Manuel Resinas and Adela del-Río-Ortega and Antonio Ruiz-Cortés and Cristina Cabanillas Macias

Specification and Automated Design-Time Analysis of the Business Process Human Resource Perspective

Specification and Automated Design-Time Analysis of the Business Process Human Resource Perspective☆

Cristina Cabanillas[a,**], Manuel Resinas[b,*], Adela del-Río-Ortega[b,*], Antonio Ruiz-Cortés[b,*]

[a]*Institute for Information Business, Vienna University of Economics and Business, Welthandelsplatz 1/D2/C, 1020 Vienna, Austria*

[b]*Dpto. Lenguajes y Sistemas Informáticos, University of Seville, Avda. Reina Mercedes s/n, E.T.S. Ingeniería Informática, 41012 Sevilla, Spain*

Abstract

The human resource perspective of a business process is concerned with the relation between the activities of a process and the actors who take part in them. Unlike other process perspectives, such as control flow, for which many different types of analyses have been proposed, such as finding deadlocks, there is an important gap regarding the human resource perspective. Resource analysis in business processes has not been defined, and only a few analysis operations can be glimpsed in previous approaches. In this paper, we identify and formally define seven design-time analysis operations related to how resources are involved in process activities. Furthermore, we demonstrate that for a wide variety of resource-aware BP models, those analysis operations can be automated by leveraging Description Logic (DL) off-the-shelf reasoners. To this end, we rely on Resource Assignment Language (RAL), a domain-specific language that enables the definition of conditions to select the candidates to participate in a process activity. We provide a complete formal semantics for RAL based on DLs and extend it to address the operations, for which the control flow of the process must also be taken into consideration. A proof-of-concept implementation has been developed and integrated in a system called CRISTAL. As a result, we can give an automatic answer to different questions related to the management of resources in business processes at design time.

Keywords: automated analysis, analysis operation, business process management, human resource perspective, RAL, resource assignment

☆This work was partially supported by the Austrian Research Promotion Agency (FFG), the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants 845638 (SHAPE), P12-TIC-1867 (COPAS), TIN2012-32273 (TAPAS), TIC-5906 (THEOS)).

*Corresponding author

**Principal corresponding author. *Phone number:* +43 1 31336 5216

Email addresses: `cristina.cabanillas@wu.ac.at` (Cristina Cabanillas), `resinas@us.es` (Manuel Resinas), `adeladelrio@us.es` (Adela del-Río-Ortega), `aruiz@us.es` (Antonio Ruiz-Cortés)

1. Introduction

The human resource perspective of a Business Process (BP) [1] (also known as the organisational perspective [2]) is concerned with the relation between the activities of a process and the human resources[1] that take part in them. The management of resources in Business Process Management (BPM) encompasses several tasks, typically divided into two groups. *Resource assignment* is the design-time definition of the conditions (resource selection conditions from now on) that must be fulfilled by the company members to become candidates to work on the process activities. The outcome is a *resource-aware BP model*, i.e., a process model annotated with resource selection conditions. *Resource allocation* is the run-time designation of the actual performers of the activities before their execution, which includes, for instance, mechanisms for resource prioritisation that may ease the distribution of work.

Like in other BP perspectives (e.g., the control flow), analysis of the resource perspective may provide insights that are relevant for the execution of the process. For instance, both assignment and allocation must guarantee a deadlock-free execution. Therefore, it is of utmost importance to ensure that the resource-aware process model is consistent, i.e., that there are candidates for all the activities. It is also helpful to know beforehand the workload a resource may have during the execution of a specific process, i.e., which activities of the process may be allocated to her.

Resource management in Business Processes (BPs) in general and analysis in particular have not yet reached the degree of maturity of other BP perspectives, such as control flow. Specifically, the following gaps have been found. First, to the best of our knowledge, only two analysis operations have been identified and tackled in the literature so far, namely, determining the candidates to execute a process activity given a set of selection conditions (i.e., the *potential participants* in a BP activity) and checking whether a resource-aware BP model is consistent. Second, there are very few software prototypes that implement these operations, and only a subset of them are independent of any BP modelling language used to specify the process. Finally, a paradigm that underpins the analysis of this BP perspective similarly to the one provided by Petri nets for the control flow perspective is missing. Therefore, the efforts necessary to formally define these operations will take more time to converge.

We focus on increasing the degree of maturity of analysis in the BP resource perspective, specifically with regard to resources. In particular:

- We define a catalogue of seven *person-activity operations* related to how resources are involved in activities. The catalogue is divided into three categories: basic, consistency checking, and criticality checking operations. Five of the seven operations are novel.

- We propose a way to define resource-aware BP models by using Resource Assignment Language (RAL) [3], a language to define resource selection conditions that is independent of any process modelling notation.

[1]For the sake of simplicity, in the rest of the paper we use *resource* to refer to *human resources*.

- We propose Description Logics (DLs) as a paradigm to underpin the analysis of resource-aware BP models based on RAL, and we show that for the R3C-processes, a term we coin to denote a class of resource-aware BP models that meet certain conditions (cf. Section 5), it is possible to interpret the entire set of analysis operations in terms of DLs.

- We offer a proof-of-concept implementation of the catalogue of analysis operations. This catalog is integrated into a larger system called Collection of Resource-centrIc Supporting Tools And Languages (CRISTAL) [4], which provides several tools for the management of the BP resource perspective. The core of the prototype is a DL reasoner, which reduces the development effort and the likelihood of failure.

A preliminary version of RAL and its semantics have been presented in previous publications [3, 5]. In this paper, we extend them as follows. First, we revisited the RAL specification and separated the RAL expressions into different modules. We also added support to define resource assignments for different degrees of involvement in the process activities, also called *task duties*. For instance, RAL allows defining selection conditions for the person in charge of carrying out the work, the person who must approve the work performed and the person who must receive notifications related to an activity. These and other duties have been identified and used in a few approaches, such as BPEL4People [6] and RACI [7]. Second, we adapted and extended the RAL semantics originally defined in DLs. The extension takes into account specific features required for the automation of the seven analysis operations mentioned above. The overall idea of the extension is to include in the DL-based Knowledge Base (KB) required information about other BP resource perspectives [8], specifically the control flow of the process. Finally, we provide the DL formulas dealing with the automated resolution of the analysis operations at design time based on the extended KB.

The rest of this paper is structured as follows. Section 2 describes a running scenario that is used throughout this paper. Section 3 defines automated resource analysis in BPs and the person-activity analysis operations, which constitute the main goal of this work. Section 4 presents the current version of RAL. Section 5 introduces the conditions a resource-aware BP model must fulfil to be an R3C-process and the characteristics that make it amenable to automatic analysis using DLs. Section 6 describes the semantics of the BP resource perspective using RAL for resource assignment. Section 7 describes the content of a KB to address the analysis operations at design time, and it presents the DL expressions for the implementation of the operations. Section 8 presents an evaluation of RAL expressiveness and describes an implementation of the analysis operations and its integration into CRISTAL [4]. Finally, Section 9 summarises the revision of the state of the art on the design-time analysis of resources in BPs, and Section 10 closes the paper by drawing several conclusions and outlining potential future work.

2. Running Example

In the following, we describe a scenario that will be used as a running example throughout this article. We highlight some concepts that we elaborate later on.

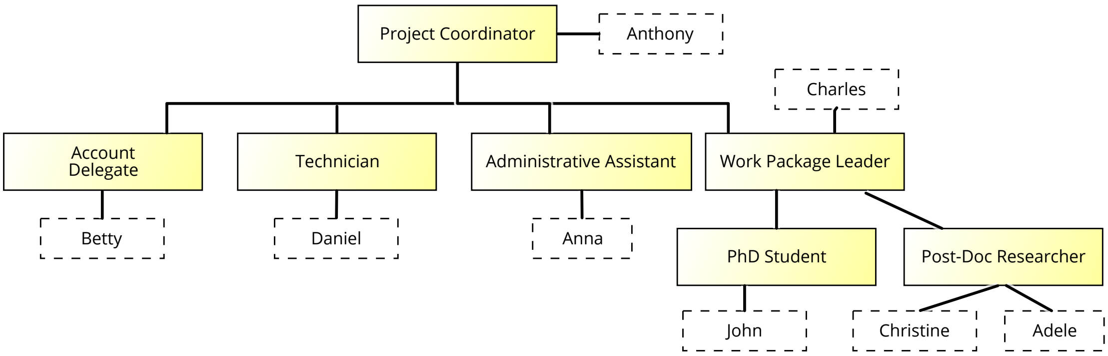
Let us assume that we belong to the ISA research group of the University of Seville and that we take part in a hypothetical research project called Human Resource Management System (HRMS). The model shown in Figure 1 represents the hierarchy of organisational *positions* that are involved in the *organisational unit* HRMS[2]. Seven positions (Project Coordinator, Account Delegate, Technician, Administrative Assistant, Work Package Leader, PhD Student and Post-Doc Researcher) *are members of* this unit, and eight *persons* (Anthony, Betty, Daniel, Anna, Charles, John, Christine and Adele) *occupy* them. The hierarchy of positions defines the reporting lines among the members of HRMS so that, for instance, the people occupying the position Work Package Leader (i.e., Charles) *report to* the Project Coordinator(s) (i.e., Anthony), and they *can delegate work to* people occupying the position PhD Student or Post-Doc Researcher (i.e., John, Christine and Adele) because they are lower in the hierarchy. Similarly, the Project Coordinator (i.e., Anthony) does not report to anyone, but he can delegate work to any other member of the project. A table attached to the figure depicts the roles people have according to the positions they occupy. Note that for the sake of brevity, people may have a set of capabilities (e.g., skills or education) that are not represented in the figure.

The procedure illustrated in Figure 2 represents a *collaboration* between two BPs modelled with Business Process Model and Notation (BPMN) 2.0[3] [9]: one BP is developed at pool *Research Vice-chancellorship* and the other at pool *ISA Research Group*. The procedure consists of a simplified version of the procedure to manage a trip to a conference, according to the rules of the University of Seville. We are going to focus on the BP carried out at pool *ISA Research Group*. The process starts when a researcher submits the camera ready version of a paper (activity A[4]) that has been accepted for publication in a conference. Then, the person who will present the paper at the conference must fill out a *Travel Authorisation* form (activity B) to request permission. Any required information she is unable to fill in can be requested from another member of the project, e.g., the funding project for the trip. Once the form is filled out, the principal investigator of the funding project is notified, as she is responsible for approving the trip (activity C). When the document is signed, it is sent to the *Research Vice-chancellorship* (activity D) for external revision to ensure that all the requirements are met. If it is approved, the potential attendee must register at the conference (activity F) and provide all the information (e.g., venue place and dates) to a clerk, who makes the reservations required (activity G). Such reservations must be checked by the attendee afterwards. If the authorisation is not approved, it must be filled out again and the evaluation process is repeated until it is finally approved.

[2]Please note that this model is inspired by reality, but the values (roles, positions, persons, *etcetera*) have been modified due to confidentiality issues.

[3]In BPMN a process takes place within a single pool. Diagrams with two or more pools, in which messages between the pools are exchanged, are called collaborations.

[4]We use letters from A to G to refer to the activities of the process.



Position	Role
Project Coordinator	Principal Investigator
	Project Account Administrator
	Project Resource Manager
Work Package Leader	Work Distributor
	Project Staff Member
Post-Doc Researcher	Project Staff Member
	Researcher
PhD Student	Researcher
Administrative Assistant	Clerk
…	…

Figure 1: Excerpt of the organisational model of the ISA group for project HRMS

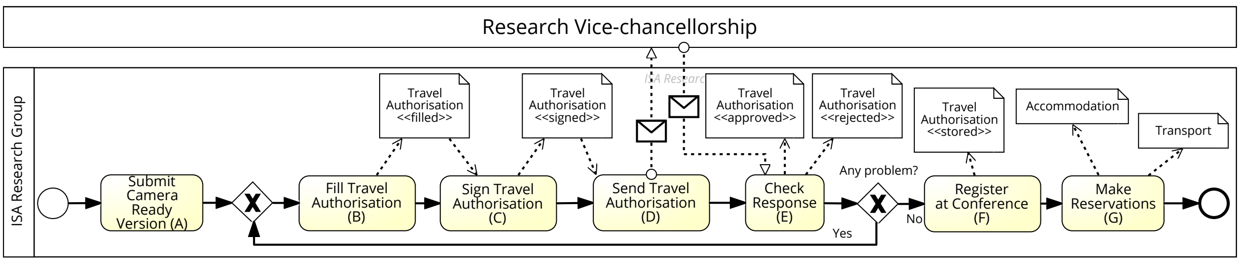


Figure 2: BP to manage the trip to attend a conference

Figure 3 shows the resource assignments for the running example. Please, note that there are several task duties associated with some of the activities, in particular those defined in RACI matrices [7], namely, Responsible, Accountable, Consulted and Informed. The expressions range from conditions merely based on the organisational structure (i.e., roles, positions, etcetera) to access-control constraints [10], specifically Binding of Duties (BoD) in the four last activities. Access-control constraints define security conditions stating either that the same resource must perform two specific activities (BoD) or that the same resource cannot execute two specific activities (Separation of Duties (SoD)). Please note that although RAL is mentioned in the figure, it will be explained in Section 4.

Submit Camera Ready Version (A). A *Researcher* or any person with role *Project Staff Member* in project *HRMS* is responsible for submitting the paper to the conference.

```
(HAS ROLE Researcher IN UNIT HRMS) OR (HAS ROLE ProjectStaffMember IN UNIT HRMS)
```

Fill Travel Authorisation (B). The authorisation form must be filled out by a researcher of HRMS.

```
HAS ROLE Researcher IN UNIT HRMS
```

Any member of the project can be consulted to fill in information required.

```
HAS UNIT HRMS
```

The principal investigator of the funding project is informed afterwards.

```
HAS ROLE PrincipalInvestigator IN UNIT IN DATA FIELD TravelAuthorisation.Project
```

Sign Travel Authorisation (C). The form must be signed by the coordinator of project HRMS.

```
HAS POSITION ProjectCoordinator
```

Send Travel Authorisation (D). This activity must be performed by the person that filled out the travel authorisation form.

```
IS ANY PERSON responsible for ACTIVITY FillTA
```

Check Response (E). The response received can be checked by anyone from the same project having some position in common with the person that submitted the paper in the current BP instance.

```
(HAS UNIT HRMS) AND (SHARES SOME POSITION WITH ANY PERSON responsible for ACTIVITY
SubmitCRV)
```

Register at Conference (F). The person who sent the travel authorisation in the ongoing instance is due to register at the conference, as long as she occupies position HRMS PhD Student.

```
(IS ANY PERSON responsible for ACTIVITY SendTA) AND (HAS POSITION PhDStudent)
```

The information about the conference and the trip is sent to a clerk.

```
HAS ROLE Clerk
```

Make Reservations (G). The clerk who was notified before is responsible for making the reservations required.

```
(HAS ROLE Clerk) AND (IS ANY PERSON informed in ACTIVITY RegisterAtConference)
```

The person attending the conference must approve these reservations.

```
IS ANY PERSON responsible for ACTIVITY RegisterAtConference
```

Figure 3: Resource selection conditions for the activities of the process in Figure 2

3. Resource Analysis in Business Processes

The automated analysis of the BP resource perspective can be defined as the automated extraction of information from resource-aware BP models about the resources that may take part in the process activities. Following the same approach that has been used with process performance indicators [11] and in other fields such as Software Product Lines (SPLs) [12], we define the automated analysis in terms of a set of analysis operations. Specifically, from the study of the state of the art on resource analysis in BPs (cf. Section 9) and the needs identified in conversations with several Andalusian ICT companies, we have defined a catalogue of seven person-activity operations related to the involvement of resources in the BP activities.

This catalogue can be divided into three categories: basic operations, consistency checking operations and criticality checking operations. All of them can be applied to any task duty associated with a BP activity and can be executed in different phases of the BP lifecycle. The phase of the lifecycle is relevant because it may have an influence on its implementation. In this paper, we focus on design-time analysis, i.e., the design and analysis phase of the BP lifecycle [8, 13]. The operations have been defined to be as reusable as possible, and an implementation of each of them in DLs is detailed in Section 7.

3.1. Basic Person-Activity Operations

These operations analyse the relations between the activities of a process and the people who can perform them according to the resource assignments. There are four basic person-activity operations, one of which (Potential Participants) has already been identified in the literature.

3.1.1. Potential Participants (PP)

The PP operation takes an activity and a task duty and returns the people who are candidates to perform that specific task duty for the activity specified. Thus, at design time, a person is a potential participant of an activity for a specific task duty if there is *some* BP instance in which she can be an actual performer of that task duty[5].

Although obtaining the potential participants of an activity is sometimes straightforward, the presence of access-control constraints in BPs may make it significantly more difficult, especially when they affect loops. Let us illustrate this point with activities B and F of the running example (cf. Figure 2). As shown in Figure 3, the person responsible for the former is any person with role *Researcher* within unit *HRMS*; the person responsible for the latter is any person with position *PhD Student* who was responsible for activity D. Finally, the responsible for activity D is any person responsible for activity B. Therefore, activities B, D and F must be performed by the same person, i.e., there is a BoD between them.

As depicted in Figure 1, there are only three people in the project with the role *Researcher* (required for B), namely John, Christine and Adele; among them, only $John$ occupies position *PhD Student* (required for F). Consequently, only $John$ can participate in all B, D, and F. This means that if B, D, and F are executed only once in a process instance, then only $John$ can perform them.

However, note that B can be executed more times in a single BP instance, in case there is some problem with the travel authorisation form. In that case, there are two possible

[5]Note that from this definition, *participant* and *performer* can be used as synonyms in this context.

interpretations for the potential participants of B, namely, the relaxed interpretation and the strict interpretation.

