Enhancing the Management of Resource-Aware Business Processes

Cristina Cabanillas Macías

Supervisors:
Dr. Antonio Ruiz Cortés
Dr. Manuel Resinas Arias de Reyna

International Doctoral Dissertation

First published in December 2012 by
Cristina Cabanillas Macías

http://www.isa.us.es/members/cristina.cabanillas
cristinacabanillas@us.es

Support: Pre-doctoral scholarship, and scholarships for research visits granted by the University of Seville. Additional support for attending conferences provided by the European Commission (FEDER) and the Spanish Government under CICYT project SETI (TIN2009-07366), and the Andalusian Government projects ISABEL (TIC-2533) and THEOS (TIC-5906).

Universidad de Sevilla

The committee in charge of evaluating the dissertation presented by Cristina Cabanillas Macías in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Software Engineering, hereby recommends _____ of this dissertation and awards the author the grade _____.

Miguel Toro Bonilla Catedrático de Universidad Universidad de Sevilla	Xavier Franch Gutiérrez Profesor Titular de Universidad Univ. Politécnica de Barcelona
Mathias Weske Full Professor University of Potsdam	Jan Mendling Full Professor WU Vienna

Félix Oscar García Rubio
Profesor Titular de Universidad
Univ. de Castilla La Mancha

To put record where necessary, we sign minutes in _____, _____

Choose a job you love, and you will never have to work a day in your life.

Confucius (551 b.C–479 b.C),

Philosopher

To Jesús, for his understanding and patience.

To my family, for their unconditional love and support.

Acknowledgements

This is supposed to be the easiest section to write in a doctoral thesis, but finding the right words to describe my gratitude to the people that have been somehow involved in this work, takes its time.

First of all, thanks to my thesis supervisors, Antonio Ruiz Cortés and Manuel Resinas. You trusted in me from the beginning, when you did not even know me. Antonio, talking to you has always been a shot of energy and self-esteem, empowering my will to work and to excel myself. Your passion for your job is transmitted to the people around you. Manolo, working with you has been very easy. Thanks for caring about every detail in papers, you have always found the right way to get a "perfect" finish. You did a great co-supervision of this thesis. You both have been fundamental to make this happen.

Secondly, I need to mention my colleagues, especially the so-called ISA-groupis, most of them fellow team members in the PhD thesis adventure. I am grateful to all of you for your kind welcome when I joined the group almost four years ago. You made me feel home from the first minute. Pablito, thank you for your attention from the very first day I was in Seville, and for your friendly advice throughout these years, even though I did not always put in practice every recommendation ;-) Adela, thanks for being my confident in hard times. Guti, I have really enjoyed our chats and hours of work side by side in the office, and I know I can count on you. To the rest of members of the ISA group, as well as to my colleagues from the Department of Computer Languages and Systems, with whom I have shared less working hours but many conversations and laughs, thanks for your friendship and support.

In third place, I would like to express my gratitude to all the colleagues I have met along this path, and who have contributed to the development of my thesis to more or less extent. In particular, I am specially grateful to Prof. Dr. Fabio Casati, for treating me like a team member of his research group during my four months in Trento. Grazie per tutto, Fabio, ho imparato tantissimo da te. Grazie a te ricorderò sempre che "life is good" :-) Thanks also to all my colleagues and friends from the Department of

Information Engineering and Computer Science of the University of Trento, for your impeccable manner with me. A special mention to Prof. Dr. Jan Mendling, too. Thanks for your valuable research feedback and your advice on how to deal with large amount of work and stress. Honestly, vielen Dank!

And, as the best must be saved for the end, it is time to say some words in Spanish. Gracias a mi familia por su apoyo incondicional. En especial, gracias mamá por quererme tantísimo, por confiar en mí, animarme en los momentos malos y disfrutar conmigo los momentos buenos. Gracias Betty, por demostrarme muchas cosas aunque no me las digas con palabras. Aquí tienes y tendrás a tu hermana para lo que necesites. Gracias también a mis tíos y primos, en especial a mis tíos César y Emilia, y a mis primos César y Laura, mis segundos padres y hermanos. Os siento cerca aunque no lo estéis. Por último, unas gracias especiales para Jesús, el principal perjudicado durante estos años, sufridor de los daños colaterales que supone hacer una tesis doctoral, principalmente la falta de tiempo juntos. Espero poder recompensarte las horas de espera y los días de aburrimiento mientras yo trabajaba. Me has apoyado siempre en mis decisiones de trabajo, has entendido mis agobios y mis noches en vela de manera admirable y te has preocupado mucho por mí. Tu apoyo ha sido fundamental para poder seguir trabajando cuando ya no tenía fuerzas. Mil gracias.

Siento también la necesidad de mencionar a mi padre, que me ha visto en este camino desde un sitio privilegiado. Pensar en tí y en hacerte sentir orgulloso me ha ayudado en los momentos difíciles y me ha reconfortado en los buenos. Me gustaría muchísimo poder compartir esta alegría contigo y con los abuelos. Espero que os sintáis felices.

Abstract

Providing support for *Business Process Management (BPM)* requires taking into consideration several perspectives, mainly control flow, data, and resources, which are involved throughout the entire *Business Process (BP)* lifecycle. In this thesis, we focus on the management of the BP resource perspective, specifically on *human* resource specification in BPs, and on design-time and run-time *automated* resource analysis in BPs. *Resource specification in BPs* refers to the definition of the resource assignments associated to the process activities, and to how they are bound to the information in the BP models. *Resource analysis in BPs* copes with the execution of analysis operations over resource-aware BP models with the aim of extracting information such as what activities can be allocated to a specific person at run time.

In this thesis we argue that *traceability*, *expressiveness* and *binding flexibility* are three desirable features in BP resource specification, and we define *eleven analysis operations* for which automated support is convenient, some of which have to do with the relationship between persons and activities, and others are concerned with controlling how data are accessed by people. A study of current approaches has revealed that these features and operations are only partially supported so far.

To overcome the shortcomings identified we introduce *Resource Assignment Language (RAL)*, a language to define resource assignments for the activities of a BP. RAL expressions are *traceable* to the concepts of the organisational model of the company. Furthermore, RAL is *expressive*, as it allows specifying a large variety of assignment patterns for any task duty associated to a process activity, such as who is the person responsible for performing the work, or who is accountable for the activity. As aforementioned, BP resource specification needs to be bound to the BP model to which it refers in order to be considered during process execution. We provide an all-in-one binding approach to use RAL with BPMN 2.0, and an alternative approach to model all the resource-related information in a so-called RACI matrix extended with binding information, expressed with RAL. By that means, we provide *binding flexibility* to the organisation. Besides, an automated procedure to switch from the latter binding approach to the former has been developed.

Regarding resource analysis in BPs, we have endowed RAL with a *formal semantics* based on *Description Logics (DLs)* that eases the automated execution of analysis operations related to the BP resource perspective. Relying on RAL semantics, we present a *DL-based reference implementation* of the eleven analysis operations identified, both at design time and at run time. This implies providing support for analysing the BP resource perspective in isolation, and developing the required extensions for RAL semantics to take into consideration the control flow and the data perspectives as well.

Finally, *Collection of Resource-centrIc Supporting Tools And Languages (CRISTAL)* is a system developed from the contributions of this thesis to evaluate our research results. We have also validated part of the analysis functionalities in a transfer project with a multinational company.

Resumen

Proporcionar soporte para la *Gestión de Procesos de Negocio* requiere tener en cuenta varias perspectivas, principalmente el flujo de control, los datos y los recursos, que intervienen a lo largo de todo el ciclo de vida de los Procesos de Negocio. En esta tesis nos centramos en la gestión de la perspectiva de recursos de los procesos de negocio, en concreto en la especificación de recursos *humanos* en procesos de negocio y en el análisis *automático* de recursos en procesos de negocio en tiempo de diseño y en tiempo de ejecución. La *especificación de recursos en procesos de negocio* trata de la definición de las asignaciones de recursos asociadas a las actividades de los procesos y de cómo se vincula esta información a los modelos de proceso. El *análisis de los recursos en procesos de negocio* aborda la ejecución de operaciones de análisis en modelos de procesos con información de recursos, con el objetivo de extraer determinada información, como qué actividades pueden ser asignadas a una persona específica durante la ejecución de un proceso.

