattern in BP models.

Bertino et al. presented a language for defining Role-Based Access Control (RBAC) constraints to tasks in a workflow (WF) [4]. The approach is similar to RAL in the sense that it is also built on a formal basis and is also aimed at checking constraint consistency. Besides, algorithms for planning resource assignments to the various tasks were also introduced. However, the language was considerably difficult to use, as it was not user-oriented.

Wolter and Schaad assigned specific semantics to BPMN swimlanes in order to represent role hierarchies, introduced Manual Tasks and extended some BPMN artifacts to define assignments [1]. Only four creation patterns were supported by this approach, History-Based Distribution among them.

Russell and van der Aalst examined BPEL4People and WS-HumanTask[2] using the WRPs as evaluation framework [14]. Compared to RAL, these standards provide less support for the creation patterns, but past information is maintained as history task events.

Awad et al. used the WRPs again as a reference framework to study resource management in BPMN 1.2, and proposed a meta model extension for the notation [2]. The meta model used to capture history information is very similar to ours (cf. Figure 1) and the approach was quite expressive, but the solution is ad-hoc for BPMN and there is not a unified formalism to assign and analyse resources. Different techniques are utilised for each pattern instead.

Recently, Strembeck and Mendling have introduced *Business Activities*, a UML extension that enables the definition of process-related RBAC models [3]. The history of a process instance is recorded in what they call Business Activity RBAC Model (BRM). However, unlike RAL, it is constrained to UML, and other types of resource assignments are not considered in their proposal.

6 Conclusions and Future Work

In this paper, we have described how to add history execution information of a BP to RAL, extending so the scope of the language. The result is a resource assignment language that can be used in different BP notations, supports all the creation patterns [10] and can be used with other WRPs (visit

[2] `http://www.oasis-open.org/committees/bpel4people/`

`www.isa.us.es/cristal` for further information about RAL and the WRPs), and has analysis capabilities to automatically resolve RAL expressions and infer interesting information from a RAL-aware BP model, now considering past executions as well. The next improvement step is to provide RAL with accurate *execution* semantics, integrate it into a BPMS, and undertake performance tests to optimize it. Part of this work is currently being performed.

References

1. Wolter, C., Schaad, A.: Modeling of Task-Based Authorization Constraints in BPMN. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 64–79. Springer, Heidelberg (2007)
2. Awad, A., Grosskopf, A., Meyer, A., Weske, M.: Enabling Resource Assignment Constraints in BPMN. tech. rep., BPT (2009)
3. Strembeck, M., Mendling, J.: Modeling process-related RBAC models with extended UML activity models. Inf. Softw. Technol. 53, 456–483 (2011)
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. Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 50–61. Springer, Heidelberg (2012)
6. 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)
7. Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: Defining and Analysing Resource Assignments in Business Processes with RAL. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 477–486. Springer, Heidelberg (2011)
8. Decker, G., Overdick, H., Weske, M.: Oryx – An Open Modeling Platform for the BPM Community. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 382–385. Springer, Heidelberg (2008)
9. Cabanillas, C., del-Río-Ortega, A., Resinas, M., Ruiz-Cortés, A.: CRISTAL: Collection of Resource-centrIc Supporting Tools And Languages. In: Lohmann, N., Moser, S. (eds.) BPM 2012 Demos, vol. 940, pp. 51–56. CEUR-WS (2012)
10. Russell, N., ter Hofstede, A., Edmond, D., van der Aalst, W.: Workflow Resource Patterns. tech. rep., BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven (2004)
11. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logics Handbook: Theory, Implementations, and Applications. Cambridge University Press (2003)
12. 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)
13. Grosskopf, A.: An Extended Resource Information Layer for BPMN. Tech. rep., BPT (2007)
14. Russell, N., van der Aalst, W.M.P.: Work Distribution and Resource Management in BPEL4People: Capabilities and Opportunities. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 94–108. Springer, Heidelberg (2008)