The relaxed interpretation is that if activity B has already been allocated to John, the subsequent executions of the activity can be performed by Christine and Adele as well because there is already a past actual performer of the activity who can be allocated to F and D without violating the BoD constraint, which is John. Therefore, in this interpretation, the potential participants of activity B for the task duty Responsible are John, Christine, and Adele because the three of them may be actually responsible for the activity at some moment, provided that John had been responsible for the activity at least once in the same process instance.

The strict interpretation is that B can only be performed by people who can also perform activities D and F, i.e., those that could perform B, D, and F if they were executed only once. In this interpretation, the only potential person responsible for activity B is John.

The decision of which interpretation to choose is domain-specific and depends on the specific activity to which the potential participants operation is applied. Therefore, two variants of the potential participants operation are considered: PP, which uses the relaxed interpretation, and α-PP, which uses the strict interpretation. In our example, $PP(B, responsible) = \{John, Christine, Adele\}$ and α-PP$(B, responsible) = \{John\}$.

Example. In addition to the aforementioned examples, according to the scenario described in Section 2, the potential persons responsible for activity A are John, Christine, Adele and Charles, and Anthony is the only person potentially responsible for activity C.

Applicability. This operation serves for studying or checking whether people are involved in specific types of activities as well as for detecting security problems derived from an incorrect assignment of permissions in terms of activity execution, i.e., a person who was supposed to be involved in an activity but cannot take part in it due to the assignment. It is also useful to detect activities that can be assigned to the same resources and, hence, are candidates for aggregation when creating an executable BP model [13]. Furthermore, typical operations for set comparison used in Set Theory [14] can be applied to this operation, e.g., to determine whether the potential participants in two given activities are exactly the same resources.

3.1.2. Potential Activities (PA)

The PA operation lists the activities that may be allocated to one resource with regard to a specific task duty during a process instance execution. It takes the identity of a specific person and the task duty to be checked, and it returns the activities that can be potentially allocated to this person for that task duty. Like potential participants, there are two variants of this operation: PA and α-PA depending on whether one chooses the relaxed interpretation or the strict interpretation, respectively.

Example. The potential activities for which John may be responsible in the running scenario are *A, B, D,* and *F* because he is a potential participant of these activities for task duty Responsible according to the conditions defined in the resource assignments.

Applicability. This operation is useful to provide people with a personalised list of all of the activities they may be involved in or to identify the requirements for someone who is going to substitute a certain person in the organisation. It is also useful to detect the degree of involvement of a person in a BP in terms of the number of activities in which she can take part. Moreover, similar to potential participants, typical operations for set comparison can also be used to determine, for instance, whether the set of activities that can be allocated to a specific person is a subset of the set of activities potentially allocated to another person.

3.1.3. Non-potential Activities (NPA)

The NPA operation takes a person and a task duty and calculates the activities in which she *cannot* perform that task duty, if any. Like potential participants, there are two variants of this operation: NPA and α-NPA depending on whether one chooses the relaxed interpretation or the strict interpretation, respectively.

Example. In the running scenario, John cannot be responsible for activity C.

Applicability. This operation is useful when one is interested in increasing the responsibilities of a person in the organisation. The outcome of this operation is a set of activities whose resource assignments are candidates to be changed to include the resource at hand.

3.1.4. Non-participants (NP)

The NP operation takes an activity and a task duty and returns the people who can never participate in the activity performing that task duty, if any. Like potential participants, there are two variants of this operation: NP and α-NP depending on whether one chooses the relaxed interpretation or the strict interpretation, respectively.

Example. In the running example, the non-participants of task duty Responsible in activity A are Anna, Daniel, Betty, and Anthony, and all but Anthony are non-participants in the task duty in activity C.

Applicability. This operation is a way to quickly detect the relationship between people and BPs in an organisation, making it easier to ensure that certain resources do not have access to BPs that are not aligned with their duties or responsibilities in the company. Such duties may be defined in the form of access-control policies of people to specific types of processes or activities.

3.2. Consistency Checking Person-Activity Operation

This category of operations includes just one operation focused on checking whether for all activities of the process there is at least one person who is allowed to perform the task duty for any execution of the activity. Specifically, the *consistency checking (CC)* operation takes a task duty and returns whether the BP model is consistent with regard to that task duty, i.e., if it is always possible to find a potential participant for an activity during any execution of the process for that task duty. This definition is based on the definition of consistency introduced in [15], although it has been extended to address task duties.

Example. The BP in Figure 2 is consistent regarding task duty Responsible given the resource assignments defined in Figure 3 because there can be at least one potential person responsible for each activity instance in a process instance.

Applicability. An inconsistent process may result in behavioural problems at run time because there may not be anyone to whom some task duty can be allocated in case the activity needs to be executed in a BP instance. Therefore, this operation is fundamental to ensure the correct operation of the BP resource perspective, as it detects situations in which the process could fall into a deadlock.

3.3. Criticality Checking Person-Activity Operations

Apart from consistency, one aspect that is relevant to resource assignment is checking whether there is only one person who is authorised to perform a certain activity of the process. Identifying these people is useful for reducing the vulnerability of the organisation to failure, which, according to Malone et al. [16], is strongly related with the possibility to replace one resource with another. The two novel operations introduced next detect weak points of a process in the face of resource unavailability.

3.3.1. Critical Participants (CP)

One or more people are critical participants of a BP if they have to be allocated to one or more activities because there are no more potential participants for them. The CP operation takes a task duty and returns the members of the organisation who are critical in the execution of a process for that task duty.

The simplest case is when there is only one potential participant for an activity. However, this operation also has to take into account situations that may appear in the presence of access control constraints. An example is as follows. Let us suppose that the assignment of B is a person with position *Post-Doc Researcher* and the assignment of F is an SoD with B. Moreover, the participant must also have position *Post-Doc Researcher*. According to the organisational model in Figure 1, only Christine and Adele have that position. In this scenario, the potential persons responsible for both activities are $\{Christine, Adele\}$ because there may be a BP instance in which $Christine$ is allocated to B and $Adele$ is allocated to F and another process instance in which the allocations are the opposite. However, although B and G each have two potential persons responsible, both $Christine$ and $Adele$ are critical participants because they must always be allocated either to B or to F, as there are no more potential persons responsible for them.

Example. Anthony is a critical participant in the process for task duty Responsible in the running example because he is the only potential person responsible for activity C.

Applicability. A process with a critical participant for task duty Responsible is a process whose execution may eventually depend on one unique person. This fact may make the organisation vulnerable in the sense that it may depend on one specific person to complete one of its business processes. Therefore, this operation is useful for identifying those people who have this particular relevance in the organisation. Furthermore, it is also useful as a mechanism to identify potential bottlenecks without the need to gather and analyse process execution logs.

3.3.2. Critical Activities (CA)

An activity is a critical activity for a given task duty if it has only one potential participant for that task duty. The CA operation takes a person and a task duty and returns the critical activities in which that person is involved with the given task duty.

Example. Activity C is critical regarding task duty Responsible because the process gets blocked in the absence of Anthony.

Applicability. Detecting the activities of a process that can only be performed by one person helps pinpoint potential bottlenecks without the need to gather and analyse process execution logs. It is also useful for obtaining the activities whose resource assignments should be modified temporarily or permanently when a specific person is unavailable for a specific (or indefinite) period of time to avoid process deadlocks.

4. Resource Assignment in Business Processes with RAL

Resource Assignment Language (RAL) is a modular, extensible Domain Specific Language (DSL) explicitly developed to define resource selection conditions that can be used to specify resource assignments for the activities of a BP. It was first introduced in [3] and extended in [17], and it allows formulating expressions, such as those shown in Figure 3.

Reusability is at the core of RAL and has guided several high-level decisions in the language's design. Two of these decisions have a particularly strong influence on the language structure. First, RAL constructs are divided into RAL expressions and RAL constraints. Resource selection conditions are specified by means of different types of RAL expressions that may contain different types of *constraints*. This division between expressions and constraints enables the reuse of the latter in different RAL expressions. Second, RAL is a modular language that comprises RAL Core, a common part that allows defining basic assignments based on a resource's characteristics. There are several extensions that add new types of expressions and/or constraints. RAL Core has been defined to be independent of the context in which resource selection is used, i.e., it could also be used to select resources for other purposes in organisations that are not process-oriented.

In this paper, we present four extensions designed for BPM that make up the so-called RAL ODDA as depicted in Figure 4[6]. The constructs added by these extensions have been defined to make the language as expressive as possible without losing automated analysis capabilities while maintaining understandability and coherence in the expressions. Specifically, concerning expressiveness, all RAL ODDA constructs have been chosen to cover (i) the constraints related to the organisational model of the organisation; (ii) a subset of the Workflow Resource Patterns (WRPs) [18] that capture behaviour related to resource assignment in Workflows (WFs), namely the *creation patterns* (see Section 8.1 for details on how

[6] OM and BP represent resp. the organisational and BP metamodels used in RAL.

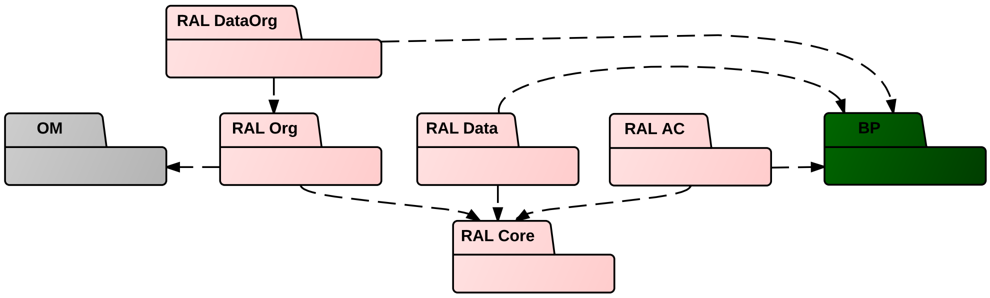


Figure 4: RAL ODDA

RAL ODDA supports them); and (iii) the task duties associated to the activities. In fact, there is no element in the organisational metamodel or creation pattern that is not covered by a RAL ODDA construct except for history-based allocation. Moreover, thanks to RAL modularity, the expressiveness can be improved with new RAL modules that, for instance, could provide support for other organisational metamodels as detailed in [19]. Regarding understandability and coherence, RAL ODDA constructs have been carefully designed to be close to natural language and to feel similar to a unique language despite being four different modules.

In the remainder of this section, we detail RAL Core and all RAL ODDA extensions. Furthermore, the Extended Backus-Naur Form (EBNF) syntax of RAL ODDA is presented in Appendix A. The RAL expressions for the running example are shown in Figure 3. As can be observed, RAL modules can be composed with each other to define conditions, e.g., in activity F RAL AC is used in conjunction with RAL Org.

4.1. RAL Core

RAL Core contains generic resource selection expressions independent of any domain, specifically:

`ANYONE`. It allows selecting any person.

`IS PersonConstraint`. It limits the set of people selected by means of a *PersonConstraint*. In RAL Core the only *PersonConstraint* considered consists of explicitly indicating the identity of one person. For instance, in the domain at hand, the expression `IS David` indicates that David is the only potential performer of the task duty in question.

`NOT (DeniableExpr)`. It allows selecting people who do not meet certain conditions. For instance, the expression `NOT(IS Anthony)` excludes Anthony from a set of potential performers of the task duty in question.

`(Expr) OR (Expr) | (Expr) AND (Expr)`. It allows specifying multiple conditions in the same RAL expression, connecting them with the OR and AND operators. For instance, in Figure 3 the assignment for activity A shows two alternative conditions for resource selection, and the assignments for activities E, F and G indicate that several conditions must be met.

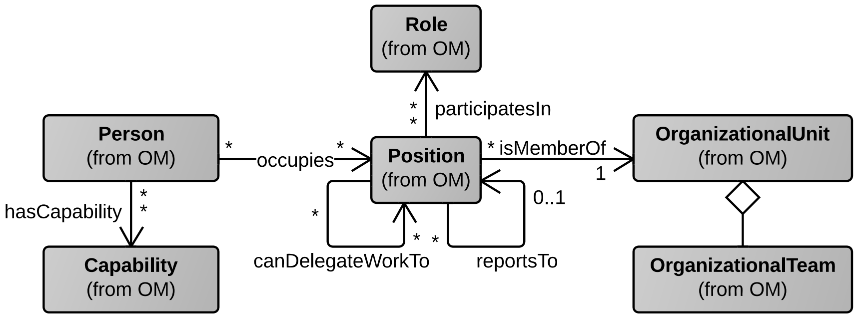


Figure 5: Excerpt of the organisational model described by Russell et al. [20]

4.2. RAL Org

RAL Org extends RAL Core by adding four types of *expressions* and four types of *constraints* that allow selecting people according to their organisational information based on the organisational metamodel depicted in Figure 5. This metamodel is part of the metamodel described by Russell et al. [20] as a basis for the definition of the WRPs [18]. In a nutshell, it consists of persons, capabilities, positions, roles and organisational units. A *person* (also called *individual resource*, *individual* or just *resource*) may have a set of *capabilities*, such as her skills or information related to her professional experience. Each person occupies one or more *positions* within an organisation. In turn, each position participates in one or several *roles* and belongs to one *organisational unit*. Note that because a *position* is a member of just one organisational unit, each organisational unit has its own hierarchy of positions representing the lines-of-reporting within it, and work can be *reported* and/or *delegated* between members of an organisation according to their positions in the organisational units. In particular, the people who occupy a position can report work to their superiors, i.e., the people who occupy the position immediately above and who are directly connected to the lower position in the model; and they can delegate work to those who occupy any position that is lower in the hierarchy as long as it is directly or indirectly connected to it. The organisational metamodel described in the running scenario (cf. Section 2) fits within this metamodel. In the rest of this paper, we use the term *group resource* to refer to positions, roles and organisational units as a whole.

RAL Org expressions and constraints have been defined to cover all of the relations that appear in the metamodel (*occupies*, *isMemberOf*, *participatesIn*, *hasCapability*, *reportsTo*, *canDelegateWorkTo*) plus one expression that selects people based on the group resources shared with a specific person:

`HAS (PositionConstraint | UnitConstraint | RoleConstraint [IN UnitConstraint])`. It enables position-based, organisational unit-based and role-based people selection by means of a *PositionConstraint*, a *UnitConstraint* or a *RoleConstraint*. In RAL Org, these constraints consist of explicitly specifying the position, the organisational unit or the role in question, respectively. Optionally, the role can be constrained to a specific organisational unit using the *isMemberOf* relation of the organisational metamodel. In the running example (cf. Figure 3), examples of position-based selection are the assignment for C and the second part of the assignment for F. An example of organisational unit-based selection is the first condition of the assignment for activity E. Finally, examples of role-based selection are the assignments in activities G, A and B.

`HAS CAPABILITY CapabilityConstraint`. It allows selecting resources based on their capabilities by means of a *CapabilityConstraint*, which consists of either having a certain capability or meeting a certain condition on the value of a capability. For instance, the expression `HAS CAPABILITY MSc` selects all the people with a master's degree.

`[DIRECTLY] REPORTS TO PositionRef | IS [DIRECTLY] REPORTED BY PositionRef`. It allows expressing constraints based on the *reportsTo* relation of the organisational metamodel. `DIRECTLY` is used for stating whether we do not want to move up more than one reporting level by transitivity. For instance, the expression `DIRECTLY REPORTS TO Anthony` selects the people who are one level down with regard to Anthony in the hierarchy shown in Figure 1, i.e., Betty, Daniel, Anna and Charles.

`CAN DELEGATE WORK TO PositionRef | CAN HAVE WORK DELEGATED BY PositionRef`. It is similar to the previous one but using the organisational relation *canDelegateWorkTo*, i.e., moving down in the positional hierarchy. In this case transitivity is implicit by definition (cf. Figure 5). For instance, the expression `CAN DELEGATE WORK TO POSITION OF John` selects the people occupying superior positions in the hierarchy who are connected by transitivity with John's position according to Figure 1, i.e., Charles and Anthony.

`SHARES Amount (POSITION | UNIT | ROLE [IN UnitConstraint]) WITH PersonConstraint`. It allows selecting an individual who has *some* or *all* position(s), role(s) or organisational unit(s) in common with a specific person, indicated by a *PersonConstraint*. An example is the second condition of the assignment for activity E in Figure 3.

4.3. RAL Data and RAL DataOrg

These modules allow selecting individuals or group resources indicated in a data field of a data object of the process according to the BPMN [9] specification of the BP data perspective[7]. Therefore, the required information is unknown until run time; hence, these extensions provide support for the Deferred Allocation creation pattern [20]. We will call the constraints that are focused on the run-time selection of participants *run-time constraints*.

Specifically, RAL Data extends the *PersonConstraint* of RAL Core with the condition `PERSON IN DATA FIELD dataObject.fieldID`. RAL DataOrg extends the *PositionConstraint*, *RoleConstraint*, and *UnitConstraint* of RAL Data in a similar way. An example for the running scenario would state that the potential performer of activity C is the person specified as the main researcher of the project in document *Travel Authorisation*, with the expression `IS PERSON IN DATA FIELD TravelAuthorisation.MainResearcher`.

[7]A *Business Process* can have a set of *Data Objects*, which can contain one or more *Data Fields*, whose values may change throughout execution of the process.

4.4. RAL AC

RAL AC stands for RAL Access-Control and it extends RAL Core to enable the specification of run-time constraints related to the resources allocated to other activities of the process, thus providing support for the SoD, Case Handling and Retain Familiar creation patterns. Furthermore, RAL AC allows selecting resources allocated to process activities with different degrees of responsibility related to their execution, i.e., different task duties.