En esta tesis argumentamos que la *trazabilidad*, la *expresividad* y la *flexibilidad de vinculación* son tres características deseables en la especificación de recursos en procesos de negocio y definimos *once operaciones de análisis* para las que es conveniente proporcionar soporte automático, algunas de las cuales tratan sobre la relación entre personas y actividades, y otras están relacionadas con el control de acceso a los datos por parte de las personas. Un estudio de las aproximaciones actuales ha revelado que estas características y operaciones sólo están parcialmente soportadas en la actualidad.

Para solventar las deficiencias identificadas introducimos *Resource Assignment Language (RAL)*, un lenguaje para definir asignaciones de recursos para las actividades de los procesos de negocio. Las expresiones RAL son *trazables* con los conceptos del modelo organizacional de la empresa. Además, RAL es *expresivo*, en tanto en cuanto permite especificar una gran variedad de patrones de asignación para cualquier deber relacionado con una actividad, por ejemplo quién es el responsable de realizar el trabajo o quién es responsable de aprobar la ejecución de la actividad. Como se ha dicho anteriormente, la especificación de recursos en procesos de negocio necesita vincularse al modelo de proceso al que se refiere para poder ser considerada durante la ejecución

del mismo. Nosotros proporcionamos una aproximación todo-en-uno para usar RAL con BPMN 2.0 y una aproximación alternativa para modelar toda la información de recursos en una matrix RACI extendida con información de vinculación, expresada con RAL. De este modo, proporcionamos *flexibilidad de vinculación* a la organización. Además, hemos desarrollado un procedimiento para pasar automáticamente de la segunda aproximación a la primera.

En cuanto al análisis de recursos en procesos de negocio, hemos provisto a RAL de una *semántica formal* basada en Lógicas Descriptivas que facilita la ejecución automática de operaciones de análisis sobre la perspectiva de recursos de los procesos. Haciendo uso de la semántica de RAL, presentamos una *implementación de referencia basada en lógicas descriptivas* para las once operaciones de análisis identificadas, tanto en tiempo de diseño como en tiempo de ejecución. Esto implica proporcionar soporte para analizar la perspectiva de recursos individualmente y desarrollar las extensiones de la semántica de RAL necesarias para tener en cuenta también la perspectiva de flujo de control y la de datos.

Por último, se ha desarrollado un sistema llamado *Collection of Resource-centrIc Supporting Tools And Languages (CRISTAL)* a partir de las contribuciones de esta tesis para evaluar los resultados de nuestra investigación. También hemos validado parte de la funcionalidad de análisis en un proyecto de transferencia con una empresa multinacional.

Contents

List of Figures

List of Tables

Part I

PREFACE

1

Introduction

"It is during our darkest moments that we must focus to see the light"

Aristotle Onassis (1906–1975),
Businessman

In this thesis, we report on our contributions to develop a set of techniques and tools to enhance the management of human resources in business processes at different phases of the business process lifecycle, and considering the rest of perspectives involved in business process management. In this chapter, we introduce the research context and we outline the work performed. In Section §1.1, we describe the concepts of the research context required to understand the contributions of the thesis, which frame the scope of our work. In Section §1.2 we present the research goals pursued in the thesis, for which we have developed the approaches that are summarized in Section §1.3. Section §1.4 introduces a brief description of the context in which the thesis has been developed, and outlines the main publications achieved. Finally, in Section §1.5 we describe how the content of this dissertation is organised.

1.1 Research Context

A *Business Process (BP)* is a collection of activities that are executed in a logical order through time to achieve a specific goal within an organisational and technical environment [137]. *Business Process Management (BPM)* is aimed at supporting *Business Processes (BPs)* using methods, techniques, and software to design, enact, control and analyse operational processes involving humans, organisations, applications, documents and other sources of information [127]. There exists a growing interest in BPM from academia and industry. Indeed, more and more nowadays organisations tend to adopt a process-oriented perspective, and they are increasingly caring about providing proper BPM. Good management helps save effort and time, and it contributes to increase the quality of service offered by the company.

At least five different *perspectives*, also known as *dimensions*, have been identified in BPM [48, 57, 137]. Among them, the BP control flow, data, and resource perspectives have received special attention in academia [8]. *Control flow* refers to the order in which the activities of a process are executed. The *data* perspective involves the information (a.k.a. data objects) that may be accessed and manipulated in a process to complete the activities. *Resources* represent the extra support required in some activities, which usually translated into the use of non-human or human resources in the process. The former involves software applications, external devices and any system that can be necessary to complete an activity. The latter refers to the human cooperation needed to accomplish or approve work involved in BP activities.

The BP perspectives are involved in all the phases of the *BP lifecycle*. According to the description provided by Weske [137], the BP lifecycle is composed of 4 phases. The Design and Analysis phase deals with the design, representation, and validation of BPs. The Configuration phase is in charge of for implementing and preparing the system where processes are deployed and *process instances* are run during the Enactment phase, which controls the execution. Finally, the Evaluation phase uses the run-time information stored in history logs, as well as design-time information that may be necessary, to identify potential improvement points that help to re-design the BPs and re-start the lifecycle[1].

There are two issues that stand out in the BP lifecycle, namely BP *specification* and *analysis*. The former is related to the Design and Analysis phase, and deals with BP

[1]Notice that although the structure of the BP lifecycle is cyclic, there is not a strict temporal ordering in which the phases need to be executed [137].

modelling and representation. According to Weske [137], a *BP model* is a set of activity models and execution constraints between them. The graphical representation of a BP model is called *BP diagram*. The de-facto standard for BP modelling is *Business Process Modelling Notation (BPMN)* [94]. However, languages such as *Unified Modeling Language (UML)* [62] and Petri Nets [89], also allow the definition of BP diagrams.

Regarding *analysis*, BP analysis consists of carefully examining activities, tasks, durations, responsibilities, documents, and bottlenecks inherent in administrative, operational, and project support processes [45]. It can be applied at several phases of the BP lifecycle. Specifically, at the Design and Analysis phase, analysis (a.k.a. *design-time analysis* in this context) may be convenient in order to verify that the process has been designed according to the requirements established and to detect potential problems that may lead to unexpected behaviour at run time. During the Enactment phase, analysis (a.k.a. *run-time analysis*) mechanisms can be used to foresee potential problems that can be avoided or solved in time. Finally, BP analysis at the Evaluation phase (a.k.a. *post-mortem analysis*) aims at identifying issues that can be improved in BPs, for which process mining techniques are typically used [130]. Summing up, BP analysis helps improve efficiency, increase organisational effectiveness, reduce costs, and optimize work [45].

In this doctoral thesis, we focus on the *BP resource perspective* and, in particular, on providing support for *human resource*[2] specification and analysis in BPs. Regarding resource specification, the resources of an organisation are typically structured in an *organisational model* according to their characteristics and permissions. Common concepts in organisational models involve person, role and organisational unit. However, there is a plethora of different organisational metamodel that can be used to structure an organisation. Taking into consideration this structure, resources can be assigned to the different *task duties* associated to each BP activity. For instance, there may be a person that performs the activity, and another person in charge of approving the work as a requirement to consider the activity accomplished. We have identified up to five different task duties for an activity, namely Responsible, Accountable, Support, Consulted, and Informed. A set of twelve *assignment patterns* have been identified to characterize the types of resource assignments that can be associated to the task duties of an activity, which range from simple assignments, e.g. based on roles or skills; to complex assignments that can contain constraints with respect to the assignments related to other activities, e.g. the well-known separation of duties (SoD) constraint to indicate that two activities cannot be performed by the same person with the aim

[2]Term *resource* might be used to refer to *human resources* throughout this thesis.

of avoiding conflicts of interests. Finally, the last issue to be considered is how much resource-related information is modelled in the BP diagram together with the BP control flow and data perspectives, and how it is bound to it. In this regard, we have identified three different *binding strategies*, to be named all-in-one, split, and separate binding (cf. Section §3.2.4).

As far as resource analysis is concerned, the *automated* analysis of the BP resource perspective can be defined as the computer-aided extraction of information from BP models and/or other models that contain information related to the resources associated to the BP activities. It can be defined in terms of *analysis operations*. In particular, the design-time analysis of BPs is usually performed from the information represented in process diagrams and similar models in order to infer information such as who will potentially be allowed to perform the task duties associated to the activities of the BP, if there is somebody that can participate in all the activities, which information a person is allowed to access given the activities he/she is assigned to, and the like. From the result of this analysis, we can make decisions and modify the model to make it compliant with our expectations.