Therefore, RAL AC extends *PersonConstraint* with the condition `ANY PERSON TaskDuty ACTIVITY activityID` to express that the person specified in the constraint must be the actual performer of a specific or any task duty defined for another activity of *the same* BP instance. The set of task duties considered in RAL AC is open and may vary depending on the organisation. For instance, in case of using the task duties defined in the RASCI matrices [7], there would be one person responsible (R) for the activity, one person accountable (A) for it, one person providing support (S) for its execution, one person who can be consulted (C) during its execution, and one person being informed (I) about milestones related to the activity. For these task duties, the element *TaskDuty* can be defined as follows:

```
TaskDuty := RESPONSIBLE FOR | ACCOUNTABLE FOR | PROVIDING SUPPORT FOR
          | CONSULTANT OF | INFORMED ABOUT
```

In this paper, we take this definition as a reference for the examples provided. Examples are the assignment for D and the first condition in the assignment for F in Figure 3.

5. Properties of R3C-processes

As discussed in Section 3, the person-activity analysis operations must take the semantics of the BP control flow into account. This fact increases both the conceptual and computational complexity of the implementation of these operations, thus making their automation much more difficult. However, as we show next, for some BPs it is not necessary to model the full semantics of the control flow, making them amenable to automatic analysis using DL reasoners, as detailed in Sections 6 and 7. We have coined the term R3C-process to denote such BPs.

An R3C-process is a resource-aware BP whose control flow meets the following three requirements:

- There are no dead activities in the BP, i.e., all activities in the process can be executed.
- For all pairs of activities in the process that are related to each other with an access-control constraint, both are either executed at least once or not executed at all; there is a valid execution in which the two activities are executed exactly once; and if one or both are in a loop, they can be executed an unbounded number of times.
- For all pairs of activities in the process whose resource assignment depends on the same data field (cf. RAL Data and RAL DataOrg in Section 4), both are either executed at least once or not executed at all.

Symbol	Description
O	An organisational model
$AI = A \times P$	Set of possible activity instances of a business process.
(a, p)	Activity a was allocated to person p (task duty Responsible).
$AI^{\mathcal{F}}$	All posible complete traces of a business process that are valid w.r.t its control flow.
$\Sigma = AI^{\mathcal{F}} \times \Delta$	Set of complete executions of the BP including both its trace and the data objects.
σ	Process execution of the BP.
$\#_a^\sigma$	Number of times activity a is executed in σ.
ρ	Resource assignment. For convenience $\rho^\sigma(a)$ represents the people who meet the resource selection condition of activity a according to O and σ.
$R - valid(\sigma)$	Evaluates whether the execution σ has a resource allocation valid w.r.t. the resource assignment.
$D_{A'}$	The data objects used by the resource assignment of any activity $a \in A'$.
$ACg(a)$	The activities that belong to the same AC-group as activity a.
T	The subset of Σ that includes all $R - valid$ process executions.
T_a	The subset of T that includes all $R - valid$ process executions whose trace contains activity a.
$\mathcal{S}$	A set of $R - valid$ tuples similar to T but assuming that all activities are executed at least once.

Table 1: Summary of the most relevant symbols used in the formalisation

The first requirement is actually a requirement for any process from a practical perspective. The second requirement is a restriction only applicable to activities that are related to each other with an access-control constraint, and it is usually applied in the related literature [10]. In fact, as far as we know, all proposals apply a similar requirement or even require that activities with access-control constraints cannot be in a loop. Finally, the third requirement only applies to activities that depend on the same data field, which is an improvement in comparison with related literature because the related studies do not even support the use of data objects in resource assignments (cf. Section 9).

In the following, we formalise the notions that have been intuitively introduced in the previous sections and prove that for R3C-processes it is not necessary to model the full semantics of the control flow to perform person-activity analysis operations. For the sake of simplicity, we first consider solely one task duty and in Section 5.5 we show how it can be extended to several task duties.

5.1. Preliminaries

Some definitions are necessary to formalise the notions that have been intuitively introduced in the previous sections. Table 1 summarises the most relevant ones.

Definition 1 (Activity instance)**.** *Let A be the set of activities of a business process bp, and P be the set of persons in an organisation O. An activity instance is a tuple (a, p) that represents the execution of an activity $a \in A$ by a person $p \in P$, which also means that p has been allocated to activity a. The set of all possible activity instances is $AI = A \times P$. For convenience, we define two operations on activity instances: $\pi_a(ai) = a$ and $\pi_p(ai) = p$ for any activity instance $ai = (a, p)$.*

Note that this definition of activity instance assumes that only one person can be allocated to an activity, and hence, there is only one task duty associated to the execution of an activity. However, the results for one task duty can be easily extended to several task duties as described in Section 5.5.

Definition 2 (Execution trace)**.** *Let bp be a business process, A be the set of activities of bp, $A_{init} \subseteq A$ be the subset of activities with which bp can start, and $A_{end} \subseteq A$ be the subset of activities with which bp can end. An execution trace of length $n \in \mathbb{N}$ is a function $\tau : \{0, \ldots, n-1\} \mapsto AI$ that specifies a sequence of activities that can be executed in sequential order according to the control flow of business process bp and the person allocated to the activity,*[8] *where $\pi_a(\tau(0)) \in A_{init}$. The set of all traces of arbitrary length over AI such that $\pi_a(\tau(n-1)) \in A_{end}$ is denoted as $AI^{\mathcal{F}}$. Therefore, $AI^{\mathcal{F}}$ represents all possible complete traces of business process bp that are valid according to its control flow.*

Definition 3 (Data objects assignment)**.** *Let bp a business process, $D = \{df_1, \ldots, df_n\}$ be the fields of data objects of bp that have data related to resources, and P, R, PS and U be the people, roles, positions and organisational units defined in the organisational model O, we define the assignment of values to the data objects of bp that have data related to resources by means of function $\delta : D \mapsto P \cup R \cup PS \cup U$. The set of all possible assignment of values for the data objects of a business process is denoted as Δ.*

Definition 4 (Process executions)**.** *Let bp be a business process, τ be an execution trace of bp and δ be a data object assignment of bp. A process execution $\sigma = (\tau, \delta)$ is a tuple that includes both its trace and the state of its data objects that have data related to resources. The set of all possible process executions of bp is denoted as $\Sigma = AI^{\mathcal{F}} \times \Delta$.*

For a process execution $\sigma = (\tau, \delta)$, with $\tau = \{(0, ai_x), \ldots, (n-1, ai_y)\}$, we write $ai_j \in \sigma$ if ai_j is an element of the trace in the process execution, and $\sigma(i)$ to refer to the activity instance $\tau(i)$. Moreover, we define $\#_a^\sigma = |\{ai \in \sigma | \pi_a(ai) = a\}|$ as the number of times activity a is executed in the process execution σ

A consequence of this definition is that we assume that the assignment of values to data objects that have data related to resources do not change in a BP execution. Note also that this restriction do not apply to other data objects used in the process that are not involved in resource assignment.

According to these definitions, any person in the organisation can be allocated to any activity of the process. However, this is often not true and there are restrictions concerning who can participate in an activity. These restrictions are specified by the resource selection conditions included in a resource assignment, which can be defined as follows:

Definition 5 (Resource selection condition and resource assignment)**.** *Let O be an organisational model with P persons and bp a business process with A activities:*

[8] This definition is an extension of the one of firing sequence in [21] to include the performer of the activity.

- *A resource selection condition $c \in \mathcal{C}$ is a predicate defined over P and Σ that selects a certain subset of the people in the organisation according to the information present in σ, where $\mathcal{C}$ is the set of all possible resource selection conditions.*
- *A resource assignment is a function that assigns a resource selection condition to the activities of the process: $\rho : A \mapsto \mathcal{C}$. For convenience, we write $\rho^\sigma(a)$ to refer to the people who meet the resource selection condition of activity a according to the information present in σ: $\rho^\sigma(a) = \{p \in P \mid \rho(a) = c \wedge c(p, \sigma)\}$.*

In the following, when we talk about a BP, we assume it includes an specification of its resource assignments, i.e., it is a resource-aware BP.

In a resource-aware BP, a resource allocation defined by a process execution σ is valid if the people allocated to each activity fulfills the restrictions specified in the resource assignment.

Definition 6 (Resource-valid process execution)**.** *Let O be an organisational model and let bp be a business process. A process execution σ of bp is valid with respect to the resource assignment specified by ρ^σ, denoted as $R - valid$ if:*

$$R - valid(\sigma) \Leftrightarrow \forall ai \in \sigma(\pi_p(ai) \in \rho^\sigma(\pi_a(ai)))$$

For convenience, we denote $T = \{\sigma \in \Sigma | R - valid(\sigma)\}$ as the set of all R-valid process executions and $T_a = \{\sigma \in T | \#^\sigma_a > 0\}$ as the set of all R-valid process executions whose trace contains activity a.

5.2. Formalisation of person-activity operations

Building on the previous definitions, the person-activity operations detailed in Section 3 can be formalised as follows:

Definition 7 (Person-activity operations)**.** *Let O be an organisational model with P persons and A be the activities of a business process bp, we define:*

- *The potential participants of an activity a as those people who meet the resource selection conditions of a for some process execution $\sigma \in T$:*

$$PP(a) = \{p \in P \mid \exists \sigma \in T_a(p \in \rho^\sigma(a))\}$$

- *The potential activities of a person p as those activities whose resource selection condition is met by p for some process execution $\sigma \in T$:*

$$PA(p) = \{a \in A \mid \exists \sigma \in T_a(p \in \rho^\sigma(a))\}$$

- *A resource assignment of bp is consistent if for any process execution of bp ($\sigma \in \Sigma$), it is possible to find a $R - valid$ process execution ($\sigma' \in T$) that is activity-equivalent (i.e. the same activities are executed in the same order) with σ:*

$$CC \Leftrightarrow \forall \sigma \in \Sigma(\exists \sigma' \in T(\sigma \equiv^A \sigma'))$$

where $\sigma \equiv^A \sigma'$, if their traces have the same length n and contain exactly the same sequence of executed activities: $\pi_a(\sigma(i)) = \pi_a(\sigma'(i))$ for all $0 \leq i \leq n - 1$

- *The critical participants as those people for which there are one or more activities in the process such that they have to be allocated to some activity instance of any of these activities in any possible execution that involves them:*

$$CP = \{p \in P \mid \exists A' \subseteq A(T_{A'} \neq \emptyset \wedge \forall \sigma \in T_{A'}(\exists a \in A'(p \in R_a^\sigma)))\}$$

where $T_{A'} = T_{a_1} \cup \ldots \cup T_{a_n}$ *with* $A' = \{a_1, \ldots, a_n\}$

- *The critical activities of a person p as those activities whose resource selection condition is only met by p:*

$$CA(p) = \{a \in A \mid \forall \sigma \in T_a(\rho^\sigma(a) = \{p\})\}$$

The non-potential participants and non-potential activities can be trivially defined from the potential participants and the potential activities, respectively. Moreover, α-PP and α-PA can be defined just by considering T^α instead of T, with $T^\alpha = \{\sigma \in T \mid \forall a \in A(\#_a^\sigma \leq 1)\}$, i.e., those $R-valid$ process executions in which all activities are executed at most once.

5.3. RAL-based Resource Assignments

In our proposal, RAL expressions are used to define resource selection conditions, which means that resource selection conditions may depend on either the organisational model (if it contains RAL Org expressions, e.g. `HAS ROLE r1`), the values assigned to data objects (if it contains RAL Data or RAL DataOrg expressions, e.g. `IS PERSON IN DATA FIELD d.f`), or the allocation of people to other activities (if it contains RAL AC expressions, e.g. `IS ANY PERSON RESPONSIBLE FOR ACTIVITY a1`). The last two dependencies determine the influence of a process execution on the people selected by a RAL expression and can be defined with the following relations.

Definition 8 (Data relation)**.** *Let* $A = \{a_1, \ldots, a_n\}$ *be the activities of a process and let* D *be the assignment of values to data fields related to resources. We denote by* $D_a \subseteq D$ *the data fields that are used by the resource assignment of activity* a*. Furthermore, let* $A' \subseteq A$*, then* $D_{A'}$ *is the set of data fields used by any activity* a *in* A'*:* $D_{A'} = \{d \in D_a \mid a \in A'\}$*.*

For instance, if the assignment of a is `IS PERSON IN DATA FIELD d.f`, then $D_a = \{d.f\}$.

Definition 9 (AC-relation)**.** *Let* $A = \{a_1, \ldots, a_n\}$ *be the activities of a business process. The AC relation* $\sim \subseteq A \times A$ *contains all pairs* (x, y) *and* (y, x) *such that the RAL expression of* x *contains an access-control constraint with* y*. Furthermore, we write* $x \nsim y$ *if* $(x, y) \notin \sim$

For instance, in our running example we have that:

$$\begin{aligned}\sim = \{&(RegisterAtConference, SendTA), (SendTA, RegisterAtConference),\\ &(CheckResponse, SubmitCRV), (SubmitCRV, CheckResponse),\\ &(SendTA, FillTA), (FillTA, SendTA)\}\end{aligned}$$

Using this relation, we can partition the set of activities of a business process based on whether they are related by means of an access-control constraint as follows.

Definition 10 (AC-group)**.** *Let $A = \{a_1, \ldots, a_n\}$ be the activities of a business process bp, $\mathcal{P}(A)$ be the power set of A, and $\sim$ be the AC-relation of bp.* AC-groups $\subseteq \mathcal{P}(A)$ *is the set of connected components of the graph defined as* AC-graph $= (A, R)$*, where the nodes are the set of activities A and the edges $R = \{\{x, y\} | x \in A \wedge y \in A \wedge (x \sim y)\}$ represent the fact that two activities are related by means of an access-control constraint. Furthermore, we use $ACg(a)$ for each activity $a \in A$ to refer to the* AC-group *to which activity a belongs.*

Therefore, each activity in an AC-group is AC-related with another activity in the same AC-group and it is not AC-related with any other activity outside from that AC-group. In our example:

$$\begin{aligned} \text{AC-groups} = \{ & \{SubmitCRV, CheckResponse\}, \\ & \{RegisterAtConference, SendTA, FillTA\}, \\ & \{MakeReservations\}, \\ & \{SignTA\}\} \end{aligned}$$

AC-groups are relevant because the influence of the process execution on the people selected by a resource selection condition can be analysed based on it because of two reasons. First, the order in which activities are performed is not relevant from the perspective of resource assignments because they only depend on data objects and the people allocated to activities. This means that, from now on, we can consider a trace τ of length n as a multi-set of AI whose elements are $\tau(i)$ for all $0 \leq i \leq n-1$. Moreover, the number of times an activity is performed is also irrelevant with respect to the people who meet a resource selection condition provided that they are performed by the same set of people.

Second, the people who meet the resource selection condition of an activity are also not influenced by the executions of the activities that belong to a different AC-group because there is no $AC - relation$ between them by definition of AC-group. For instance, in our example, the people who meet the resource selection condition of C is not going to change regardless of who is or may be allocated to other activities of the process. However, the people who meet the resource selection condition of D depend directly on the people allocated to B, and hence, if the set of people allocated to B changes, the set of people who meet the resource selection condition of D changes as well.

5.4. Person-activity operations with R3C-processes

Based on the definition of AC-group, we can formalise the concept of R3C-process that was introduced at the beginning of this section.

Definition 11 (R3C-process)**.** *Let $AC = \{a_1, \ldots, a_n\}$ be an* AC-group *of a process bp, AC is a* AC3C-group *iff:*

- *For all process execution $\sigma \in \Sigma$ of bp, there is not $a_i, a_j \in AC$, such that $\#^{\sigma}_{a_i} \geq 1 \wedge \#^{\sigma}_{a_j} = 0$.*
- *There exists a process execution $\sigma \in \Sigma$ of bp such that for all $a_i \in AC (\#^{\sigma}_{a_i} = 1)$.*

- *If there exists a process execution $\sigma \in \Sigma$ of bp such that $\exists a_i \in AC(\#^{\sigma}_{a_i} > 1)$, then it must exist at least a process execution $\sigma' \in \Sigma$ of bp such that $\exists a_i \in AC(\#^{\sigma}_{a_i} > n)$, with n arbitrarily large.*

An R3C-process *is a process whose* AC-groups *are all* AC3C-groups *and for all process execution $\sigma \in \Sigma$, there is not $a_i, a_j \in A$, such that $D_{a_i} \cap D_{a_j} \neq \emptyset$ and $\#^{\sigma}_{a_i} \geq 1 \wedge \#^{\sigma}_{a_j} = 0$.*

Consequently, if an AC-group AC has only one activity, then AC is an AC3C-group. In our example, all of the AC-groups of the business process are AC3C-groups because: (i) G and *Sign Travel Authorisaton* are the only activities in their group; (ii) in all process instances if A is executed at least once then E is also executed at least once; and (iii) if either F, D or B are executed at least once, then the others are executed at least once as well. Furthermore, for each AC-group there is a valid process instance in which all its activities are executed exactly once, specifically the one that does no take the loop. Therefore the process of our example is an R3C-process.

Note that AC3C-group does not imply that the activities in the same AC-group must be executed the same number of times. For instance, B may be executed an unbounded number of times, but F is executed only once.

The most interesting aspect of R3C-processes is that the person-activity analysis operations can be defined over a tuple of a multi-set of activity instances and assignments of values to data objects $\mathcal{S}$ that do not model the full semantics of the control flow. Specifically, it only needs to identify the activities that are in a loop, which despite being well-known that the general case requires exponential time and space, can be obtained very efficiently for *sound free-choice* systems [22] as demonstrated by Weidlich et al. [23].