Run-time analysis serves for controlling how resources are actually managed in the Enactment phase of the BP lifecycle. For instance, we can check if there is somebody that no longer participates in a process instance, or the persons of the company that are likely to change the content of the data objects handled in the BP. We have identified a total of eleven analysis operations, some of which are related to resource management focused on the relation of people and activities, e.g. the calculation of the activities that can be allocated to a specific person in a single BP instance; and the rest are related to resource management focused on how people access data, e.g. obtaining which information a specific person can read in a BP instance. Naturally, the BP resource perspective is involved in all the operations, but in some cases the control flow and/or data perspectives must be considered as well. For instance, sometimes it is necessary to check whether an activity is executed in all the possible process instances, which is related to control flow. Similarly, data is involved in the operations that deal with information access control.

1.2 Thesis Goals

This work strives to improve the support for resource specification and analysis in BPs. We have studied the state of the art with regard to all the issues described in the

previous section, and we have identified a set of problems that constitute the goals to be achieved in this thesis. In particular, the lack of traceability, expressiveness, and binding flexibility appear as three sources of problems to be addressed by resource specification approaches. With regard to resource analysis, providing support for the eleven analysis operations identified is an appealing challenge that, to the best of our knowledge, has only been partially addressed so far.

Thus, regarding resource specification in BPs, we aim at providing a mechanism to assign resources to the BP activities on the ground of the organisational model of the company where the process is used. This would provide resource assignments that are *traceable* to the concepts involved in an organisational model. Furthermore, we intend to support the specification of a large variety of resource assignments involving simple expressions, complex expressions, and constraints between the expressions of two activities, which can be associated to the different task duties associated to the BP activities. This would provide high *expressiveness* to the resource specification approach developed. Finally, we aim at offering *flexible binding* of the resource specification to the organisation, which implies developing approaches for several binding strategies in order to enable the system administrator to select the one that best meets their expectations and needs. The automated switch between different binding strategies is also desirable.

Regarding the design-time and run-time analysis of the BP resource perspective, we aim at developing a technique that allows the *automated* execution of the catalogue of analysis operations identified, both at the Design and Analysis phase, and at the Enactment phase of the BP lifecycle. This involves providing support for the analysis of the BP resource perspective in isolation, and in combination with control flow and/or data. Our objective is to provide a reference implementation to show that the automation of the execution of such analysis operations is feasible at design time and at run time. Efficiency is not a must, although it is desirable.

1.3 Solution Proposal

Our approach to address the previous goals is highly inspired in some solutions proposed for automated management problems in feature models [19] and *Service Level Agreements (SLAs)* [103], as well as in *Model Driven Development (MDD)* techniques [97]. The overall result is conceptually called *Collection of Resource-centrIc Supporting Tools And Languages (CRISTAL)*, a system composed of a resource assignment language and

a set of tools that allow: (i) the specification of resources in BP models using two different binding strategies; and (ii) the automated execution of the eleven target analysis operations at design time and at run time. The specific contributions of this thesis are depicted in Figure §1.1 and summarized below.

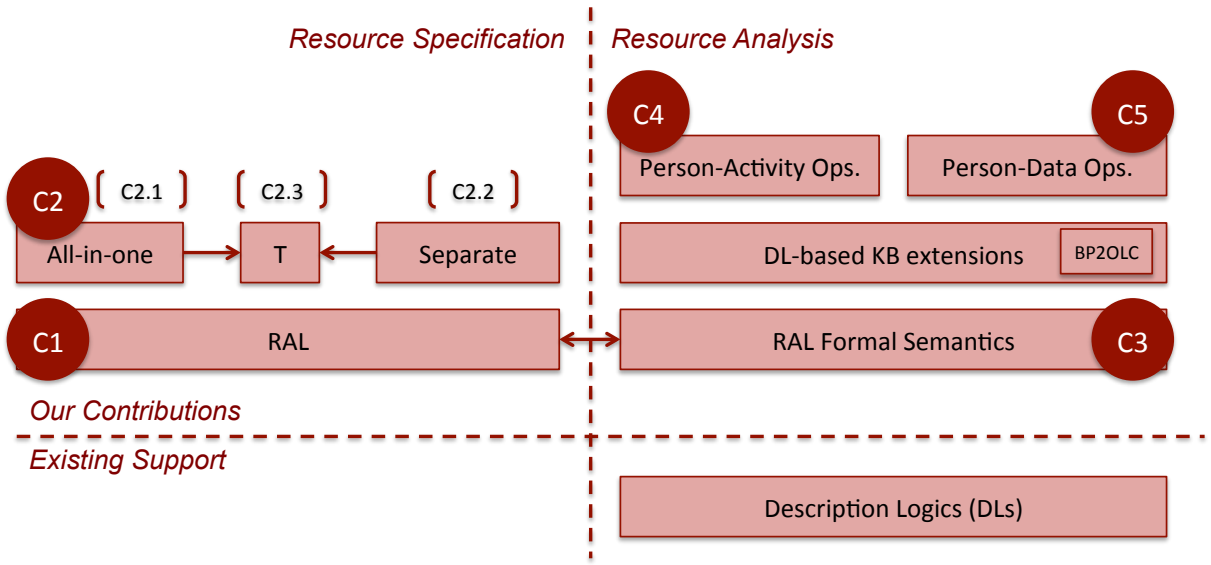


Figure 1.1: Overview of the contributions of this doctoral thesis

Contributions regarding Resource Specification

Our first contribution is the development of a new language called *Resource Assignment Language (RAL)* [30, 35], which allows us to specify resource assignments that can be used with any task duties that can be associated to a BP activity. The language has been developed on the ground of a well-known organisational metamodel described by Russell et al. [105] that handles the concepts of person, capability, position, role and organisational unit. Thus, RAL expressions are *traceable* with the organisational model in the company, as long as it considers the aforementioned concepts. Furthermore, RAL is *very expressive*, according to the criteria we have used to evaluate similar approaches from literature on resource specification.

The second contribution of this thesis is three-fold. On the one hand, we have used RAL with BPMN 2.0 to provide an all-in-one binding approach. It makes use of the resource assignment mechanisms offered by the BP modelling standard [94], with no need of extending or modifying its metamodel or semantics. On the other hand, pursuing the goal of binding flexibility, we have developed a separate binding approach that is grounded on the use a RACI matrix that is extended with the so-called *binding*

information [34] for resource specification. RACI matrices are a mechanism to specify the resources that perform different task duties associated to activities [118]. In particular, RASCI matrices, a variant of the original RACI, deal with the 5 task duties considered in this thesis. RAL can be used to define the binding information. Finally, we have developed a procedure based on transformations that manage to automatically shift from one separate binding approach to the all-in-one approach, maintaing all the resource-related information of the source model.

Contributions regarding Resource Analysis

The first contribution aimed at providing support for automated resource analysis, consists of the definition formal semantics for RAL [27]. This provides the RAL expressions with precise meaning, and eases the automated extraction of information from *resource-aware BP models* that use RAL for resource specification (a.k.a. *RAL-aware BP models*). In particular, RAL semantics is based on *Description Logics (DLs)*.

Next, relying on RAL semantics, we have developed an approach to automatically analyse how resources are managed in a BP. In particular, we have developed automated support for the eleven analysis operations identified, which has been divided into two different modules: one module for the resolution of the analysis operations focused on the relation of people and activities (Person-Activity Operations), and the other dealing with the analysis operations focused on the relation of people and data (Person-Data Operations). As an intermediate step, extensions of the DL-based *Knowledge Base (KB)* created to define RAL semantics have been developed in order to prepare it for automated operation resolution. A reference implementation based on DLs has been developed to support the execution of all the operations at design time and at run time.

The specification and analysis contributions have been implemented in CRISTAL and partially validated in the management system of a multinational company, in the scope of a project in which we have implemented and put together mechanisms for resource analysis introduced in this thesis.

1.4 Thesis Context

This thesis has been developed in the context of the BPM research area of the Applied Software Engineering (Ingeniería del Software Aplicada-ISA) research group of the University of Seville (Spain). However, as a holder of a pre-doctoral research schol-

arship of the University of Seville, my research has not been developed in the scope of a specific research project. Instead, I have participated in several projects during my PhD, which has given rise to publications and contributions related to more than one topic, always associated to BPM. The results of my participation in the different projects are the following:

- ISABEL: Ingeniería de Sistemas Abiertos Basada en LínEas de productos. Excellence project of the Andalusian Government. My participation in this project was focused on researching on the management of the BP data perspective. As a result, we published a survey on data anomalies that can arise in BPs [26], and a procedure to automatically generate a data-centered view of BPs [31], which has been useful as intermediate state for part of the analysis capabilities developed in the scope of this thesis.