Definition 12. *Let A be the activities of a business process, let $A^L \subseteq A$ be the activities of a business process that are in a loop, i.e. $A^L = \{a \in A | \exists \sigma \in \Sigma(\#^{\sigma}_a > 1)\}$. Let AI be the set of all possible activity instances and let Δ be the set of all possible data states. $\mathcal{S}$ is a tuple defined as $\mathcal{S} = \{S \in \mathcal{B}(AI) \times \Delta \mid \forall a \in A(\#^S_a \geq 1) \wedge R - valid(S) \wedge \forall a \in A \setminus A^L(\#^S_a \leq 1)\}$. Note that $\mathcal{B}$ is the set of all multi-sets over AI.*

Both $\mathcal{S}$ and T represent sets of $R - valid$ tuples of multi-sets defined on AI and data states δ. Apart from the order relation in traces, which is not relevant from the perspective of resource assignments as discussed above, there are two main differences between them. The first one is that in $\mathcal{S}$ there must be at least one activity instance for each activity, whereas this does not hold in T, in which one can find a trace where an activity is not executed at all. The second difference lies on the relation between the number of times an activity can be executed. In $\mathcal{S}$ there is no relation at all between the execution of different activities. However, this does not hold in T. For instance, activities that are in sequential order in a loop are always executed the same number of times. Nevertheless, despite these differences, the following theorem holds.

Theorem 1. *For any* R3C-process *bp with A activities whose resource assignment is consistent, it holds that for any $a \in A$, T_a and $\mathcal{S}$ are equivalent with respect to the people who meet the resource selection conditions of an activity, i.e., $\forall \sigma \in T_a(\exists S \in \mathcal{S}(\rho^{\sigma}(a) = \rho^S(a))$ and $\forall S \in \mathcal{S}(\exists \sigma \in T_a(\rho^S(a) = \rho^{\sigma}(a))))$.*

Proof. See Appendix B. □

Based on this result, we can now prove that $\mathcal{S}$ can be used instead of T to compute the potential participants at design-time in R3C-processes.

Corollary 1. *For any* R3C-process *bp whose resource assignment is consistent, it holds that the potential participants of* T *and* $\mathcal{S}$ *at design-time coincide, i.e., for all* $a \in A$, $PP(a) = \{p \in P | \exists s \in \mathcal{S}(\#_a^s \wedge p \in \rho^S(a))\}$.

Proof. The potential participants are defined as $PP(a) = \{p \in P | \exists \sigma \in T_a(p \in \rho^\sigma(a))\}$. According to Theorem 1 we have that for any $a \in A$ we can use $\mathcal{S}$ instead of T_a and the set of people who meet the resource selection condition ($\rho^\sigma(a)$) does not change. Therefore, the potential participants can be defined as $PP(a) = \{p \in P | \exists S \in \mathcal{S}(p \in \rho^s(a))\}$ □

A similar proof can be done for the non-potential participants, the potential activities, the non-potential activities, the critical activities and the critical participants.

Concerning consistency checking, we have to introduce first the notion of an α-consistent process as follows.

Definition 13 (α-consistency)**.** *A process with A activities is* α-consistent *if there is an element of* $\mathcal{S}$ *with exactly one activity instance for all of the activities, i.e.,* $\exists S \in \mathcal{S}(\forall a \in A(\#_a^S = 1))$

The interesting aspect of α-consistency is that in R3C-processes it is equivalent to normal consistency.

Theorem 2. *For any* R3C-process *bp, it holds that bp is consistent* $\Leftrightarrow$ *bp is* α-consistent

Proof. See Appendix B. □

As a result, checking the consistency of a process can be reduced to checking its α-consistency.

5.5. Extension to several task duties

A resource assignment with several task duties can be defined as follows:

Definition 14 (Resource assignment with several task duties)**.** *Let* O *be an organisational model with* P *persons, bp a business process with* A *activities,* TD *the set of all possible task duties and* $\mathcal{C}$ *be the set of all possible resource selection conditions. A resource assignment with several task duties is a partial function that assigns a resource selection condition to a pair activity-task duty:* $\rho : A \times TD \mapsto \mathcal{C}$.

Based on this definition, the results presented above can be easily extended to resource assignments with several task duties. Let *bp* be a business process whose resource assignment ρ involves different task duties. We just have to build a new process *bp′* such that each activity a of *bp* is substituted by several activities a_d that are executed sequentially, one for each task duty d such that $\rho(a, d)$ is defined. For instance, if *bp* has an activity C with resource assignment defined for two task duties *responsible* and *accountable*, in *bp′*

Axiom	DL Syntax	Semantics
Subconcept	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
Equivalent concept	$C_1 \equiv C_2$	$C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$
Disjoint with	$C_1 \sqsubseteq \neg C_2$	$C_1^{\mathcal{I}} \neq D_2^{\mathcal{I}}$
Same Individual	$u_1 \doteq u_2$	$\{a \in \triangle^{\mathcal{I}} \mid \exists b \in \triangle^{\mathcal{I}}.u_1^{\mathcal{I}}(a) = b = u_2^{\mathcal{I}}(a)\}$
Different from	$u_1 \not\doteq u_2$	$\{a \in \triangle^{\mathcal{I}} \mid \exists b_1, b_2 \in \triangle^{\mathcal{I}}.u_1^{\mathcal{I}}(a) \neq b = u_2^{\mathcal{I}}(a)\}$
Subproperty	$P_1 \sqsubseteq P_2$	$\{a \in \triangle^{\mathcal{I}} \mid \forall b.(a,b) \in P_1^{\mathcal{I}} \rightarrow (a,b) \in P_2^{\mathcal{I}}\}$
Equivalent property	$P_1 \equiv P_2$	$\{a \in \triangle^{\mathcal{I}} \mid \forall b.(a,b) \in P_1^{\mathcal{I}} \leftrightarrow (a,b) \in P_2^{\mathcal{I}}\}$
Inverse	P^-	$\{(b,a) \in \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}} \mid (a,b) \in P^{\mathcal{I}}\}$

Table 2: DL axioms

two activities are added instead, namely $C_{\text{responsible}}$ and $C_{\text{accountable}}$. The resource assignment ρ' of *bp*′ for these new activities is defined as $\rho'(C_{\text{responsible}}) = \rho(C, \text{responsible})$ and $\rho'(C_{\text{accountable}}) = \rho(C, \text{accountable})$, respectively.

Because *bp*′ is itself a business process, all of the results presented above can be applied for *bp*′ as well. For instance, the potential participants operation for several task duties is defined as $PP(a, d) = \{p \in P \mid \exists \sigma \in T_{a_d}(p \in \rho^{\sigma}(a_d))\}$.

6. DL Semantics of Resource Assignments with RAL

According to the formalisation principles defined by Hofstede and Proper [24], the selection of the style and target domain to formalise a language should be driven by the goal pursued with the formalisation (*Primary Goal Principle*). In our case, we propose a formalisation based on a semantic mapping to Description Logics (DLs) [25] with the primary objective of establishing a sound basis for sophisticated automated support. DL is a decidable subset of First Order Logic (FOL) that serves primarily for formal descriptions of *concepts*, *properties*[9] (relations between concepts), and *individuals* (instances of the concepts). In particular, a Knowledge Base (KB) comprises two components, the *TBox* and the *ABox*. The TBox describes *terminology*, i.e., the KB in the form of *concepts* and *property* definitions, and their relations; the ABox contains *assertions* about individuals using the terms from the TBox.

As exemplified in Tables 2 and 3, DLs have a rich set of knowledge representation constructs that can be used to formally specify knowledge about the BP resource perspective, which in turn can be exploited by DL reasoners for inference purposes, i.e., for deductively inferring new facts from knowledge that is explicitly available [26]. In particular, in the tables, C_i denotes a concept description, P_i denotes a property, and u_i denotes an individual. A is typically used to refer to atomic concepts. An *interpretation* $\mathcal{I}$ consists of a non-empty set $\triangle^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function that assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$ and to every atomic property P a binary relation $P^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$.

[9]They are also called *roles*, but we use *properties* because it is common in FOL and helps us avoid confusion.

Constructor	DL Syntax	Semantics
Universal, top	$\top$	$\triangle^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Intersection	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
Union	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
Negation	$\neg C$	$\triangle^{\mathcal{I}} \setminus C^{\mathcal{I}}$
All values from	$\forall P.C$	$\{a \in \triangle^{\mathcal{I}} \vert \forall b.(a,b) \in P^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
Some values	$\exists P.C$	$\{a \in \triangle^{\mathcal{I}} \vert \exists b.(a,b) \in P^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
Max cardinality	$\leq nP$	$\{a \in \triangle^{\mathcal{I}} \vert \vert\{b \in \triangle^{\mathcal{I}} \vert (a,b) \in P^{\mathcal{I}}\}\vert \geq n\}$
Min cardinality	$\geq nP$	$\{a \in \triangle^{\mathcal{I}} \vert \vert\{b \in \triangle^{\mathcal{I}} \vert (a,b) \in P^{\mathcal{I}}\}\vert = n\}$
Qualified at-most restriction	$\leq nP.C$	$\{a \in \triangle^{\mathcal{I}} \vert \vert\{b \in \triangle^{\mathcal{I}} \vert (a,b) \in P^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\vert \leq n\}$
Qualified at-least restriction	$\geq nP.C$	$\{a \in \triangle^{\mathcal{I}} \vert \vert\{b \in \triangle^{\mathcal{I}} \vert (a,b) \in P^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\vert \geq n\}$

Table 3: DL concept constructors

There are two reasons to choose DLs as a formalisation mechanism for RAL. First, RAL expressions can be observed as a way to specify a subset of the people of an organisation by defining a set of conditions they must satisfy (e.g., *HAS ROLE Researcher*). This way of defining RAL expressions fits nicely into the way DLs express their concepts, and hence, they provide a very natural way to describe the problem. This allows the *Semantics Priority Principle* [24] to be followed. Furthermore, this makes it easier to avoid unnecessary representational choices, as suggested by the *Conceptualisation Principle* [24]. Consequently, we can define RAL Core semantics and then extend them for RAL Org, RAL Data, RAL DataOrg, and RAL AC without modifying the essence. The second reason for choosing DLs is that there is a plethora of off-the-shelf DL reasoners that can be used to automatically analyse RAL expressions and, thus, to automatically infer information from them. This stems from the fact that the semantics of the W3C recommendation Web Ontology Language (OWL) 2 [27] to express ontologies for the semantic web are defined in DLs, and hence, many tools have been developed in the last few years to support a variety of semantic web use cases.

To formalise RAL using DLs, the organisational and BP models both have to be mapped into DL elements as well as RAL expressions themselves and the resource assignments. Thus, although there are a significant number of concepts in the problem domain, we have tried to keep the number of concepts in the formalisation as small as possible, as suggested by the *Orthogonality Principle* [24], which encourages one to keep a one-to-one relation between semantic concepts and domain concepts.

Next, we describe every mapping in detail, divided into four groups: the mapping of the organisational information, the mapping of the BP elements, the mapping of RAL expressions into DL concept descriptions, and the mapping of the resource assignments. For all the DL expressions, we use a syntax commonly used for DLs [28] (cf. Tables 2 and 3).

6.1. Mapping the Organisational Information

To map the organisational metamodel into DLs, one *concept* is added to the TBox for each and every class included in the metamodel (cf. Figure 5). We keep the same names for the sake of understanding. Hierarchies are also included in the TBox by using the

Property	Subproperty Of	From	To	Property Type
occupies		Person	Position	
participatesIn		Position	Role	
isMemberOf		Position	OrganisationalUnit	
reportsTo	extendedReportsTo	Position	Position	Functional
extendedReportsTo				Transitive
canDelegateWorkTo		Position	Position	Transitive
hasCapability		Person	Capability	
hasDegree	hasCapability			
hasExperience	hasCapability			

Table 4: Properties in the TBox related to organisational information

conceptInclusion axiom that DLs provide. Data properties are added for the classes that contain attributes. For example, capabilities can have their own properties, e.g., a *Degree* has a property value of standard type *xsd:string*, and capability *Experience* has fields "years" of type *xsd:integer* and "topic" of type *xsd:string*.

The explicit relations among the classes of the metamodel are mapped into *properties* of the TBox, i.e., the properties *hasCapability*, *occupies*, *reportsTo*, *participatesIn*, and the like are added to the TBox (cf. Table 4). *Cardinality* must be configured for all the properties according to the relations in the organisational metamodel. If the cardinality is less than or equal to 1, the property is defined as *functional*. Otherwise, an axiom to specify the cardinality is added. For instance, to specify that "a Person occupies one or more Positions", axiom $Person \sqsubseteq \geq 1\ occupies.Position$ is added. The information about cardinality has not been included in Table 4 for the sake of readability.

As seen in the table, the hierarchical relations among the positions of an organisation have received special treatment. Specifically, a *superproperty extendedReportsTo* has been created to make the property corresponding to the *reportsTo* relation *transitive*. This enables defining assignments such as "activity C can only be performed by a person who is reported by somebody reported by a person who occupies the position *PhD Student*". However, there is no functional variant of the property *canDelegateWorkTo* as the relation in the organisational metamodel is N:M.

Once the organisational metamodel is mapped into the KB, it is possible to map specific organisational models. The elements of a model are defined as *individual assertions* in the ABox. Thus, each *specific* person, role, position, organisational unit, and capability is added to the ABox and associated with the corresponding concept of the TBox. For instance, the following DL assertion specifies that *Principal Investigator* (ABox) is a *Role*.

$$Role(PrincipalInvestigator)$$

Furthermore, all individuals are defined as disjoint from each other because DLs do not assume it:

$$PrincipalInvestigator \neq ProjectAccountAdministrator \neq ... \neq ProjectResourceManager$$

The relations among elements are defined using *equivalence axioms* (cf. Table 2). For instance, the relation between *Post-Doc Researcher* and the roles it participates in is defined as:

$$\exists participatesIn^{-}.\{\text{PostDocResearcher}\} \equiv \{\text{ProjectStaffMember}, \text{Researcher}\}$$

The reason for using equivalence axioms instead of property assertions is to avoid the *open world assumption* in DLs [28]. The open world assumption consists of assuming that the information in the KB may be incomplete, and hence, the absence of a property assertion does not imply the fact being false. However, in our case, we assume that the information defined in the organisational model is complete.

6.2. Mapping Business Process Information

From an abstract point of view, the goal of the KB concerning the modelling of BP information is that each KB models the execution of one process instance. Consequently, the TBox includes two concepts (*Activity* and *DataObject*) that represent the elements from the BP model and two concepts (*ActivityInstance* and *DataObjectInstance*) that represent the instances of activities and data objects that appear in the process instance during execution. All these concepts are disjoint with each other. The two sets of concepts are related by means of functional property $isOfType$, whose domain is *ActivityInstance* and *DataObjectInstance* and whose range is *Activity* and *DataObject*, respectively. Allocations are modelled by means of property *hasDuty*, which is a super-property for all the task duties defined for a specific BP model and relates an *ActivityInstance* with the concept *Person* from the organisational model. All of these concepts and properties are generic and appear in every TBox regardless of the BP model that is being mapped into the KB.

Concerning the elements that are specific to a BP model *bp*, Algorithm 1 shows the axioms and assertions that must be added to the KB. First, the algorithm adds properties to the KB for each task duty included in the assignment (lines 4–6). Then, the individuals of *Activity* and a subconcept of *ActivityInstance* for each activity in the BP are added (lines 7–11). After that, the individuals of *DataObject* and a subconcept of *DataObjectInstance* that represents all the instances for each data object used in the process are added. Finally, DL properties are added for each relevant property of the data object, i.e., those that refer to people, roles, positions, or organisational units (lines 12–20).

Three of the axioms added deserve specific attention. The first is axiom $\{do\} \sqsubseteq = 1 isOfType^{-}$ in line 16, which is added to follow the assumption made by BPMN [9] and many other process modelling notations that in a process instance there is just one instance for each data object. The other two are the different individual axioms of lines 8 and 13, which are the usual way to axiomatize the unique name assumption in DLs [27]. An alternative that avoids the enumeration of all individuals is to give unique names to activities and data objects by means of a data property and use a key axiom to state that all individuals of *Activity* (resp. *DataObject*) are uniquely identified by such data property [29].

One characteristic of this mapping is that all possible process executions of the BP can be modelled with the KB. Specifically, let *bp* be a BP extended for different task duties as detailed in Section 5.5, σ be a process execution of *bp*, and $\pi_{ac}(ai^{bp}) = a$ and $\pi_d(ai^{bp}) = d$ be the activity (resp. the task duty) of $ai^{bp} \in \sigma$, i.e., $a_d = \pi_a(ai^{bp})$. The process execution σ can be modelled as follows:

Algorithm 1 This algorithm maps a set of activities A, task duties TD, and data objects DO of a business process bp to the DL-based KB.

```
IN: A^bp, TD^bp, DO^bp the set of activities, task duties and data objects of process bp
IN: KB a DL knowledge base
OUT: KB updated with the corresponding axioms and assertions
for all task duty d^bp ∈ TD^bp do
    add property d as subproperty of hasDuty with domain ActivityInstance and range Person
end for
add axiom Activity ≡ {a_1, ..., a_n} for all activity a_i^bp ∈ A^bp
add axiom stating that all activities a are different from each other
for all activity a^bp ∈ A^bp do
    add axiom AI_a ≡ ∃isOfType.{a} ⊓ ActivityInstance
end for
add axiom DataObject ≡ {do_1, ..., do_n} for all data objects do_i^bp ∈ DO^bp
add axiom stating that all data objects do are different from each other
for all data object do^bp ∈ DO^bp do
    add axiom DOI_do ≡ ∃isOfType.{do} ⊓ DataObjectInstance
    add axiom {do} ⊑= 1isOfType^-
    for all property f of data object do referred to a person (resp. position, role, unit) do
        add property f with domain DOI_do and range Person (resp. Position, Role, Unit)
    end for
end for
```

1. Adding an assertion $AI_{\pi_{ac}(ai^{bp})}(ai)$ for each $ai^{bp} \in \sigma$. This assertion adds an individual to the ABox of the KB called ai with the same type of activity as ai^{bp}.
2. Adding a property assertion $d(ai, \pi_p(ai^{bp}))$ for each $ai^{bp} \in \sigma$, where $d = \pi_d(ai^{bp})$ is the task duty performed by $\pi_p(ai^{bp})$. This assertion adds to the KB the information about the resource allocation of ai^{bp}.
3. Adding a property assertion $f(do, x)$ for each $do_f^{bp} \in D$, where D is the fields of data objects of the process that have data related to resources and $x = \delta(do_f^{bp})$.

Note that the KB also models other executions that are not allowed in the BP. For instance, in our example, an execution without any activity instance of F would be valid in the KB, but not in the BP.

6.3. Mapping RAL Expressions and Constraints

A RAL expression defines the conditions that must be met for each task duty involved in an activity. Consequently, a subset of all the people in the organisation is selected to become potential performers of the task duty for the activity. This idea can be naturally expressed in DLs by mapping each RAL expression to a DL concept description that is a subconcept of $Person$. This mapping is formalised by means of the following definition.

Definition 15 (RAL expression mapping)**.** *Let RAL be the set of all possible RAL expressions and constraints and DL be the set of all possible concept descriptions in DLs.* $\phi : RAL \mapsto DL$ *is a function that maps RAL expressions and constraints to their corresponding concept description in DLs and is defined as shown in Table 5.*

The mapping specified by ϕ makes the following assumptions:

1. The type of data fields used in RAL Data expressions contain valid references to the organisational model and is coherent with the type of resource expected in the RAL Data expression, i.e., if the expression is `IS PERSON IN DATA FIELD d.f`, the value of data field f of data object d must be the name of a person who belongs to the organisation.
2. The people selected by RAL AC expressions such as `IS ANY PERSON responsible for ACTIVITY a` are all people who have performed activity a with the task duty *responsible for*. Therefore, if a is in a loop and is executed more than once, any of the performers of the corresponding task duty in a are selected by this RAL expression.

Finally, note that ϕ is not a DL construct but an auxiliary function that we use outside the context of DLs to make the description of the mapping more readable. Furthermore, for the sake of brevity, not all of the possible expressions and constraints that can be defined are included in Table 5, where the first column indicates in which RAL module the type of expression or constraint (second column) is defined (cf. RAL Specification in Section 4), the third column contains a subset of all the possible RAL expressions and constraints, and the last column shows the description in DLs. In Figure 6, we provide the DL concept descriptions for the RAL expressions shown in Figure 3.

6.4. Mapping Resource Assignments with RAL

An allocation of a person p to an activity instance i_a of activity a for a task duty d can be easily represented in the DL-based KB as a property assertion $d(i_a, p)$. Therefore, a resource assignment of a for d, $\rho(a, d)$, can be modelled as an axiom that states that all activity instances (AI_a) of a must have as performers for task duty d only people who fulfil the RAL expression specified in $\rho(a, d)$. Because the result of the mapping ϕ defined in the previous section is a subconcept of $Person$ that represents all the people who fulfil the given RAL expression, the axiom can be written as:

$$AI_a \sqsubseteq \forall d.\phi(\rho(a, d))$$

In addition, together with this axiom, it is necessary to state that if activity a has a resource assignment defined for task duty d, then all activity instances of a have exactly one person as performer for task duty d:

$$AI_a \sqsubseteq = 1\, d.Person$$

However, if activity a does not have a resource assignment defined for task duty d, then all activity instances of a must not have any performer for task duty d:

$$AI_a \sqsubseteq = 0\, d.Person$$

Algorithm 2 shows how these axioms can be automatically added to the KB from a resource assignment ρ.

RAL	**Expression Type**	**RAL Expression** ($expr$)	**DL Concept Description** ($\phi(expr)$)
Core		ANYONE	$Person$
	PersonExpr	IS pc	$\phi(pc)$
	DenyExpr	NOT (expr)	$Person \sqcap \neg\phi(expr)$
	CompoundExpr	(expr1) AND (expr2)	$\phi(expr1) \sqcap \phi(expr2)$
		(expr1) OR (expr2)	$\phi(expr1) \sqcup \phi(expr2)$
Org	GroupResourceExpr	HAS POSITION poc	$\exists occupies.\phi(poc)$
		HAS UNIT uc	$\exists occupies.(\exists isMemberOf.\phi(uc))$
		HAS ROLE rc	$\exists occupies.(\exists participatesIn.\phi(rc))$
		HAS ROLE rc IN UNIT uc	$\exists occupies.(\exists participatesIn.\phi(rc) \sqcap \exists isMemberOf.\phi(uc))$
	CommonalityExpr	SHARES SOME POSITION WITH pc	$\exists occupies.(\exists occupies^{-}.\phi(pc))$
		SHARES ALL UNIT WITH pc	$\exists occupies.(\exists isMemberOf.(\forall isMemberOf^{-}.(\exists occupies^{-}.\phi(pc))))$
		SHARES SOME ROLE WITH pc	$\exists occupies.(\exists participatesIn.(\exists participatesIn^{-}.(\exists occupies^{-}.\phi(pc))))$
		SHARES ALL ROLE IN UNIT uc WITH pc	$\exists occupies.(\exists participatesIn.(\exists participatesIn^{-}.isMemberOf.(\phi(uc)) \sqcap \forall participatesIn^{-}.(\exists occupies^{-}.\phi(pc))))$
	CapabilityExpr	HAS CAPABILITY cc	$\exists hasCapability.\phi(cc)$
	ReportExpr	REPORTS TO POSITION poc	$\exists occupies.(\exists extendedReportsTo.poc)$
		IS REPORTED BY POSITION poc	$\exists occupies.(\exists extendedReportsTo^{-}.poc)$
		DIRECTLY REPORTS TO pc	$\exists occupies.(\exists reportsTo.(\exists occupies^{-}.(\phi(pc)))$
		IS DIRECTLY REPORTED BY pc	$\exists occupies.(\exists reportsTo^{-}.(\exists occupies^{-}.(\phi(pc)))$
	DelegateExpr	CAN HAVE WORK DELEGATED BY pc	$\exists occupies.(\exists canDelegateWorkTo^{-}.(\exists occupies^{-}.\phi(pc)))$

RAL	**Constraint Type**	**RAL Constraint** ($constr$)	**DL Concept Description** ($\phi(constr)$)
Core	PersonConstr	IS personName	$\{personName\}$
Org	PositionConstr	POSITION positionName	$\{positionName\}$
	RoleConstr	ROLE roleName	$\{roleName\}$
	UnitConstr	UNIT unitName	$\{unitName\}$
	CapabilityConstr	capabilityID	$\{capabilityID\}$
		cap = val	$Person \sqcap \exists cap.\{val\}$
Data	PersonConstr	PERSON IN DATA FIELD do.f	$\exists f^{-}.(DOI_{do})$
DO	PositionConstr	POSITION IN DATA FIELD do.f	$\exists f^{-}.(DOI_{do})$
AC	PersonConstr	ANY PERSON duty ACTIVITY a	$\exists duty^{-}.(AI_{a})$

Table 5: Mapping of RAL expressions and constraints to DL concept descriptions

Submit

Camera Ready Version (A). `(HAS ROLE Researcher IN UNIT HRMS) OR`
`(HAS ROLE ProjectStaffMember IN UNIT HRMS)`
$\exists occupies.(\exists participatesIn.\{Researcher\} \sqcap \exists isMemberOf.\{HRMS\}) \sqcup$
$\exists occupies.(\exists participatesIn.\{ProjectStaffMember\} \sqcap \exists isMemberOf.\{HRMS\})$

Fill Travel Authorisation (B). `HAS ROLE Researcher IN UNIT HRMS`
$\exists occupies.(\exists participatesIn.\{Researcher\} \sqcap \exists isMemberOf.\{HRMS\})$

Sign Travel Authorisation (C). `HAS POSITION ProjectCoordinator`
$\exists occupies.\{ProjectCoordinator\}$

Send Travel Authorisation (D). `IS ANY PERSON responsible for ACTIVITY FillTA`
$\exists responsibleFor^{-}.(AI_{FillTA})$

Check Response (E). `(HAS UNIT HRMS) AND (SHARES SOME POSITION WITH ANY PERSON`
`responsible for ACTIVITY SubmitCRV)`
$\exists occupies.(\exists isMemberOf.\{HRMS\}) \sqcap \exists occupies.(\exists occupies^{-}.(\exists responsibleFor^{-}.(AI_{SubmitCRV})))$

Register at Conference (F). `(IS ANY PERSON responsible for ACTIVITY SendTA) AND`
`(HAS POSITION PhDStudent)`
$\exists responsibleFor^{-}.(AI_{SendTA}) \sqcap \exists occupies.\{PhDStudent\}$

Make Reservations (G). `(HAS ROLE Clerk) OR (IS PERSON RESPONSIBLE FOR ACTIVITY`
`MakeReservations IN ANOTHER INSTANCE)`
$\exists occupies.(\exists participatesIn.\{Clerk\}) \sqcup \exists h^{-}_{responsibleFor}.\{MakeReservations\}$

Figure 6: DL concept descriptions for the RAL expressions shown in Figure 3

7. Automated Analysis of the Resource Perspective

The approach we follow to provide a DL-based reference implementation for each person-activity operation is based on the results detailed in Section 5. It involves using the mappings described in Section 6 to model the organisational model, the business process and the resource assignment as a DL-based KB and then expressing the analysis operations in terms of standard DL reasoning operations, which are implemented by existing off-the-shelf DL reasoners. Our goal is not to provide the most efficient implementation of every operation but an implementation that can be used as a reference for the development of more efficient implementations for some of these operations, which could be done using other formalisms or ad-hoc algorithms.

7.1. A DL-Based KB for Analysis Operations

Before defining the analysis operations in terms of standard DL reasoning operations, it is necessary to introduce the DL-based KB that will be used.

Definition 16 (DL-based knowledge base $\mathcal{K}_C$). *Let O be an organisational model, bp be a business process, and ρ be a resource assignment for the activities of bp. $\mathcal{K}_C$ is a DL-*

Algorithm 2 This algorithm maps a resource assignment ρ for a business process bp to the DL-based KB. ϕ is the mapping of RAL expressions detailed in Section 6.3.

IN: ρ **a resource assignment,** bp **a business process**
IN: KB a DL-based knowledge base
for all activity a^{bp} **in the business process** bp **do**
    for all task duty d^{bp} **in the task duties of** bp **do**
        if is defined $\rho(a^{bp}, d^{bp})$ **then**
            add axiom $AI_a \sqsubseteq \forall d.\phi(\rho(a^{bp}, d^{bp}))$ **to KB**
            add axiom $AI_a \sqsubseteq = 1\, d.Person$ **to KB**
        else
            add axiom $AI_a \sqsubseteq = 0\, d.Person$ **to KB**
        end if
    end for
end for

based KB obtained after mapping the elements of O, bp, and ρ into DLs using the mappings described in Section 6 and including the following axioms:

1. *For every activity a in the business process that is not in a loop: $\{a\} \sqsubseteq \leq 1 isOfType^{-}$*
2. *For every activity a in the business process: $\{a\} \sqsubseteq \geq 1 isOfType^{-}$*

With these two axioms, $\mathcal{K}_C$ is defined so that it models the set of tuples $\mathcal{S}$ (cf. Definition 12). Specifically, the first axiom restricts the KB to take into account the fact that activities that are not in a loop should have only one activity instance in each BP instance. Thus, it models the third condition of $\mathcal{S}$. The second axiom models the first condition of $\mathcal{S}$ by assuming that all activities are executed at least once. Finally, it is not necessary to explicitly include the second condition of $\mathcal{S}$, which imposes that all its elements are $R-valid$ because, by definition, the only valid activity instances in $\mathcal{K}_C$ are those that are $R-valid$.

7.2. Person-Activity Analysis Operations in DL

Equipped with the KB $\mathcal{K}_C$, the person-activity analysis operations can be formulated in terms of standard DL reasoning tasks that are implemented by most DL reasoners. In particular, the following DL reasoning tasks are used.

- Concept subsumption, which is the problem of deciding whether a concept C_1 is subsumed by another concept C_2 with respect to a KB $\mathcal{K}$. In particular, we are interested in obtaining all concepts that are subsumed by a concept C_1 and denote this reasoning task as $subconcepts_{\mathcal{K}}$.

- Concept retrieval, which is the problem of computing the set containing exactly every instance of a concept C with respect to a KB $\mathcal{K}$. We denote this reasoning task as $individuals_{\mathcal{K}}$.

- Consistency, which is the problem of deciding whether a KB $\mathcal{K}$ is consistent. We denote this reasoning task as $consistent_{\mathcal{K}}$.

7.2.1. Basic Person-Activity Analysis Operations

The non-participants of an activity a for task duty d are those people p for which there is no $i_a \in AI_a$ such that $d(i_a, p)$, i.e., those people p such that $p \in Person \sqcap \neg\exists d^{-}.AI_a$. This corresponds to the concept retrieval reasoning task, and hence, the non-participants operation can be expressed in terms of a DL reasoner as follows:

$$NP(a^{bp}, d^{bp}) = individuals_{\mathcal{K}_C}(Person \sqcap \neg\exists d^{-}.AI_a)$$

Having the non-participants of an activity a for a task duty d, the potential participants of a for task duty d can be obtained as those people who are not non-participants of a for task duty d because for any person p and task duty d, it holds that $PP(a,d) \cup NP(a,d) \equiv Person$, and $PP(a,d) \cap NP(a,d) = \emptyset$.

The same approach can be followed for the operations that obtain the activities in which a person can participate. The non-potential activities of a person p for task duty d are those activities for which there is no $i_a \in AI_a$ such that $d(i_a, p)$. Therefore, an activity a is a non-potential activity of a person p regarding a task duty d if its activity instances $AI_a \sqsubseteq ActivityInstance \sqcap \neg\exists d.\{p\}$. This corresponds with the concept subsumption reasoning task as follows:

$$NPA(p^{bp}, d^{bp}) = subconcepts_{\mathcal{K}_C}(ActivityInstance \sqcap \neg\exists d.\{p\})$$

Finally, similar to potential participants, the potential activities of a person p for a task duty d can be obtained as those activities of the process that are not amongst its non-potential activities.

Apart from these four operations, there are situations, such as those discussed in Section 3, in which it is convenient to consider that each activity of the process is executed only once, i.e., loops are executed only once. This fact can be modelled as described in the following definition.

Definition 17 (DL-based knowledge base $\mathcal{K}^1_C$)**.** *Let O be an organisational model and bp be a business process, $\mathcal{K}^1_C$ is a DL-based KB obtained after adding to $\mathcal{K}_C$ the axiom $\{a\} \sqsubseteq = 1 isOfType^{-}$ for every activity a.*

The intuitive effect of adding these axioms is that it limits the number of activity instances per BP instance to one. Therefore, because $\mathcal{K}_C$ models $\mathcal{S}$, $\mathcal{K}^1_C$ models $\{S \in \mathcal{S} \mid \forall a \in A(\#^S_a = 1)\}$, where A is the set of activities of the business process. Consequently, α-NP (resp. α-PP, α-NPA and α-PA) can be defined exactly the same as NP (resp. PP, NPA and PA) but using $\mathcal{K}^1_C$ instead of $\mathcal{K}_C$. For instance:

$$\alpha\text{-NP}(a^{bp}, d^{bp}) = individuals_{\mathcal{K}^1_C}(Person \sqcap \neg\exists d^{-}.AI_a)$$

7.2.2. Consistency Checking Person-Activity Operations

According to Theorem 2, checking the consistency of a BP is equivalent to checking its α-consistency. Next, we show that the α-consistency of a process can be computed by checking the consistency of $\mathcal{K}^1_C$ as detailed by the following property.

Lemma 1. *If the mapping to DL of both the organisational model and the business process model are consistent, for any* R3C-process *bp with A activities, it holds that bp is* α-consistent $\Leftrightarrow$ $\mathcal{K}_C^1$ *is consistent.*

Proof. $\Rightarrow$ Let *bp* be α-consistent and assume $\mathcal{K}_C^1$ is inconsistent. Because the mapping to DL of both the organisational model and the business process model are consistent, the only reason $\mathcal{K}_C^1$ is inconsistent is because of a contradiction caused by the three axioms that are added to those mappings by $\mathcal{K}_C^1$, namely:

$$\begin{aligned} AI_a &\sqsubseteq = 1d.Person \\ AI_a &\sqsubseteq \forall d.\phi_{bp}(\rho_{bp}(a^{bp}, d^{bp})) \\ \{a\} &\sqsubseteq = 1isOfType^{-}.AI_a \end{aligned}$$

However, because *bp* is α-consistent, for each activity a of *bp* there is a person p such that $d(i_a, p)$, and $isOfType(i_a, a)$ holds. This satisfies the three axioms and, hence, yields a contradiction with $\mathcal{K}_C^1$ inconsistent.