- SETI: reSearching on intElligent Tools for the Internet of services. Research and Development project of the Spanish Government. The work packages in which I participated in this project were related to the management of compliance between business rules and BPs. In this context, I studied the state of the art on BP compliance, which gave rise to the publication of a survey on how to deal with compliance in BPs [25], a survey on the features that a compliance management system should have in order to provide full support for BP compliance [28], and the development of an approach to specify and automatically check compliance rules at design time and at run time [37].

- THEOS: Tecnologías Habilitadoras para EcOsistemas Software. Project funded by the Andalusian Government from March 2011 to March 2015. Most of the research results of this thesis have been produced in the scope of this project. Specifically, I have reviewed the literature on resource management in BPs, focused on resource specification and analysis. The consequence has been the development of the contributions summarized in Section §1.3, which can be found at [27, 29, 30, 32, 33, 34, 35, 36].

- BPCMS: Implantación de un Sistema de Gestión de la Conformidad para Procede. Project funded by a well-known multinational company[3]. Another result of the aforementioned project THEOS was the development of this transfer project, whose goal was to develop a system to ease the definition and the automated

[3]Name is not provided due to privacy reasons.

checking of business rules. I was involved in the elicitation of requirements, designing of the architecture, analysis of solutions, and implementation stages. We applied the results of previous research on BP compliance, and the publication of the results of this project is part of my post-doctoral goals.

1.5 Structure of this Dissertation

This dissertation is organised as follows:

Part I: PREFACE. It comprises this introductory chapter, in which we have introduced the research context of the work developed necessary to understand the content of the thesis, we have defined the goals to be achieved with this work, we have summed up our contributions to overcome the problems identified, and we have described the context in which the thesis has been developed.

Part II: BACKGROUND INFORMATION. It provides specific information about the elements involved in the scope of the research context of our work. Specifically, in Chapter §2 the concepts related to BPM that are handled in this thesis are introduced. Afterwards, in Chapters §3 and §4 we provide a detailed description of the issues involved in resource specification and analysis in BPs, respectively. In each of these chapters, we also present a summary of a study conducted considering all the issues identified for the related approaches found in literature.

Part III: OUR CONTRIBUTION. This is the core of the thesis and it is organised in seven chapters. First of all, in Chapter §5 we define the problems identified regarding the specification and analysis of the BP resource perspective from the results of the studies described in previous chapters, and we review the literature on the topic with regard to each and every problem identified. Then, we present our approaches to overcome the problems. In Chapter §6, we introduce RAL, the resource assignment expressions it allows defining, and the results of the assessment of RAL's expressiveness. In Chapter §7 we explain how RAL can be integrated into BPMN to provide an all-in-one binding of resource specification, we present a separate binding approach grounded on the extension of the RACI matrices with binding information defined in RAL, and we introduce a procedure to switch from the latter approach to the former one in an automated way. Chapter §8 presents RAL formal semantics, with a detailed description of the mapping to DLs developed. Chapter §9 describes the DL-based implementation we propose

to execute the analysis operations that deal with the resource between people and BP activities, both at design time and at run time. Finally, Chapter §10 describes a similar approach that address the design-time and run-time automated execution of analysis operations focused on the relation of people and data in BPs. This contribution block ends up in Chapter §11 with a description of CRISTAL, the tool support for the contributions of the thesis, and the scenario in which it has been tested and validated.

Part IV: FINAL REMARKS. Chapter §12 concludes the thesis with a summary of the contributions and their added value, a definition of the limitations and the future work identified, and a summary of the publications that have been achieved from the approaches described throughout this manuscript.

Part V: APPENDICES. Three appendices have been attached to this thesis to complement the content of some of its chapters. In particular, Appendices §A and §B contain a one-by-one description of the approaches considered for the studies summarized in Chapters §3 and §4, respectively, as a complement of the background information. Appendix §C introduces a brief definition of the main concepts of DLs, useful to ease the understandability of the DL formulas introduced in Chapters §8, §9 and §10.

Part II
BACKGROUND INFORMATION

2

Business Process Management

"Business, that's easily defined - it's other people's money"

Peter Drucker (1909–2005),
Businessman

"If you can't describe what you are doing as a process, you don't know what you are doing"

W. Edwards Deming (1900–1993),
American statistician and professor

"Management is nothing more than motivating other people"

Lee Iacocca (1924–),
Businessman

Business process management (BPM) has received considerable attention recently by both business administration and computer science communities [137]. This "hot topic" is introduced in Section §2.1. Section §2.2 defines business processes (BPs) and business process (BP) models, accompanied by an example and a brief introduction to BP modelling notations. The main perspectives involved in BPs and, thus, represented in the BP models, are described in Section §2.3. A description of the phases that make up the BP lifecycle can be found in Section §2.4. The components and human roles that participate in a Business Process Management Systems (BPMS) are outlined in Section §2.5, which precedes the summary of the chapter presented in Section §2.6.

2.1 Introduction

According to van der Aalst et al. [127], *Business Process Management (BPM)* intends to support BPs using methods, techniques, and software to design, enact, control, and analyse operational processes involving humans, organisations, applications, documents and other sources of information. It can be seen as a principle to manage business, thus a company provides products or services to the market, which are the outcome of a number of activities performed, and BPs are the key instrument to organise these activities and to improve in general their relationships [137].

BPM is gaining increasing interest from both academia and industry. Many companies are taking this process-oriented perspective in their business, as a way of identifying which steps really create value, who is involved in the process and which is the exchanged information; ultimately, finding out how to improve, where to increase quality, reduce waste or save time [10].

The goal of this chapter is to provide an overview of the major concepts related to BPM, focusing on those aspects that are most interesting and useful for the purpose of this dissertation.

2.2 Business Processes

A process is a set of *activities* related to each other targeted at the same goal. This may be the simplest and most straight definition of a process. Consciously or not, we participate in plenty of processes daily. For instance, cooking a recipe is a process in which a set of ingredients have to be cooked and mixed in a specific order to obtain the desired dish. When the process is performed in the scope of an organisation, it is called *business process (BP)*. Thus, a BP is a collection of activities that are executed in a logical order along time to achieve a defined goal within an organisational and technical environment. To do so, they take one or more kinds of input and create an output that is of value to the customer or the market [46, 72, 137]. As stated by van der Aalst et al. [127] and Decker [48], the basis of BPM is the explicit representation of BPs. Representing a BP helps to discover weaknesses related to the order in which activities are performed, the information that is handled, and any other issue involved in BP execution. Hence, an easy-to-understand-and-use, editable, and executable mechanism to represent BPs is convenient. *BP models* are defined for that purpose, that is, a *business process model*

is the representation of the activities, documents, people and all the elements involved in a BP, as well as the execution constraints between them [137]. It serves as a starting point to be analysed and improved before, during and after execution.

A BP can be executed an indefinite number of times in an organisation. Each execution of a process is called BP instance. As stated by Weske [137], a *business process instance* represents a concrete *case* in the operational business of a company, consisting of *activity instances*. Each BP model acts as a blueprint for a set of BP instances, and each activity model acts as a blueprint for a set of activity instances. The BPs that are *automatically* executed (totally or partially) are also known as *Workflows (WFs)*. The documents, information, or tasks involved in a WF are passed from one participant to another according to a set of procedural rules [137].

Let us exemplify all this with an example. The BP model depicted in Figure §2.1 represents the process to select the venue place for the Olympic Games, a procedure known worldwide that has been simplified for the sake of clarity. The International Olympic Committee is in charge of this process. In a nutshell, this committee first receives the applications of the cities that want to organise the Olympic Games. Each city is evaluated, so that only those cities that meet all the requirements pass to the approval step, where the candidates still need a final approval by the committee in oder to participate in the voting rounds. Once the list of candidates is ready, a first round of secret voting is carried out. If there is consensus and only one city is selected, then the winner venue is published. Otherwise, the least voted city is eliminated from the list of candidates and a new voting round is performed. This is repeated until there are only two cities left. Then, the city with a greatest number of votes wins.