$\Leftarrow$ We shall prove its contraposition, i.e., *bp* not α-consistent $\Rightarrow$ $\mathcal{K}_C^1$ is not consistent. If *bp* is not α-consistent, it means that $\{S \in \mathcal{S} \mid \forall a \in A(\#_a^S = 1)\}$ is empty, i.e., there is some activity x for which there is no person p such that $d(i_x, p)$, and $i_x \in AI_x$. However, from Section 6.4 we have that for each activity a with a resource assignment it holds that $AI_a \sqsubseteq = 1d.Person$, making AI_a insatisfiable. Furthermore, because in $\mathcal{K}_C^1$, as in $\mathcal{K}_C$, we have that for every activity a in the BP there is at least one activity instance ($\{a\} \sqsubseteq \geq 1isOfType^{-}.AI_a$), then AI_a insatisfiable makes $\mathcal{K}_C^1$ inconsistent. □

Consequently, the consistency checking operation can be expressed in terms of the consistency reasoning task as follows:

$$CC \Leftrightarrow consistent_{\mathcal{K}_C^1}$$

7.2.3. Criticality Checking Person-Activity Operations

The two criticality checking person-activity operations can be defined in terms of DL reasoning tasks as follows. A person p is a critical participant for task duty d if there is a subset of activities in the process such that p has to be allocated to task duty d of some activity instance of any of these activities in any possible execution that involves any of them. In other words, a person p is critical if $\mathcal{K}_C$ entails that p participates with task duty d in some activity instance of the process $\mathcal{K}_C \models p \in \exists d^{-}.ActivityInstance$, which can be easily computed using a DL reasoner by means of the concept retrieval reasoning task:

$$CP(d^{bp}) \equiv individuals_{\mathcal{K}_C}(\exists d^{-}.ActivityInstance)$$

An activity a is critical for person p and task duty d if p is the only person who can perform task duty d in activity a. In other words, a is critical if $AI_a \sqsubseteq \exists d.\{p\}$. Therefore, to obtain all critical activities of a person, the concept subsumption reasoning task can be used as follows:

$$CA(p^{bp}, d^{bp}) \equiv subconcepts_{\mathcal{K}_C}(\exists d.\{p\})$$

7.3. Considerations about RAL Data and RAL DataOrg

A particular aspect of RAL expressions that include RAL Data or RAL DataOrg is that there is no possible way of controlling *a priori* which value will have a data field because it might be a human user who decides it. This could lead to potential consistency issues in the resource assignment.

The typical approach to facing this type of situation is defining a validation function that checks whether the value used in the data object is valid. In our case, the validation function is the Consistency Checking operation. Therefore, to check whether the value v for field f of the data object do is valid, a data object instance i_{do} and the assertion $f(i_{do}, v)$ must be added to $\mathcal{K}_C^1$. Then, the Consistency Checking operation can be used to check whether there is a possible allocation for this value v.

In many cases, it is very convenient to know not only whether a value is valid or not but all the possible valid values so that the user only has to choose one value amongst them. To do so, we can follow exactly the same approach used to obtain the potential participants of an activity. Therefore, if DO represents all data object instances of data object do such that $i_{do} \in DO$, one can use $instances(Person \sqcap \neg\exists f^{-}.DO)$ to obtain all the people who cannot be in the data object instance i_{do} for field f. Consequently, all the remaining people of the organisation can be in the data field.

Furthermore, the same approach can be used to check the possible values for group resources for RAL DataOrg. For instance, the reasoning operation to obtain the roles that cannot be in the data object instance for field f would be $instances(Role \sqcap \neg\exists f^{-}.DO)$.

8. Evaluation

In the following, we report on the evaluation of RAL and of the implementation of the seven analysis operations.

8.1. RAL Expressiveness

One of our greatest concerns when developing RAL was to make it expressive as well as automatable. The WRPs have been used as a reference framework to assess the expressiveness of a number of proposals pursuing the same goal as RAL [30, 31, 6, 10, 32]. We specifically use the creation patterns for such evaluation, as they are the patterns related to resource selection. These patterns, as defined in [20], include *Direct Allocation*, i.e., the ability to specify at design time the identity of the resource that will execute a task; *Role-Based Allocation*, i.e., the ability to specify at design time that a task can only be executed by resources that correspond to a given role; *Deferred Allocation*, i.e., the ability to defer specifying the identity of the resource that will execute a task until runtime; *SoD*, i.e., the ability to specify that two tasks must be allocated to different resources in a given BP instance; *Case Handling*, i.e., the ability to allocate the activity instances within a given process instance to the same resource; *Retain Familiar*, i.e., the ability to allocate an instance within a given BP instance to the same resource that performed a preceding activity instance, when several resources are available to perform an activity instance; *Capability-Based Allocation*, i.e., the ability to offer or allocate instances of an activity to resources based on specific capabilities they possess; *History-Based Allocation*, i.e., the ability to offer or allocate activity instances to resources based on their previous execution history; and *Organisational Allocation*, i.e., the ability to offer or allocate activity instances to resources based their organisational position and their relationship with other resources.

Patterns *Authorisation* and *Automatic Execution* are not on the list. The former is not included because it is unrelated to the definition of conditions for resource selection and the latter because it is unrelated to the assignment language and is inherently supported by all Business Process Management Systems (BPMSs). RAL provides support for eight of them, as shown with the examples in Table 6. Only History-Based Allocation is not covered at the moment.

8.2. Analysis

A framework for the analysis of the resource perspective in BPs called Collection of Resource-centrIc Supporting Tools And Languages (CRISTAL) [4], available at `http://www.isa.us.es/cristal`, has been developed. CRISTAL serves two main purposes: i) to show the feasibility[10] of implementing the analysis operations described in Section 7; ii) to pave the way for a successful API that can be integrated into a broad variety of tools, from process modellers to process engines through process monitoring consoles, and that can be extended to provide further management capabilities for the resource perspective in BPs.

Next, we detail how these two purposes have been achieved and we conclude with some performance considerations.

8.2.1. Implementation of the Analysis Operations

We have developed the support necessary for the automated execution of all of the person-activity operations, using the procedures described in the previous sections, in a component of CRISTAL called RAL Analyser.

The first step that needs to be performed to implement the analysis operations as described in Section 7 is to create the DL-based KBs ($\mathcal{K}_C$, $\mathcal{K}_C^1$). This implementation has been performed with OWL ontologies [27] because most DL reasoners are designed to use OWL ontologies as input. OWL is a knowledge representation scheme designed specifically for use on the Semantic Web that exploits existing Web standards (XML and RDF) and the formal rigor of DLs. The following OWL ontologies are created:

- Two ontologies obtained after mapping the organisational metamodel and the BP metamodel used in RAL as detailed in Section 6.1 and 6.2, respectively.
- Two ontologies obtained after mapping the organisational model as detailed in Section 6.1 and the BP model as detailed by Algorithm 1.
- One ontology obtained after mapping the resource assignments with RAL following Algorithm 2.

[10]We refer to feasibility from a theoretical point of view, i.e., whether something is doable.

Pattern	**Assignment**	**RAL Expression**	**RAL Mod.**
Direct Allocation	*Anna* is responsible for checking the response received from the *Research Vice-chancellorship*	`IS Anna`	RAL Core
Role-Based Allocation	A *Researcher* of project *HRMS* is in charge of submitting the paper to the conference	`HAS ROLE Researcher IN UNIT HRMS`	RAL Org
Deferred Allocation	Instances of the *Send Travel Authorisation* activity must be performed by the person referenced in the field *Attendee* of the data object *Travel Authorisation*	`IS PERSON IN DATA FIELD TravelAuthorisation.Attendee`	RAL Data / DataOrg
Separation of Duties (SoD)	The travel authorisation form cannot be signed by the person who filled in the document	`(NOT (IS ANY PERSON responsible for ACTIVITY FillTravelAuthorisation))`	RAL AC
Case Handling	A single person with role *Researcher* is responsible for performing all the activities of the process	`(HAS ROLE Researcher) AND (IS ANY PERSON responsible for ACTIVITY SubmitCRV)*`	RAL AC
Retain Familiar	The person that submits the paper is due to register at the conference	`IS ANY PERSON responsible for ACTIVITY SubmitCRV`	RAL AC
Capability-Based Allocation	Instances of the *Sign Travel Authorisation* activity must be allocated to someone holding a degree	`HAS CAPABILITY Degree`	RAL Org
History-Based Allocation	-	-	-
Organisational-Based Allocation	The authorisation form must be filled in by someone who occupies position HRMS PhD Student or by someone who directly reports work to the project coordinator	`(HAS POSITION PhDStudent) OR (DIRECTLY REPORTS TO POSITION ProjectCoordinator)`	RAL Org

Table 6: Specification of the Creation Patterns with RAL
(*) The assignment is the same for all the activities of the process; however, the second part of the composition is not necessary for the first activity, so either it is omitted or it has to be ignored during resource allocation

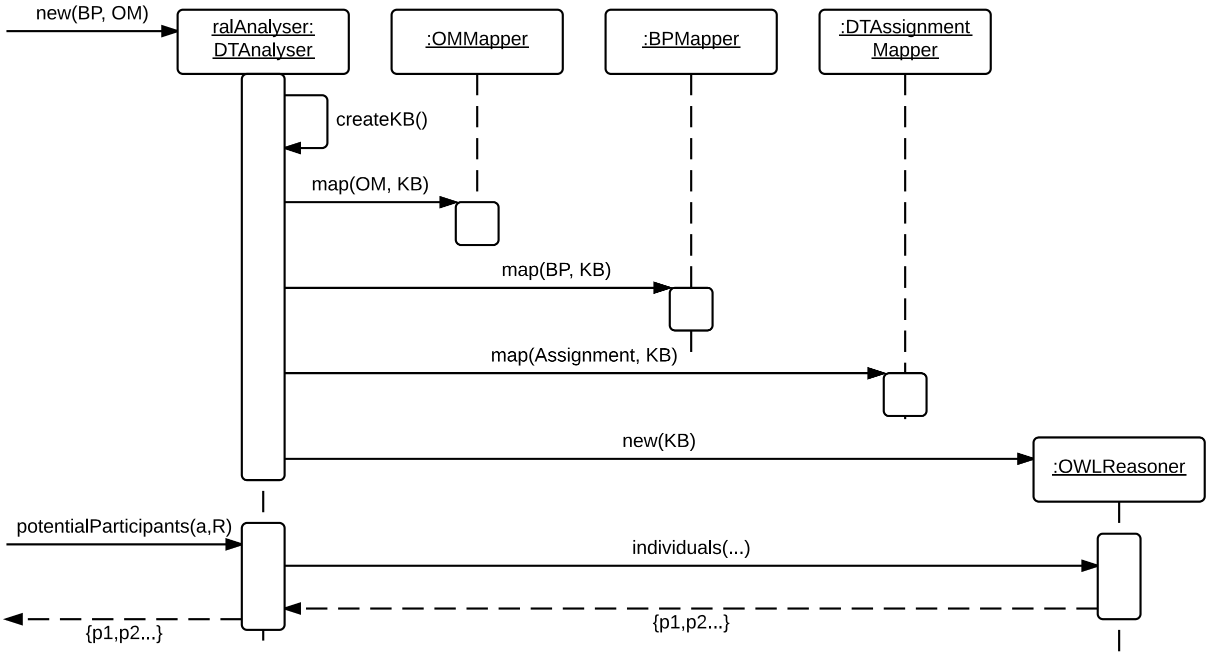


Figure 7: Sequence diagram of an analysis operation as implemented by RAL Analyser

- Two ontologies for $\mathcal{K}_C$ and $\mathcal{K}_C^1$ obtained by importing the aforementioned ontologies and adding the axioms that are specific for each KB.

The first two ontologies have been manually defined in OWL because they do not change with new organisational models or BP models. The other ontologies are automatically generated by RAL Analyser using the Java OWL API[11].

Figure 7 depicts a sequence diagram that illustrates all these steps for resolving a design-time analysis operation. First, a design-time RAL Analyser is instantiated with its context, and it creates a new KB and uses the different mappers (`OMMapper`, `BPMapper`, and `DTAssignment Mapper`) to map the context into it. It also creates an `OWLReasoner` that will be used during the execution of analysis operations. When an analysis operation is invoked, the analyser transforms it in terms of DL standard reasoning operations, as detailed in Section 7, and uses the `OWLReasoner` to solve them. In the current version, RAL Analyser uses *HermiT* [33]. Other DL reasoners that implement the OWL API reasoner interface can be seamlessly used instead.

8.2.2. API for Resource Analysis in Business Processes

CRISTAL [4] provides a common interface for the resource analysis operations and a pluggable framework into which many different implementations of them can be integrated. In fact, apart from RAL Analyser, CRISTAL includes another implementation of the resource

[11] `http://owlapi.sourceforge.net/`

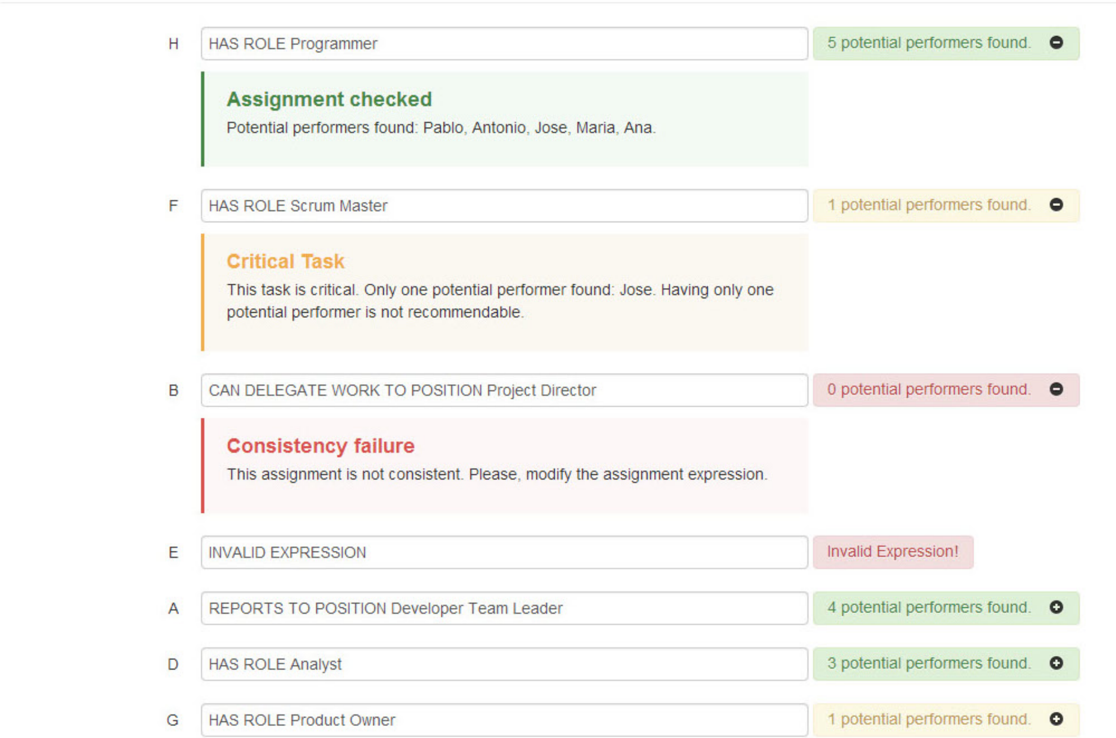


Figure 8: RAL analyser operations integrated into PRspectives

analysis operations called RAL-neo4j, which is based on the graph database Neo4J[12]. The approach followed in this implementation is very similar to the one used in RAL Analyser: the models are mapped into the database and RAL expressions are mapped as database queries. However, there is no support for RAL AC constraints or for the considerations regarding RAL Data and DataOrg detailed in Section 7 because they require reasoning about future activity instances that may occur, and Neo4J does not provide the reasoning capabilities of DLs.

CRISTAL also implements a REST API for the analysis operations. This enables their integration with other Web applications. Using this feature, RAL Analyser has been integrated with PRspectives[13], a BP modeller with support for multiple perspectives, including the resource perspective. Specifically, PRspectives uses the REST API to invoke the design-time analysis operations to guide the user while defining resource assignments for a BP. Figure 8 illustrates how it shows information about the potential participants, the critical activities and the consistency of the assignments.

8.3. Performance Considerations

As previously mentioned, our goal is not to provide the most efficient implementation of every operation but (1) a definition of several novel analysis operations for the BP resource

[12] `www.neo4j.org`
[13] `www.isa.us.es/prspectives`

perspective, (2) a formalisation of all these operations, and (3) a reference implementation that can be used as a guide for the development of more efficient implementations for some of the operations. Therefore, it is not our purpose to provide a thorough performance evaluation of the implementation. However, we do provide some figures to give an idea of how this reference implementation performs.

8.3.1. Execution Environment

The experiments were performed in a MacBook Pro featuring a 2,66 GHz Intel Core 2 Duo processor and 8GB 1067 MHz DDR3. The tests were run using Java 1.7 and the HermiT OWL reasoner 1.3.8. In order to reduce significance of possible outliers produced by occasional interferences with the operating system or the network, averaged times in 15 runs were registered and the maximum and minimum timings for each experiment were discarded.

The goal of this performance evaluation is to analyse the performance the reasoner would have while changing resource assignments, but not while changing the structure of the organisational model. This means that the tests include the time it takes to load the resource assignments in the reasoner, but they do not include the time it takes to load the base ontologies and the organisational model.

8.3.2. Significant Factors

Both from a theoretical and a practical point of view, the analysis to determine the tendency of the performance of a DL reasoner is a difficult task because it may depend on a variety of factors. In our experiments, we have considered the following ones:

1. The size of the organisational model (O). Intuitively, the bigger the organisational model (i.e., more positions, more people, more roles, more units), the more complex the reasoning, and hence, the more time the analysis operations should take.
2. The size of the process model in terms of the number of activities (A). Intuitively, the more activities, the more concepts should be added to the KB, and hence, the more time the analysis operations should take.
3. The type of RAL expressions used in the resource assignments. Intuitively, simple expressions such as `HAS ROLE r` would be faster to solve than composite expressions such as `(HAS ROLE r) OR (HAS POSITION p)`. Furthermore, the inclusion of RAL AC expressions is expected to introduce additional complexity due to the additional dependencies they add to the potential participants of an activity as discussed in Section 5.

The first factor has been taken into account by analysing the performance using randomly generated organisational models of different sizes. In all of them, the same proportion of people, roles and positions is kept. The second factor has been taken into account by analysing processes of different sizes. Finally, the third factor has been taken into account by analysing the performance of different resource assignments. In particular, three categories of RAL expressions have been established: simple, composite and AC, which correspond with the three types of RAL expressions discussed above; and two sizes of BP models have been considered, namely BP models with 5 and 20 activities. These numbers have been chosen based on experiments in the understandability of BPs that suggest that a BP model should not have more than 20 activities [34]. The details about how the organisational models are generated and the concrete resource assignment expressions used in the tests are available at `https://github.com/isa-group/cristal/tree/master/ral-performance-tester`.

8.3.3. Results

Figure 9 depicts the results of the performance evaluation for three person-activity operations, one for each category of person-activity operations, namely consistency checking, critical participants, and potential participants. Note that the first two operations are applied to the whole process but the potential participants must be applied to a particular activity. Therefore, the numbers for the potential participants are the average of the performance evaluation of the potential participants for each activity of the process.

The following observations can be made from these results:

- Operation consistency checking performs much better than the other two operations. Specifically, it takes between 4 and 6.5 seconds to analyse the consistency of an organisational model of 450 people, whereas it takes the same time to execute a potential participants or critical participants operation for an organisational model of 60 people. The reason for this behaviour is that reasoners are usually more efficient when checking if the ABox is consistent than when retrieving all individuals of a concept of the ontology. As a matter of fact, many individual retrieval operations require first a consistency checking of the KB.

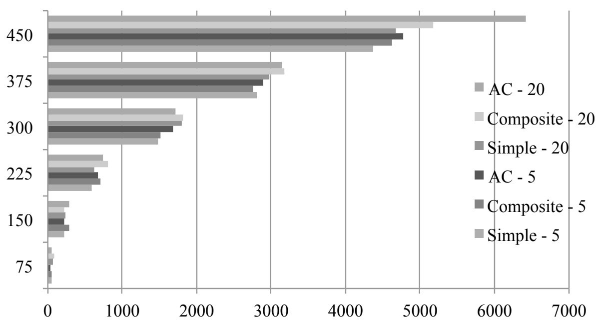
- The factor that has the greatest influence is the size of the organisational model. Moreover, the performance of RAL Analyser seems to exhibit an exponential behaviour with respect to the size of the organisational model.

- The outlier in the operation potential participants for AC models with 5 activities and more than 60 people in the organisational model is caused because the computation of the potential participants of two out of the five activities of the process take much longer than the other three. This makes the average higher than, for instance, in the case of AC models with 20 activities in which only 2 out of 20 take much longer than the other ones.

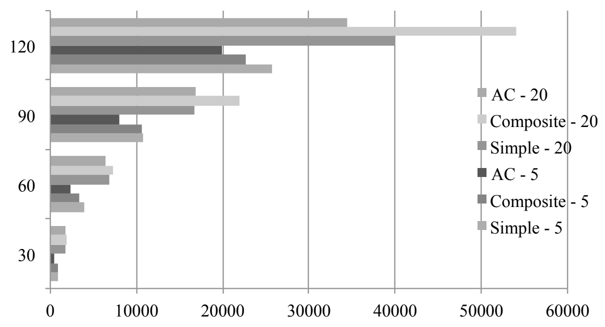
8.3.4. Threats to validity

The internal validity refers to whether there is sufficient evidence to support the conclusions and the sources of bias that could compromise those conclusions. In order to minimise the impact of external factors in our results, each analysis operation was executed 15 times for each experiment to get average values. Regarding the random generation of organisational models, we avoided the risk of creating incorrect models by introducing a validity check of the model before executing the analysis operation.

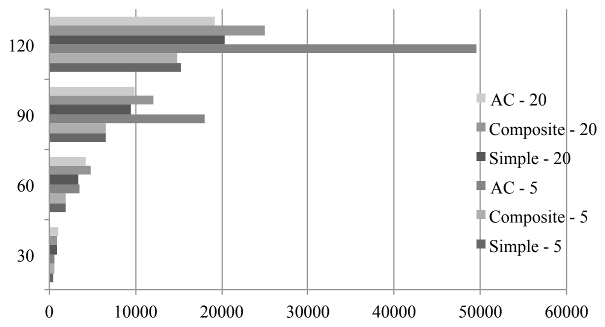
The external validity is concerned with how the experiments capture the objectives of the research and the extent to which the conclusions drawn can be generalised. As mentioned



(a) Consistency checking



(b) Critical participants



(c) Potential participants

Figure 9: Performance evaluation of RAL Analyser. The x-axis represents time in milliseconds. The y-axis represents the size of the organisational model in terms of number of people. The names of the categories identify the type of resource assignment (simple, composite and AC) and the size of the BP model (5 and 20).

before, the goal of this performance evaluation is to analyse the performance the reasoner would have while changing resource assignments. Therefore, if the analysis operations are used in another context (e.g. evolutions in the organisational model), the conclusions obtained here may not be representative. Another threat to validity is how the results obtained can be extrapolated to the performance of a person-activity analyser in a real setting. To this end, it would be convenient to compare the structure of the organisational models used in the experiment with the structure of real organisations to better extrapolate the results obtained here to a real setting because the structure of the organisation could also have influence over the performance results. The same thing applies to the type of RAL expressions used in the resource assignments.

8.3.5. Discussion

From the results obtained in the performance analysis, we can conclude that the consistency checking operation performs reasonably well with organisational models of medium size. However, there is still much room for improvement concerning the performance results for critical participants and the potential performers, especially for the latter. Next, we detail several directions in which one can look for improving the performance of the RAL

Analyser:

- Using hybrid analysers. This optimisation is based on the fact that if an activity is not involved in a RAL AC expression, then all the operations can be applied to the activity in isolation without considering the rest of the BP model. Therefore, all those activities could be sent to an implementation without reasoning capabilities such as RAL-neo4j, while the others could be sent to the DL-based implementation. This could improve the performance, especially if processes do not have many RAL AC expressions.

- Transforming concept retrieval into consistency checking problems in the DL reasoner. This optimisation is based on the fact that DL reasoners are usually more efficient when checking if the ABox is consistent than when retrieving all individuals of a concept of the ontology. Therefore, non-reasoning implementations can be used to obtain a set of possible potential participants for a RAL AC expression following an approximation such as the one defined in [5] and, then, checking in the DL reasoner which of them are actually potential participants. If the number of possible potential participants is low, the performance could be improved significantly.

- Using filters to reduce the size of the KB before the analysis is executed. This optimisation is based on proposals that have faced similar issues in the matchmaking of semantic Web services [35]. The idea is to use a filter that removes from the KB all the elements that are not used in the RAL expressions involving RAL AC constraints. For instance, if activity A has the assignment `HAS POSITION pos1` and activity B has the assignment `(IS ANY PERSON responsible for ACTIVITY A) AND (HAS ROLE r1)`, the filter would remove all positions other than $pos1$, all roles other than $r1$, all activities other than A and B, and all people who have neither position $pos1$ nor role $r1$. This reduces significantly the size of the KB and, thus, it makes the reasoning much more efficient.

9. Related Work

The BP resource perspective is increasingly catching the attention of the BPM community. There are many proposals dealing with resource assignment in BPs, e.g. [6, 10, 31, 36, 37]. However, despite the need of considering resources together with the other BP perspectives (e.g. data and control flow) for consistency checking and data access control purposes has been described [38], the automated analysis of the BP resource perspective has not received much attention so far, and only two operations have been addressed.

Bertino et al. have developed a *constraint analysis and enforcement module*, consisting of a set of algorithms for consistency checking and resource allocation planning. Based on Logic Programming, the approach checks the design-time consistency of a BP model with regard to its resource assignments; however, the considerations related to BP control flow are disregarded. As a consequence, the analysis operations may not be accurate with processes that contain loops and access-control constraints, as explained in Section 3.

The Business Activities introduced by Strembeck and Mendling [10] as a way to model Role-Based Access Control (RBAC) in organisations and to define all kinds of access-control constraints between process activities rely on Petri Nets to check the consistency of the process. The authors addressed consistency checking at design time and at run time, by developing ad-hoc algorithms. As a consequence of that work, subsequent work aimed at developing algorithms for the identification of several potential conflicts related to resource assignment in Business Activities, was performed by Schefer et al. [39]. Detection algorithms were developed regarding design-time constraint definition, design-time assignment relations, and runtime task allocation.

The Workflow on Intelligent Distributed database Environment (WIDE) introduced by Casati et al. [40], allows both automatic and manual allocation of tasks to resources. In automatic allocation, the local scheduler module is responsible for dispatching requests for allocation of tasks to resources, and it uses different criteria for resource selection, e.g. workload, availability of resources, and priorities. The only analysis operation mentioned in WIDE specification is referred to the calculation of the *Potential Participants* of the BP activities, which is done at run time.

Yet Another Workflow Language (YAWL) 2.0 [32] is the current version of an advanced WF modelling language that nowadays covers the BP control flow, data and resource perspectives. It is equipped with a run-time engine that deals with resource allocation, in such a way that the resource assignments are automatically resolved during BP execution. Thus, the *Potential Participants* of the process activities are automatically calculated at run time, but there is no support for the analysis of the BP at design time.

Similarly to YAWL, Architecture of Integrated Information Systems (ARIS) [41], a commercial tool suite that provides support for the management of several BP perspectives, addresses the automatic resolution of resource assignments at run time. To the best of our knowledge, design time analysis is outside the scope of ARIS, and no more resource-related analysis operations are supported.

Du et al. have developed a resource management system [42] whose resource engine is capable of automatically resolving the resource expressions associated to the process activities at the enactment phase of the BP lifecycle, i.e., at run time. Nothing is said about the technique utilised to perform the analysis or about considering including in the implementation support for more advanced resource analysis.

The Constrained WF System designed by Tan et al. [43] is focused on checking for consistency related to the resource expressions configured in a process as a set of constraints, with the aim of helping the BP designers to define a sound constrained BP authorisation schema. They define consistency rules for constraint-task pairs that guarantee that there is no inconsistency, ambiguities and redundancy contained in the set of constraints. The authors argue that by guaranteeing the non-existence of these problems, for each resource authorised in a task in the process there is always at least one successful BP instance that satisfies all the constraints. We assume that the operation for calculating the *Potential Participants* of the activities is supported by the system. Nothing about the possible existence of exclusive gateways or complex process structures (i.e., loops) is mentioned, so control-flow issues might not be considered. The approach is targeted at design time analysis.

Table 7 collects the result of our study of the state of the art regarding the design-time support for the person-activity operations, which are identified with the acronyms defined

Approaches	NP	PP	NPA	PA	CC	CP	CA	Creation Patterns
Bertino et al. [15]		✓			✓			5
Schefer et al. [10]		✓			✓			5
WIDE [40]								7
YAWL [44, 32]								6
ARIS [41]								7
Du et al. [42]								4
Tan et al. [43]		✓			✓			4
RAL	✓	✓	✓	✓	✓	✓	✓	8

Table 7: Current support for the person-activity operations at design time

in Section 3. In the cells: ✓indicates that automated support is provided; and a blank indicates either that the analysis operation is not supported, or that the information for that operation could not be extracted from the description of the proposal. Nevertheless, we argue that for the approaches supporting the Potential Participants operation, support for the other basic person-activity operations (cf. Section 3) could be developed by extending the approach at a "not very high cost" (regarding time and effort). In addition, the last column of the table shows the number of creation patterns fully supported by the assignment language used for resource selection by the approaches, among the nine patterns defined in Section 8.1. This is important, since the use of expressive languages introduces complexity in the automation of the operations, as is the case of RAL Data, RAL DataOrg and RAL AC due to the run-time constraints.

As shown in the table, the operation supported by more approaches is *Potential Participants*, specifically supported by the approaches described in [10, 15, 43]. This is not very significant, since it is the most basic operation for an organisation that uses resource-aware BP models and is interested in automating resource allocation. The same three approaches also address design-time *Consistency Checking* by means of ad-hoc algorithms. We find it reasonable, since at least before launching a process we should make sure that it does not contain inconsistencies related to resource assignment and, hence, there will always be somebody to which every activity can be allocated during the execution of the process. Furthermore, Bertino et al. [15], and Strembeck and Mendling [10] consider both static and dynamic access-control constraints. However, these approaches rely on the RBAC model for resource assignment, so the languages used for resource selection are less expressive than RAL in terms of WRPs. In addition, they implement a relaxed notion of consistency checking where the control flow of the process is not taken into consideration. Besides, the task duties are neither considered in the resource assignments of current approaches.

Therefore, the RAL-based approach presented in this paper is more expressive than most of the approaches for resource assignment, and provides further capabilities for automatic resource analysis, since RAL supports eight out of the nine creation patterns defined in Section 8.1, and we provide design-time support for the seven analysis operations identified using it as resource assignment language.

10. Conclusions and Future Work

We have addressed gaps related to resource specification and analysis in BPs. Specifically, we demonstrated how RAL can be used to define expressive resource selection conditions and how its DL-based semantics can be extended to extract useful, valuable information in an automated way. In particular, we have defined a catalogue of seven person-activity operations related to how resources are involved in BP activities, for which we have developed design-time support. Due to the expressive power of RAL, other BP perspectives need to be taken into account, namely, the data perspective for the assignments that required information provided in data fields and the control flow perspective for access-control constraints defined between activities.

The main conclusion drawn from this paper is that for the category of processes called R3C-processes, it is unnecessary to model the full semantics of the control flow to implement person-activity analysis operations, and they can be implemented solely using DL reasoners. Giving support to the whole catalogue solely with DLs makes it easier and quicker to build a reference implementation of the whole catalogue such as the one we have developed and integrated as part of CRISTAL[14]. This implementation can be used as a baseline and guide for developing alternative and perhaps more efficient implementations of the catalogue. In this sense, the proof-of-concept implementation has also revealed that there is still much room for improvement concerning the performance of some of the person-activity operations. We have already identified some potential ways to address this issue in the future, as detailed in Section 8. Finally, we plan to develop run-time support for the catalogue presented in this paper and to extend the work to support teamwork.

Acknowledgements

We thank the reviewers for their thorough revision of the paper and their constructive feedback and Dr. José Antonio Parejo for sharing with us his expertise on performance measurement.

References

[1] E. Scherer, M. Zölch, Design of activities in shop floor management: A holistic approach to organisation at operational business levels in BPR projects, in: J. Browne, D. O'Sullivan (Eds.), Re-engineering the Enterprise, IFIP, Springer US, 1995, pp. 261–272.

[2] G. Decker, Design and Analysis of Process Choreographies, Ph.D. thesis, University of Potsdam (2009).

[3] C. Cabanillas, M. Resinas, A. Ruiz-Cortés, RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes, in: BPM Workshops (BPD'11), 2011, pp. 50–61.

[4] C. Cabanillas, A. del Río-Ortega, M. Resinas, A. Ruiz-Cortés, CRISTAL: Collection of Resource-centrIc Supporting Tools And Languages, in: BPM 2012 Demos, Vol. 940, 2012, pp. 51–56.

[5] C. Cabanillas, M. Resinas, A. Ruiz-Cortés, Defining and Analysing Resource Assignments in Business Processes with RAL, in: ICSOC, Vol. 7084, 2011, pp. 477–486.

[6] WS-BPEL Extension for People (BPEL4People), Tech. rep., OASIS (2009).