The modelling notation used in the figure is BPMN, the *de facto* standard for BP modelling. Its primary goal is to provide a notation that is understandable by all business users, ranging from the process modellers that design and create the initial drafts of the BPs, to the business people who manage and monitor those processes when they are deployed and executed in a *Business Process Management System (BPMS)* [94]. Apart from BPMN, there are other notations that allow the definition of BP models, e.g. *Event-driven Process Chain (EPC)*, UML Activity Diagrams, *Business Process Execution Language (BPEL)* or *Web Service Business Process Execution Language (WS-BPEL)*, Petri Nets, WF Nets and *Yet Another Workflow Language (YAWL)*. All of them support the definition of activities, decision points with alternative paths of execution, event handling, and other elements and structures required to model a BP [12]. EPC [114] is a modelling language to specify the temporal and logical relationships between ac-

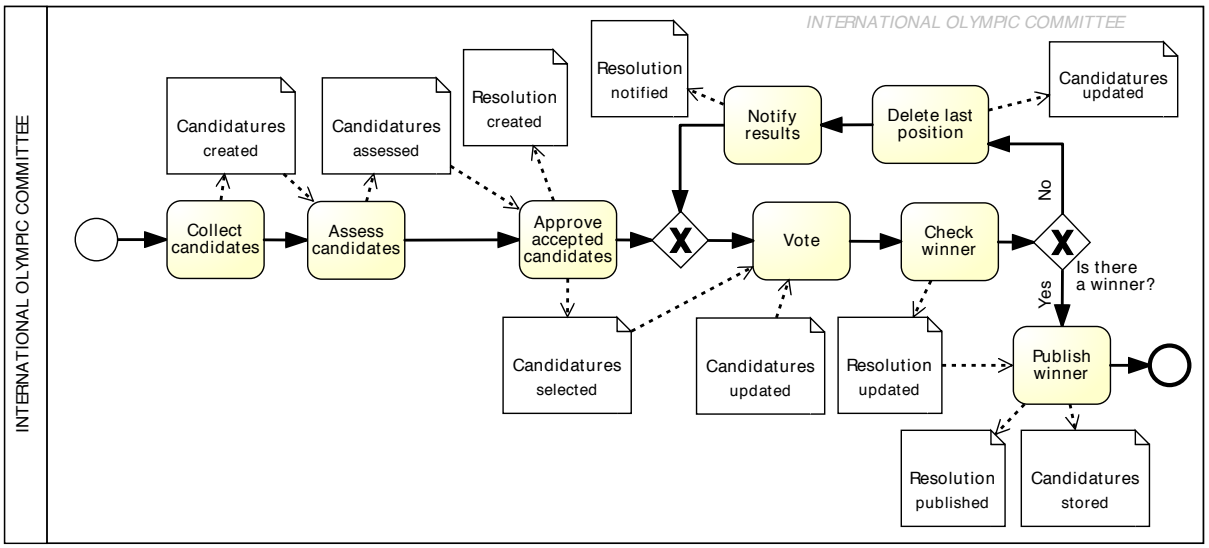


Figure 2.1: BPMN model of a BP to select the venue place for the Olympic Games

tivities of a BP. UML is a language proposed by the *Object Management Group (OMG)* for object-oriented visual modelling [62, 104], especially focused on the development of software systems. Since it was not developed specifically for BP modelling, BPs are modelled by extending the Activity Diagrams provided by the language by means of the so-called UML profiles. WS-BPEL [1] is a language that allows the specification of Executable and Abstract BPs based on Web services[1]. It is aimed at facilitating the expansion of automated BP integration in both the intra-corporate and the business-to-business spaces. Petri Nets [89, 129] are a special form of graphs that can be used to represent BPs. However, they present some limitations when dealing with complex BPs. In those cases, we can use WF Nets, a class of Petri Nets suitable for the representation, validation and verification of WF processes [68, 128]. YAWL was developed with the main aim of overcoming the limitations of Petri Nets and WF Nets for process modelling [132]. In particular, Petri Nets were extended with constructs to address the multiple BP instances, advanced synchronization, and cancellation patterns, among other issues.

BPs can collaborate with other BPs, even crossing organisational boundaries. The interaction between BPs developed in two different organisations is usually performed by means of bidirectional message interchange. In BPMN, this scenario is called *collaboration*. Please, notice that we will use the BPM vocabulary provided by BPMN [94] throughout this thesis document.

[1] According to WS-BPEL [1], "Executable BPs model actual behaviour of a participant in a business interaction, while Abstract BPs are partially specified processes that are not intended to be executed".

2.3 Business Process Perspectives

The core elements in a BP are the activities and their execution order. However, there are other elements also involved in BPs, which must be also considered when designing and modelling the processes. These elements are called *business process perspectives* or *business process dimensions*. There are typically 5 perspectives in BPs [48, 57, 137]:

The functional perspective provides a *description* of all the activities to be performed in a BP. It usually consists of a textual definition of the activity, its goal, the elements involved in its execution, and any other restriction that needs to be taken into account for its performance. According to BPMN [94], there are two types of activities: atomic activities are called *Tasks*, and decomposable activities are known as *Sub-Processes*. Activities can be executed by a system (*automated*) or by people (*manual*). Furthermore, all activities share common attributes and behaviour such as states and state transitions, that is, an activity has a *lifecycle* generally characterizing its operational semantics. There is not consensus in literature on the states of the lifecycle of a BP activity. Several proposals can be found in [94, 106, 137]. The excerpt of the activity lifecycle depicted in Figure §2.2, which is a simplification of a generic lifecycle focused on the states dealing with resource allocation, will be our reference activity lifecycle. In addition, the terms *activity* and *task* will be used indistinctly to refer to activities, disregarding whether they are actually sub-processes or tasks as defined by BPMN.

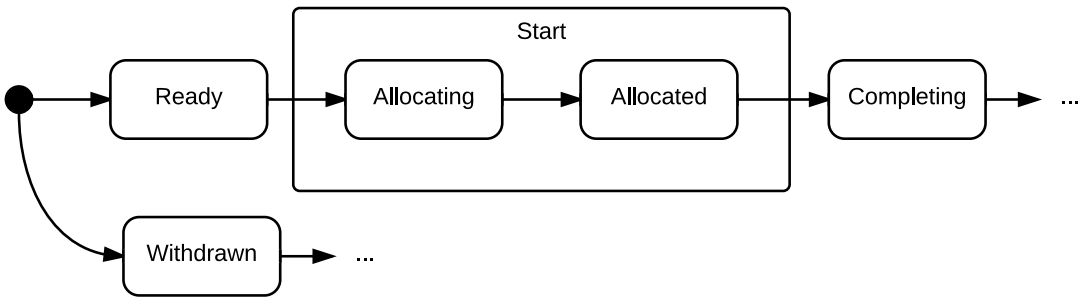


Figure 2.2: Excerpt of the BP activity lifecycle focused on resource allocation

The control flow perspective (also known as (a.k.a.) *behavioural dimension*) specifies the *control flow* dependencies between these activities, that is, the order in which the activities of a process must be performed, e.g. some candidate cities must be selected *before* carrying out a voting round. The BP modelling languages usually represent the control flow by means of arrows that connect the process elements

(i.e. activities, control flow structures, events) to define the dynamic behaviour of the process.

The resource perspective (a.k.a. *organisational dimension*) focuses on the *people, roles, organisational units and any other entities of the organisational model of a company* that are involved in a process, e.g. how the members of the International Olympic Committee are structured in the organisational model. That is, this dimension is focused on the management of the *resources* that participate in a process, in particular the *human resources* involved in the BP activities. The people interacting with the BP should also be modelled and associated to the process diagram in order to enable the automation of the allocation of work to specific persons at run time. We delve into how resources can be specified in BPs in Chapter §3.

The data perspective (a.k.a. *informational dimension*) defines the information that must be produced or consumed by activities, that is, the *data* handled in the process, e.g. a document containing the description of the candidate cities, or a report with the summary of each voting round during the selection process. The execution of BP activities may involve managing a large amount of data that can flow throughout the process, which can and should also be represented in the BP models. Notations such as BPMN provide two forms of modelling data [94]. First, as *data objects* attached to the process activities and/or to control flow elements. Each data object can appear multiple times in a BP diagram, but each of these appearances references the same data object instance. Optionally, data objects can be associated to *states* (a.k.a. *data states*), representing the state of the information they contain (cf. Figure §2.1). Second, as *data stores* representing *repositories* from which activities can take information, and where activities can leave information produced as a result of the action performed. These repositories are supposed to be accessible by the process activities and/or the human resources in charge of manipulating the data used in the activities.

The technical perspective makes reference to the different tools or machines that may be required in order to perform certain activities, e.g. the voting may be done electronically so that the votes can be counted automatically. Thus, it is related to the BPMS where the process is deployed and enacted (cf. Section §2.5 for information about BPMSs).