[14] www.isa.us.es/cristal

[7] Website, The RASCI matrix, `http://www.ha-ring.nl/en/doc_en/rasci-matrix` (Last accessed in January 2014).
[8] M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer Verlag, 2012.
[9] OMG, BPMN 2.0, Recommendation, OMG (2011).
[10] M. Strembeck, J. Mendling, Modeling process-related RBAC models with extended UML activity models, Inf. Softw. Technol. 53 (2011) 456–483.
[11] A. del Río-Ortega, M. Resinas, C. Cabanillas, A. R. Cortés, On the definition and design-time analysis of process performance indicators, Inf. Syst. 38 (4) (2013) 470–490.
[12] P. Trinidad, A. Ruiz-Cortés, Abductive Reasoning and Automated Analysis of Feature Models: How are they connected?, in: VaMoS, 2009, pp. 145–153.
[13] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, 2013.
[14] H. Enderton, Elements of Set Theory, Acad. Press, 1977.
[15] E. Bertino, E. Ferrari, V. Atluri, The specification and enforcement of authorization constraints in workflow management systems, ACM Trans. Inf. Syst. Secur. 2 (1999) 65–104.
[16] T. W. Malone, Modeling Coordination in Organizations and Markets, Management Science 33 (10) (1987) 1317–1332. doi:10.1287/mnsc.33.10.1317.
URL `http://0-pubsonline.informs.org.fama.us.es/doi/abs/10.1287/mnsc.33.10.1317`
[17] C. Cabanillas, M. Resinas, A. Ruiz-Cortés, Designing Business Processes with History-Aware Resource Assignments, in: BPM 2012 Workshops (BPD'12), Vol. 132, 2012, pp. 101–112.
[18] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, D. Edmond, Workflow Resource Patterns: Identification, Representation and Tool Support, in: CAiSE, 2005, pp. 216–232.
[19] C. Cabanillas, M. Resinas, A. Ruiz-Cortés, J. Mendling, Methodology to Extend RAL, in: Jornadas de Ciencia e Ingeniería del Software, 2014.
[20] N. Russell, A. ter Hofstede, D. Edmond, W. van der Aalst, Workflow Resource Patterns, Tech. rep., BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven (2004).
[21] W. M. P. van der Aalst, The Application of Petri Nets to Workflow Management, Journal of Circuits, Systems, and Computers 8 (1) (1998) 21–66.
[22] J. Desel, J. Esparza, Free choice Petri nets, Cambridge University Press, New York, NY, USA, 1995.
[23] M. Weidlich, J. Mendling, M. Weske, Efficient Consistency Measurement Based on Behavioral Profiles of Process Models, IEEE Trans. Software Eng. 37 (3) (2011) 410–429.
[24] A. H. M. T. Hofstede, H. Proper, How to Formalize It? Formalization Principles for Information System Development Methods, Information and Software Technology 40 (1998) 519–540.
[25] B. Motik, R. Rosati, Reconciling description logics and rules, J. ACM 57 (2008) 30:1–30:62.
[26] M. Bhatt, W. Rahayu, S. P. Soni, Carlo, Ontology driven semantic profiling and retrieval in medical information systems, Web Semantics: Science, Services and Agents on the World Wide Web 7 (4) (2009) 317–331.
[27] B. Motik, P. F. Patel-Schneider, B. C. Grau, OWL 2 Web Ontology Language Direct Semantics, `http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/` (2009).
[28] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider, The Description Logics Handbook: Theory, Implementations, and Applications, Cambridge University Press, 2003.
[29] D. Calvanese, G. De Giacomo, M. Lenzerini, Keys for free in description logics, in: Proc. of the 13th Int. Workshop on Description Logics (DL 2000), Vol. 33 of CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, 2000, pp. 79–88.
[30] W. M. P. van der Aalst, A. Kumar, A reference model for team-enabled workflow management systems, Data Knowl. Eng. 38 (3) (2001) 335–363.
[31] A. Awad, A. Grosskopf, A. Meyer, M. Weske, Enabling Resource Assignment Constraints in BPMN, Tech. rep., BPT (2009).
[32] M. Adams, YAWL v2.3-User Manual (2012).
[33] B. Motik, R. Shearer, I. Horrocks, Hypertableau Reasoning for Description Logics, Journal of Artificial Intelligence Research 36 (2009) 165–228.

[34] J. Mendling, L. Sánchez-González, F. García, M. La Rosa, Thresholds for error probability measures of business process models, Journal of Systems and Software 85 (5) (2012) 1188–1197. doi:10.1016/j.jss.2012.01.017.
URL http://www.sciencedirect.com/science/article/pii/S0164121212000040

[35] J. M. García, D. Ruiz, A. Ruiz-Cortés, Improving semantic web services discovery using SPARQL-based repository filtering, J. Web Sem. 17 (2012) 12–24.

[36] Web Services-Human Task (WS-HumanTask) v1.1, Tech. rep., OASIS (2010).

[37] M. Adams, The Resource Service, in: Modern Business Process Automation, 2010, pp. 261–290.

[38] V. Künzle, M. Reichert, Integrating Users in Object-Aware Process Management Systems: Issues and Challenges, in: BPM Workshops, 2010, pp. 29–41.

[39] S. Schefer, M. Strembeck, J. Mendling, A. Baumgrass, Detecting and Resolving Conflicts of Mutual-Exclusion and Binding Constraints in a Business Process Context, in: OTM Conferences (CoopIS'12), 2011, pp. 329–346.

[40] F. Casati, P. Grefen, B. Pernici, G. Pozzi, G. Sanchez, WIDE workflow model and architecture (1996).

[41] A.-W. Scheer, ARIS-Business Process Modeling, 3rd Edition, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[42] W. Du, J. Davis, Y.-N. Huang, M.-C. Shan, Enterprise Workflow Resource Management, in: RIDE, 1999, pp. 108–115.

[43] K. Tan, J. Crampton, C. A. Gunter, The Consistency of Task-Based Authorization Constraints in Workflow Systems, in: IEEE workshop on Computer Security Foundations, 2004, pp. 155–169.

[44] W. M. P. van der Aalst, A. H. M. ter Hofstede, YAWL: Yet Another Workflow Language, Inf. Syst. 30 (4) (2005) 245–275.

Appendix A. RAL EBNF Specification

```
RALExpression := ANYONE
          | PersonExpr                       | HierarchyExpr
          | GroupResourceExpr                | DenyExpr
          | CommonalityExpr                  | CompoundExpr
          | CapabilityExpr

PersonExpr := IS PersonConstraint

GroupResourceExpr := HAS (PositionConstraint | UnitConstraint)
                  | HAS RoleConstraint [IN UnitConstraint]

CommonalityExpr := SHARES Amount (POSITION | UNIT) WITH PersonConstraint
                  | SHARES Amount ROLE [IN UnitConstraint] WITH PersonConstraint

CapabilityExpr := HAS CAPABILITY CapabilityConstraint

HierarchyExpr := ReportExpr | DelegateExpr

ReportExpr := Depth REPORTS TO PositionRef | IS Depth REPORTED BY PositionRef

DelegateExpr := CAN DELEGATE WORK TO PositionRef | CAN HAVE WORK DELEGATED BY PositionRef

DenyExpr := NOT '(' DeniableExpr ')'

CompoundExpr := '(' Expr ')' OR '(' Expr ')' | '(' Expr ')' AND '(' Expr ')'

DeniableExpr := PersonExpr | GroupResourceExpr | CommonalityExpr | CapabilityExpr

PersonConstraint := personName
                  | PERSON IN DATA FIELD dataObject.fieldID
                  | ANY PERSON TaskDuty ACTIVITY activityID
```

```
PositionConstraint := POSITION (positionName | IN DATA FIELD dataObject.fieldID)

RoleConstraint := ROLE (roleName | IN DATA FIELD dataObject.fieldID)

UnitConstraint := UNIT (unitName | IN DATA FIELD dataObject.fieldID)

CapabilityConstraint := capabilityID | CapabilityRestriction

PositionRef := POSITION OF PersonConstraint | PositionConstraint

Amount := SOME | ALL          Depth := DIRECTLY | λ
```

Appendix B. Proofs

This appendix includes the proofs for Theorems 1 and 2 of Section 5. In order to do that, we first define the following abbreviations:

- $X_a^\sigma = \{ai \in \sigma | \pi_a(ai) \neq a\}$ is the set of activity instances that belong to the trace in a complete process execution σ whose activity is different than a.
- $R_a^\sigma = \{p \in P | \exists ai \in \sigma(\pi_a(ai) = a \wedge \pi_p(ai) = p)\}$ is the people that have been allocated to activity a in the process execution σ.

Furthermore, several equivalences between pairs of process executions can be defined attending to the different perspectives of the business process, namely: control flow, resources and data.

Definition 18 (Process execution equivalences)**.** *Let $\sigma_1 = (\tau_1, \delta_1)$ and $\sigma_2 = (\tau_2, \delta_2)$ be two process executions of a business process with A activities whose traces have n and m activity instances respectively:*

- *σ_1 is activity-equivalent to σ_2, denoted by $\sigma_1 \equiv^A \sigma_2$, if they contain exactly the same sequence of executed activities:*

$$\sigma_1 \equiv^A \sigma_2 \Leftrightarrow n = m \wedge \pi_a(\sigma_1(i)) = \pi_a(\sigma_2(i)) \textit{ for all } 0 \leq i \leq n-1$$

- *σ_1 is resource-equivalent to σ_2, denoted by $\sigma_1 \equiv^R \sigma_2$, if the same activities have been performed by the same people in both process executions no matter the order in which activities have been performed nor the number of times an activity has been performed provided that it has been performed by the same people:*

$$\sigma_1 \equiv^R \sigma_2 \Leftrightarrow \forall a \in A(R_a^{\sigma_1} = R_a^{\sigma_2})$$

- *σ_1 is data-equivalent to σ_2, denoted by $\sigma_1 \equiv^D \sigma_2$, if they have the same assignment of values to their data objects:*

$$\sigma_1 \equiv^D \sigma_2 \Leftrightarrow \delta_1 = \delta_2$$

Moreover, we write $\sigma_1 \underset{d_1,\dots,d_n}{\equiv^D} \sigma_2$ to denote that $\delta_1(d_i) = \delta_2(d_i)$ for all $d_i \in D$ with $1 \leq i \leq n$.

We now introduce two lemmas which are used in the proof of Theorems 1 and 2. The first lemma formalises the intuition that the order in which activities are performed and the number of times an activity is performed are irrelevant with respect to the people that meet a resource selection condition provided that they are performed by the same set of people.

Lemma 2. *For any σ_1, σ_2 process executions of a business process, it holds that if $\sigma_1 \equiv^R \sigma_2$ and $\sigma_1 \equiv^D \sigma_2$ then $\rho^{\sigma_1} = \rho^{\sigma_2}$*

Proof. To prove it, we assume that there exist two σ_1 and σ_2 such that $\sigma_1 \equiv^R \sigma_2$ and $\sigma_1 \equiv^D \sigma_2$ and $\rho^{\sigma_1} \neq \rho^{\sigma_2}$. In that case, since the organisational model O is the same, the data state is exactly the same and the resource selection conditions are the same as well, the only reason why the people that meet the resource selection conditions may be different is that there exists at least one activity a such that the people that meet its resource selection conditions are defined using some RAL AC constraints that causes that $\rho^{\sigma_1}(a) \neq \rho^{\sigma_2}(a)$. Since all RAL AC constraints refer to people that have performed an activity, this means that the difference between σ_1 and σ_2 must be that there is at least one person that has performed an activity in σ_1 and it has not performed the same activity in σ_2. However, this contradicts the fact that $\sigma_1 \equiv^R \sigma_2$. □

The second lemma formalises the intuition that the people that meet the resource selection condition of an activity are not influenced by the executions of the activities that belong to a different AC-group.

Lemma 3. *Let A be the activities of a business process bp, let* AC-groups $= \{ac_1, \ldots, ac_n\}$ *be the* AC-groups *of bp and let $x, y \in A$ be two activities such that $x \in ac_i$, $y \in ac_j$ and $i \neq j$. For any process executions σ_1, σ_2 of bp such that $\sigma_1 \underset{D_{ac_i}}{\equiv^D} \sigma_2$ it holds that $X_y^{\sigma_1} = X_y^{\sigma_2} \Rightarrow \rho^{\sigma_1}(x) = \rho^{\sigma_2}(x)$.*

Proof. In order to verify this lemma, we consider the following two situations:

- x is the only activity in its AC-group ac_i. This means that x is not AC-related with any other activity, i.e., there is not an $a \in A$ such that $x \sim a$. If this is the case, the people that meet the resource selection condition of x do not change when the BP trace changes. Moreover, since $\sigma_1 \underset{D_{ac_i}}{\equiv^D} \sigma_2$, there is no change in the data fields used by x either. Therefore, we conclude that $\rho^{\sigma_1}(x) = \rho^{\sigma_2}(x)$.

- x is with at least another activity in its AC-group ac_i. Since $y \notin ac_i$, it means that $x \nsim y$ and that there is not any set of activities $\{a_i, \ldots, a_j\}$ with $1 \leq i, j \leq n$ such that $x \sim a_i, \ldots, a_j \sim y$[15]. This means that x is neither directly nor indirectly AC-related with y and, hence, the people that meet the resource selection condition of x do not change regardless of the number of executions and allocations made in y. Furthermore, since $\sigma_1 \underset{D_{ac_i}}{\equiv^D} \sigma_2$, there is no change in the data fields used by any activity in ac_i either, thus making $\rho^{\sigma_1}(x) = \rho^{\sigma_2}(x)$.

[15] Otherwise, all $\{a_i, \ldots, a_j\}$ would belong to ac_i by definition of AC-group and, hence, y would also belong to ac_i, which contradicts $y \notin ac_i$.

□

Finally, we recall Theorems 1 and 2 and prove them.

Theorem 1. *Let O be an organisational model with P persons. For any* R3C-process *bp with A activities whose resource assignment is consistent, it holds that for any $a \in A$, $\forall \sigma \in T_a(\exists S \in \mathcal{S}(\rho^\sigma(a) = \rho^S(a))$ and $\forall S \in \mathcal{S}(\exists \sigma \in T_a(\rho^S(a) = \rho^\sigma(a))))$.*

Proof. 1. Let $\sigma \in T_a$ be an execution of the BP and let $A_{>0} = \{a \in A \mid \#_a^\sigma > 0\}$. To prove the first part we have to find an $S \in \mathcal{S}$ such that $\rho^\sigma(a) = \rho^S(a)$. If for all $A = A_{>0}$, then $S = \sigma$. Otherwise, we have to build S such that it includes all of the $ai \in \sigma$ and its data state for $D_{A_{>0}}$ is the same as in σ plus at least one $(x, p_x) \in AI$ for each $x \in A \setminus A_{>0}$ and values for all data fields $D \setminus D_{A_{>0}}$. Furthermore, the addition of these activity instances and values of data fields should be done in a way such that $\rho^S(a)$ does not change and the resulting S must be $R - valid$.

The former requirement is not an issue since the BP is an R3C-process and we have that $\#_a^\sigma > 0$, which means that $\#_y^\sigma > 0$ for all y that belong to the AC-group of a and for all z such that $D_z \cap D_y \neq \emptyset$. This means that only activity instances from other AC-groups that depend on different data fields must be added and, according to Lemma 3 we have that the people that meet the resource selection condition of an activity are not influenced by the executions of the activities that belong to a different AC-group and depend on different data fields.

As for the latter requirement, since the BP is an R3C-process, we know that either all activities of an AC-group are in σ or none of them are. Moreover, the BP has no dead activities and its resource assignment is consistent. This means that for each AC-group whose activities $x_1, \ldots, x_m$ are not in σ, there is a $\sigma' \in T$ such that $(x_i, p_{x_i}) \in pp^{O,\sigma'}(x_i)$ for all $x_i \in \{x_1, \ldots, x_m\}$. Consequently, we just have to include those (x_i, p_{x_i}) and the values of the data fields on which they depend in S to make it $R - valid(S)$.

2. Let $S \in \mathcal{S}$ be a $R - valid$ tuple of a multi-set of activity instances and a data state δ. To prove the second part we have to find a $\sigma \in T_a$ such that $\rho^\sigma(a) = \rho^S(a)$.

Since there are no dead activities in the BP, we know that there exists at least one $\sigma' \in T_a$ such that $\#_a^{\sigma'} > 0$. In addition, since the BP is an R3C-process, we have that for all $x \in ACg(a)$, it holds that $\#_x^{\sigma'} > 0$ and that $\sigma' \underset{D_{ACg(a)}}{\equiv^D} S$. Therefore, by Lemma 3, we just need to make sure that $\rho^{\sigma'}(x) = \rho^S(x)$ for all $x \in ACg(a)$ to fulfill $\rho^\sigma(a) = \rho^S(a)$.

The only problem may appear if there is not any process execution σ' with the same activity instances as in S for some $x \in ACg(a)$. One reason for this may be that the activities at hand are in sequential order in a loop and, hence, they must always be executed the same number of times, whereas this restriction does not apply to the activity instances in S. However, since the BP is an R3C-process: (1) this problem can only appear if there are more than one activity instance for an activity,[16] and (2) if that is the case, the number of times the activity is executed is unbounded, which means that one can always find a

[16]If they are executed just once it is a valid execution by definition of R3C-process

σ'' that has the same activity instances as S for any $x \in ACg(a)$ and adds new activity instances (x, p_x) with $p_x \in O_x^S$ as necessary. Consequently, σ'' is resource-equivalent with S and, hence, $\rho^{\sigma''}(x) = \rho^S(x)$ for all $x \in ACg(a)$ by Lemma 2. □

Theorem 2. *For any* R3C-process *bp, it holds that bp is consistent* $\Leftrightarrow$ *bp is* α-consistent

Proof. $\Rightarrow$ To prove that *bp* is α-consistent, we have to find an $S \in \mathcal{S}$ such that $R - valid(S)$ and $\forall a \in A(\#_a^S = 1)$. *bp* is an R3C-process, which means that for each AC-group $= \{ac_1, \ldots, ac_n\}$ of *bp*, there is a process execution σ_i in which all of the elements of ac_i are executed just once and all the activities that use the same data field as the elements of ac_i are executed at least once. Furthermore, since *bp* is consistent, we know that there is an $R - valid(\sigma_i')$ for any possible sequence of execution of activities of *bp*; in particular for $\sigma_1, \ldots, \sigma_n$. Finally, according to Lemma 3, we have that the people that meet the resource assignment of a are not influenced by the activity instances of any activity $x \in A$ that does not belong to the AC-group of a. Thus, we can obtain S by taking from each σ_i' the activity instances that correspond to the activities that belong to each ac_i and the data fields used by them ($S = (\{ai \in \sigma_i' | \pi_a(ai) \in ac_i\}, \delta)$ for all $1 \leq i \leq n$, where $\delta \in \Delta$ such that $S \underset{D_{ac_i}}{\equiv}^D \sigma_i'$).

$\Leftarrow$ Since the process is α-consistent we already have a valid allocation for each activity a considering that all activities of its AC-group are executed just once. Furthermore, by Lemma 2 we know that keeping the same people allocated to the same activities regardless of the number of repetitions of the instances of the process does not change the people that meet the resource selection condition, and by Lemma 3 we have that the people that meet the resource selection condition of a are not influenced by the activity instances of any activity $x \in A$ that does not belong to the AC-group of a. Therefore, for all $\sigma \in \Sigma$ it is possible to find a $\sigma' \in \Sigma$ such that $R - valid(\sigma')$ just by keeping the same data fields and the same people allocated to the same activities as in the allocation that considers that all activities of the AC-groups whose activities are executed in σ', are executed once. □