Among these dimensions, the second, the third and the fourth ones refer to elements that can be explicitly represented within a BP model. Specifically, in YAWL

specification it is said that "a WF comprises three main perspectives: control flow, data, and resources" [8]. However, besides the aforementioned perspectives, *time* is also considered by some authors as a first-class citizen to be considered in BPM, e.g. Awad [14]. Temporal constraints, such as an activity expiration date, are handled by means of events in most BP modelling languages. Controlling these temporal constraints is usually a concern of the *BP compliance* field, whose techniques have typically focused on dealing with rules related to control flow and time, as concluded in a study we carried out about current BP compliance approaches [25].

2.4 Business Process Lifecycle

There is no consensus about the number and the name of the phases that constitute the BP lifecycle. They vary depending on the granularity degree used for identifying the phases and the way of grouping the functionality in them [12, 91, 127, 137, 138]. We will use the BP lifecycle described by Weske [137]. He proposes the four-phase BP lifecycle depicted in Figure §2.3[2]. As stated in Section §2.3, at least the BP control flow, data and resource perspectives should be considered in every BP lifecycle phase. To check whether these perspectives are being managed properly (i.e. designed, modelled, and used in the right way), *BP analysis* comes on stage. *BP analysis*[3] is any activity that helps us understand how a business unit fulfills its mission. It ensures that all business problems are addressed and reduces the risk of eliminating the benefits of existing BPs.

The BP lifecycle described by Weske [137], starts with the *Design and Analysis phase*, whose goal is to define a new BP. In case the BP already exists, then in this phase the process might be modified according to the new requirements. In either case, the informal BP description of the process (re-)designed, is translated into a particular BP modelling notation (usually a graphical one) [94, 114, 132]. Once a BP is defined, it needs to be validated to check whether all valid process instances are covered by the BP model. Furthermore, simulation techniques can help during the validation by allowing to detect possible undesired execution sequences and also to verify that the process actually exposes the desired behaviour. Finally, verification techniques allow to check correctness properties. Analysis is required for such purpose. analysing a process model after design helps prevent unexpected behaviour at run time, while enabling the

[2]Taken from Weske [137]
[3]Definition inspired in http://requirementssolutions.com/Analyze_Business_Process.html.

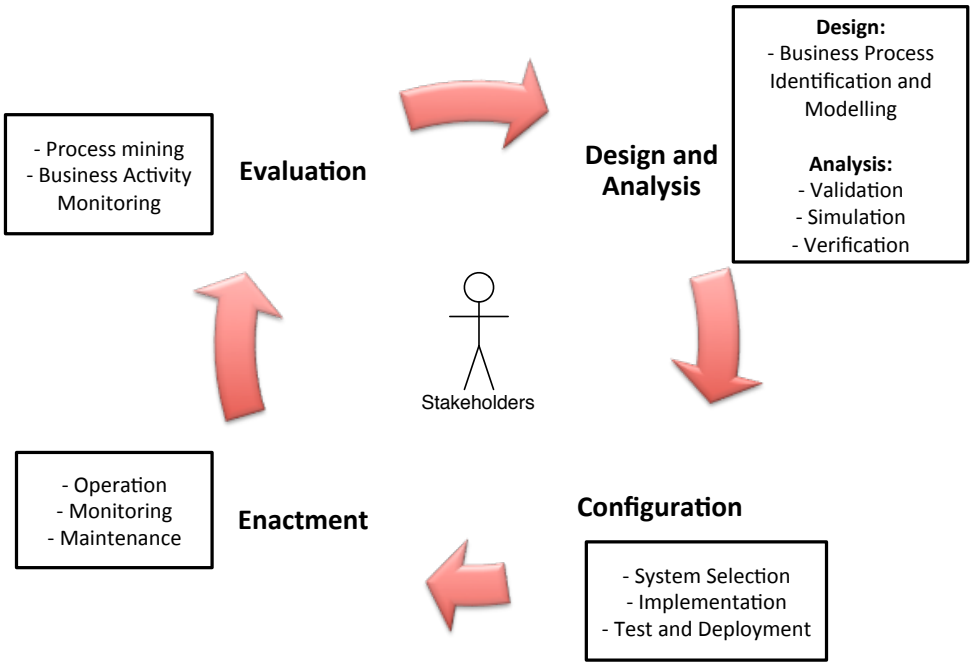


Figure 2.3: Business process lifecycle

correction of potential problems related to any BP perspective. Nowadays, there exist different techniques for design time analysis, but most of them are focused on a single BP perspective, and thus provide ad-hoc algorithms to deal with issued related to that perspective, leaving the rest of dimensions aside [111]. To the best of our knowledge, there is not yet a design-time analysis technique that considers the BP control flow, data and resource perspectives jointly (cf. Chapter §4 for insights about BP analysis).

After designing the process and verifying the model, implementation is required. This is done during the *Configuration phase*. It can be done in different ways. The implementation of a new system may not be necessary when the process is implemented from a set of policies and procedures that the employees of the enterprise need to comply with. Otherwise, the software system where the process will be executed must be selected and configured in order to take into account the interactions of the employees with the system and the integration with existing software systems. This integration with existing systems may involve some implementation work, for instance to attach legacy systems to a BPMS. Finally, this configuration must be tested, where traditional testing techniques from the software engineering area can be applied, and deployed in its target environment.

Next stage is the *Enactment phase*, which encompasses the execution of the BP. On the one hand, a correct orchestration is necessary for the business activities to be per-

formed according to the BP execution constraints. On the other hand, process monitoring is crucial for providing information about the status of running BP instances. *Business Activity Monitoring (BAM)* techniques [55] are used for this purpose. They should provide functionality to warn about possible problems, send the appropriate alerts, and allow for some recovery protocol. Furthermore, during this phase, valuable execution data is gathered. Typically, execution logs are used to orderly storage information about processes such as the start or the end of activities.

Finally, the *Evaluation phase* uses information collected to assess and improve BP models and their implementations. Post-mortem analysis of BPs is required in this case. It may have four main goals: (i) to check whether the process instances worked as expected, and detect unexpected behaviour of the process under certain circumstances. This check is related to the degree of compliance of the process with the rules that must be fulfilled in the organisation; (ii) to find out bottlenecks or conflicting activities; (iii) to perform any kind of statistical analysis over one or more BP perspectives; and (iv) to identify changes that lead to an improvement of the process. Techniques from the fields of BP intelligence [67], and thus, process mining [130, 133, 134], data warehousing and classical data mining are usually applied in this phase.

Note that the order in which these phases need to be executed is not strictly fixed. Indeed, incremental and evolutionary approaches involving concurrent activities in multiple phases are common.

2.5 Business Process Management System

All the phases of the BP lifecycle, as well as all the BP perspectives can be supported by software systems known as *Business Process Management System (BPMS)*. They aim at providing an integrated set of tools to model, simulate, deploy, enact, monitor, evaluate and continuously optimize BPs. They coordinate tasks and synchronize data across existing systems. They also help coordinate human process activities, streamlining tasks, triggers, and timelines related to a BP, and to assure they are completed as defined by the BP. As stated by Andrikopoulos et al. [12], "a BPMS makes processes more efficient, compliant, agile, and visible by ensuring that every process step is explicitly defined, monitored over time, and optimized for maximum productivity". There are plenty of BPMSs existing in the market. Some are open-source, e.g.

YAWL [9], jBPM[4], and Activiti[5]. Others, such as *Architecture of Integrated Information Systems (ARIS)*[6], Intalio|BPMS[7], and AuraPortal[8], are commercial.

BPMSs are composed of a number of subsystems that address functions belonging to different phases of the BP lifecycle. Several roles performed by people in an organisation are involved in the operation of these subsystems. An overview of the typical architecture of a BPMS is depicted in Figure §2.4. In the following we provide definitions for every subsystem (inspired by the descriptions given by Weske [137]), specify the BP lifecycle phase to which each it is related, and explain the meaning of the roles involved (inspired by the definitions provided by Andrikopoulos et al. [12]). Among the roles, the *business managers* are the highest responsible for the BPs operated in an organisation. They must abstract BPs and rules from the underlying applications and infrastructure, and control the proper operation of the processes along all the phases of the BP lifecycle.

- *BP Modelling Tool.* The BP modelling tool is used for creating BP models, in which all the BP perspectives (cf. Section §2.3) involved in a process should be properly represented. As the *BP modelers* of an organisation are aware of the BPs contained in the company, they are typically in charge of modelling them. The BP models can be then used by process analysts for analysis purposes.

  Thus, after modelling, the role of the *BP analysts* comes on stage. They deal with the more tactical aspects of BPM (discovering, validating, documenting and communicating BP-related knowledge) and typically do process and data analysis, make changes to processes, and make sure that any ramifications downstream and upstream from the process have been checked over. BP analysts are involved in several phases of the BP lifecycle, mainly at the Design and Analysis, and the Evaluation phases.

  The *business managers* may also review the process models defined and propose changes in order to make them compliant with the business rules that regulate the company. Similarly, *BP architects* work to resolve the inevitable differences that crop up between the BP analysts and business units, which may affect the design and modelling of the BPs.

[4] http://www.jboss.org/jbpm/
[5] http://www.activiti.org
[6] http://www.softwareag.com/corporate/products/aris_platform/default.asp
[7] http://www.intalio.com/bpms
[8] http://www.auraportal.com/

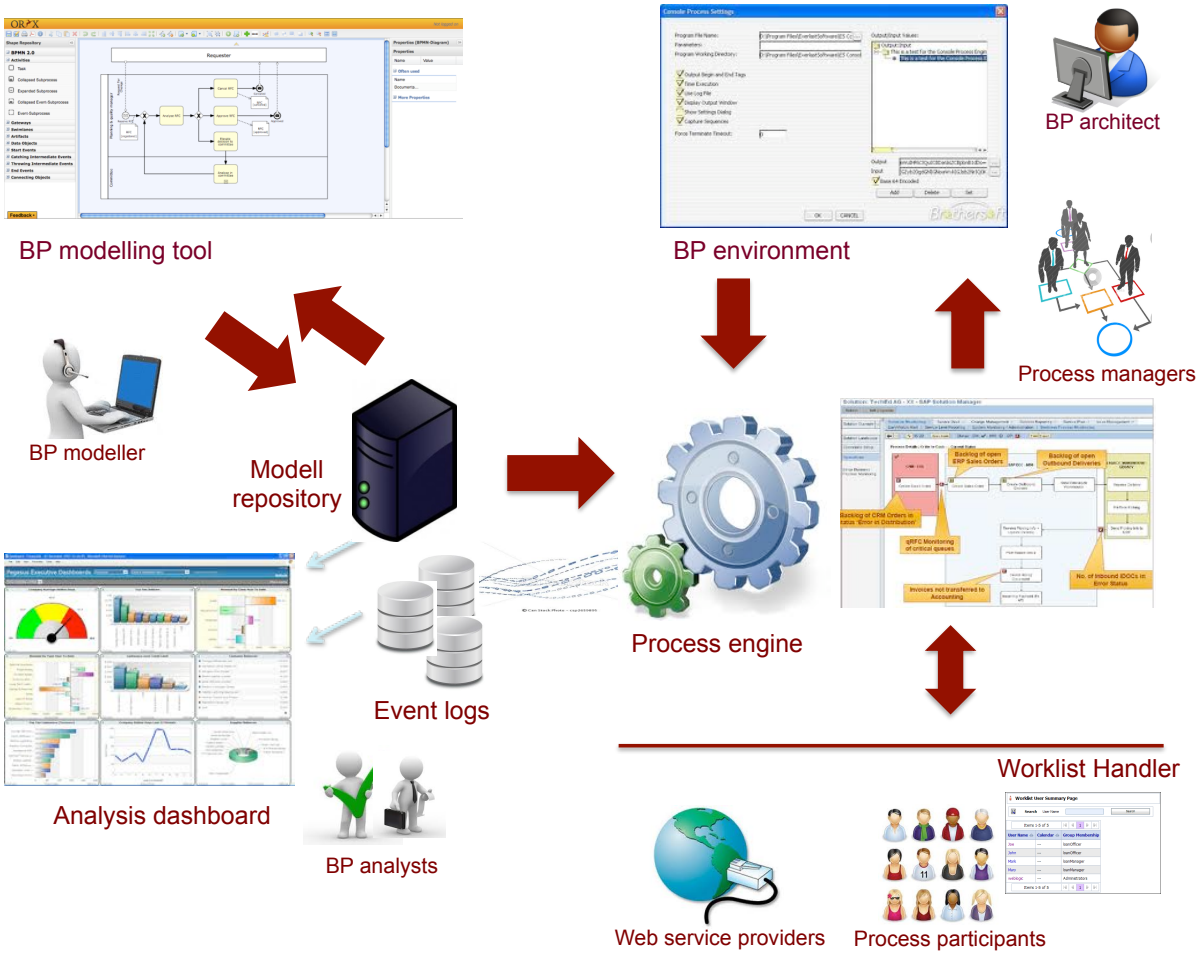


Figure 2.4: Subsystems and roles involved in a BPMS

- *BP Environment.* Once processes and system are prepared, the BP environment triggers the instantiation and enactment of BP instances based on the BP models. The *BP architects* may also participate in the preparation of the system during the Configuration phase of the BP lifecycle.

- *Process Engine.* The process engine is the core component of a BPMS. It is triggered by the BP environment and is responsible for instantiating and controlling the execution of BPs, i.e., the operation of a process at the Enactment phase of the BP lifecycle. To execute a particular activity instance, the process engine calls entities that act as providers of the required functionality. Typically, these providers are human resources of the organisation (i.e. people), or service providers (in service-oriented architectures). After deployment, the business managers review reports about the BPs and make suggestions for their refinement.

- *Worklist Handler.* In the case of having human participants, an activity typically generates one or more *work items* (a.k.a. *tasks*[9]) which together constitute the work to be undertaken by a user (or a group of users). The list of work items associated with a given participant (or with a group of people) constitutes his/her (their) worklist. The work item(s) are normally presented to the user via a *worklist*, which maintains details of the work items allocated to a user, and a *worklist handler*, which interacts with the work ist on the behalf of the user [43].

- *BP Model Repository.* The BP model repository mainly holds BP models that are created with the BP modelling tool. All the organisational and technical information that the process engine needs in order to determine the human resource in charge of an activity, or to access a service provider in case of calling a Web service that offers the required functionality, may also be stored in this repository.

- *BP Event Logs.* The event logs (a.k.a. *history logs*, or *execution logs*) are used to store the information generated during BP execution, which is necessary to perform run-time and post-mortem analysis.

- *Analysis dashboards.* Dashboards can be generated as a result of the analysis (specially design-time and post-mortem analysis) with the aim of studying the actual behaviour of a BP in order to identify and correct potential problems (at design time), or define improvement mechanisms to be applied in subsequent executions. The BP analysts are usually in charge of performing such evaluations.

2.6 SUMMARY

In this chapter we have introduced the main concepts related to BPM. In particular, we have provided definitions for BP and BPM, and we have shown a BP example accompanied with its BP model, with a brief introduction to BP modelling notations. We have also described the main BP perspectives usually involved in BPs, from which control flow, data, and resources stand out. Then, we have described the BP lifecycle, with the aim of understanding the phases through which a BP passes since its design until a BP instance is completed. Finally, a brief description of the features and human roles involved in a BPMS have been provided. For a more complete and rigorous introduction to BPM we refer the reader to Weske [137] and Andrikopoulos et al. [12].

[9] van der Aalst and Kumar [131] distinguish between task and work item, but this difference is not relevant in the context of this thesis.

3

Specification of Resources in Business Processes

"In the end, all business operations can be reduced to three words: people, product, and profits"

Lee Iacocca (1924–),
Businessman

In this chapter, we address one of the fundamental tasks in the management of the BP resource perspective: how to specify which resources are associated to the activities represented in a process model. In Section §3.1 we present the big picture of the application scenario and the elements involved, and introduce the content of the rest of the chapter. In Section §3.2, we describe the foundations to be considered when developing resource specification techniques in BPM, namely: (i) the organisational model that is used by the approach to assign resources to process activities; (ii) the task duties considered for resource assignment; (iii) the assignment patterns supported from a list of twelve patterns; and (iv) the binding strategy followed for resource specification and assignment. We have carried out a study of the existing proposals dealing with resource specification in BPs according to these foundations, whose results are summarized in Section §3.3. A more complete description of each approach can be found in Appendix §A. The main issues drawn from the chapter are outlined in Section §3.4.

3.1 INTRODUCTION

One of the main tasks in BP design in an organisation that wants to develop support for the management of the BP resources perspective, is the definition of which members of the organisation participate in each of the activities of its BPs. Figure §3.1 provides an overview of the elements involved in resource specification in BPs. There are three main elements in the picture:

(1) represents the control flow and the data flow of a BP modelled in a BP diagram. Definitions for BPs, BP models and the BP perspectives have already been provided in Chapter §2.

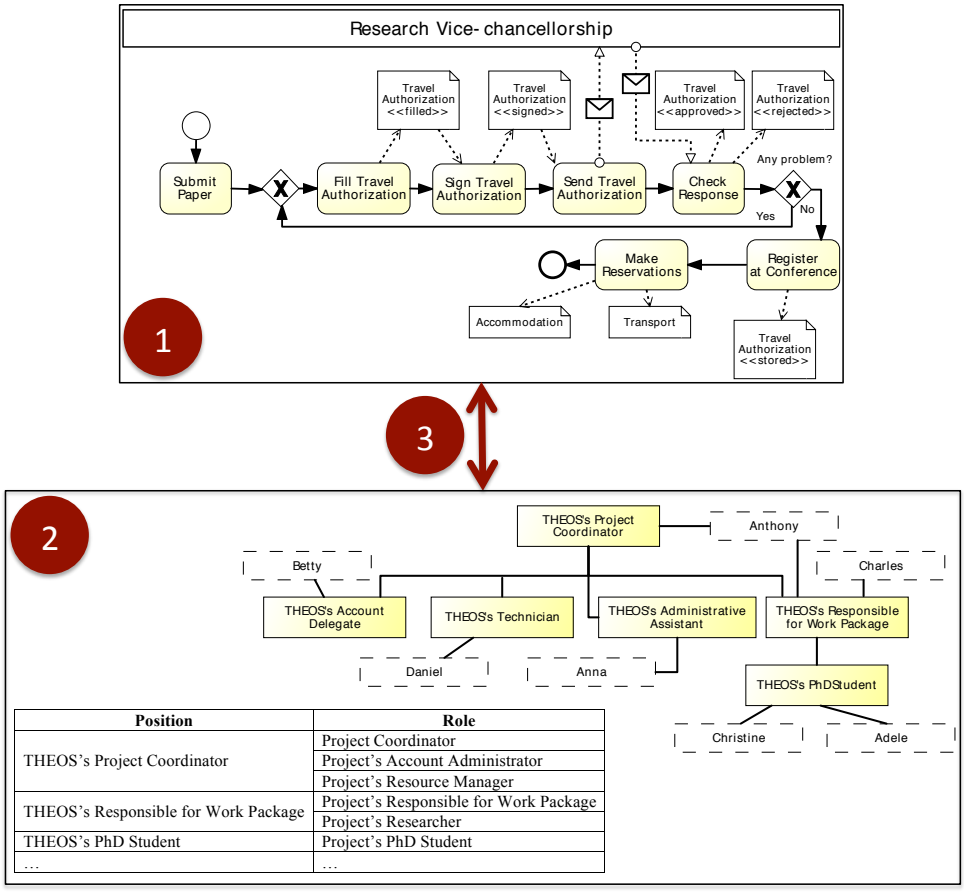
(2) depicts the *organisational model* of an organisation. There are plenty of proposals describing different structures (i.e. organisational metamodels) that can be adopted by an organisation, most of which share concepts such as *person*, *role*, and *group*. The main difference between them usually relies in *what* entities of the model can be hierarchically organised, and/or in the introduction of *new concepts*, e.g. position. Therefore, an organisation can choose *which type* of organisational model it wants to use to organise its employees, which usually depends on the characteristics and needs of the company.

(3) represents all the elements that need to be considered when dealing with the representation of the BP resource perspective, that is, the elements that must be taken into account to assign resources to BP activities. Specifically, we must: (i) select the *task duties* we are interested in; (ii) decide for which *assignment patterns* we are going to provide support; and (iii) choose the *binding strategy* to be used to represent the resource specification associated to the BP models.

In this chapter we focus on points two and three, so in next section we delve into organisational models, task duties, assignment patterns, and binding strategies.

3.2 FOUNDATIONS

In the following, we define the basic issues to consider when facing the specification of resources in BPs, according to which we will study the current approaches.



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

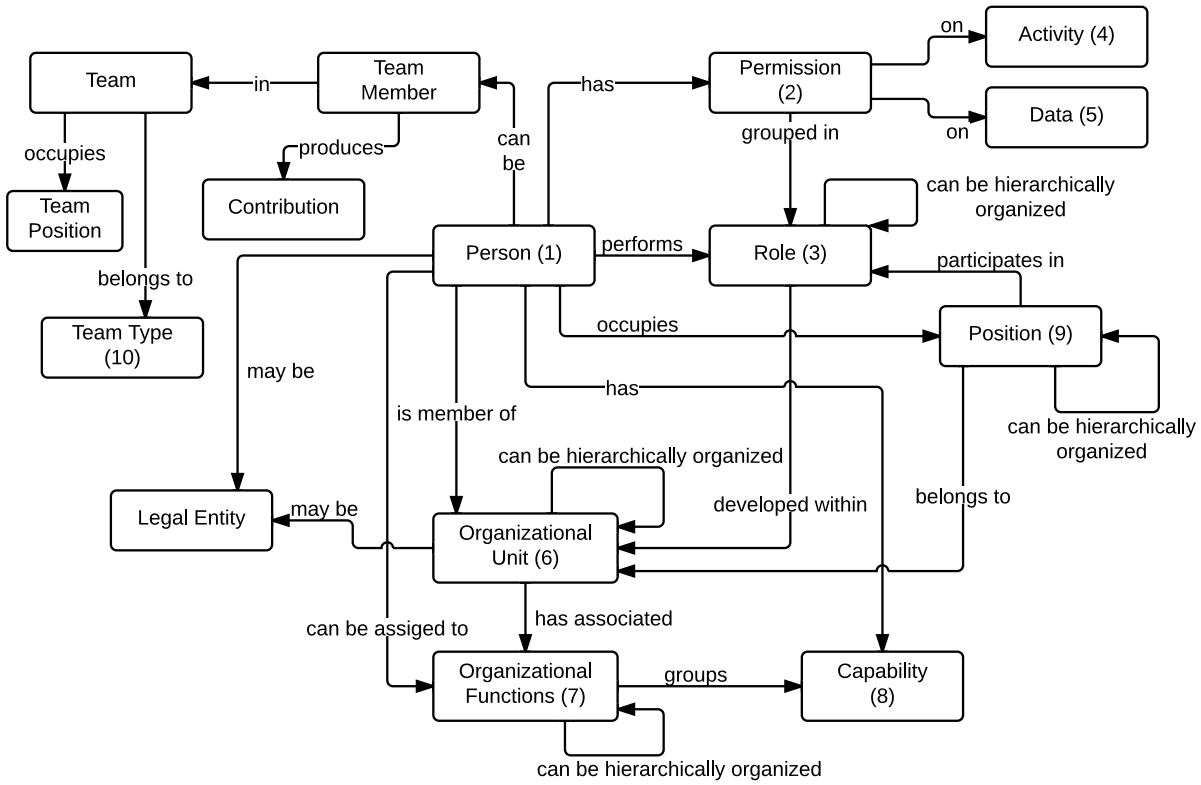
Figure 3.1: Elements involved in resource specification in BPs

3.2.1 Organizational Models

A plethora of metamodels to represent organisational structures have appeared in the last decade in different application areas, from Artificial Intelligence to Social Sciences. They differ from one another in the concepts used to structure the *organisation*. The conceptual map made up by putting together the elements of all the metamodels we have studied, is depicted in Figure §3.2, along with the synonyms found for each term. Let us define the concepts and their relations.

Person (a.k.a. *individual*, *user*, *resource* or *human resource*). It is typically an individual of an organisation to whom activities are allocated according to his/her **permissions** (a.k.a. *privileges*, *duties*), which define the type of **activities** (a.k.a. *tasks*) that he/she can undertake, and the type of **data** (a.k.a. *information*, *objects* or *data objects*) to which he/she has access[1].

[1]Notice that some approaches are object-oriented instead of task-oriented, which means they deal



Synonyms:
(1) Individual, user, resource, human resource.
(2) Privilege, duty.
(3) Organizational role, functional role.
(4) Task.
(5) Information, object, data object.
(6) Group, organizational group, organizational level, organizational role.
(7) Function, profile, person type, person class, resource type.
(8) Skill, qualification, competency, knowledge.
(9) Organizational position.
(10) Team role.

Figure 3.2: Conceptual map of organisational structures

Role (a.k.a. *organisational roles* or *functional roles*). Roles are assigned to a person according to his/her permissions in the organisation. In some proposals, roles are organised in a hierarchy, in which an inheritance relation is usually assumed, i.e. the person that plays a role implicitly has all the privileges/permissions of the roles below it in the hierarchy [11, 78, 132].

Organizational Unit (a.k.a. *group*, *unit*, *organisational group* [132], *organisational level* [41] or *organisational roles*[2] [17]). An organisational unit represents a set of per-

with the definition of permissions for access control to data rather than permissions related to activities.

[2]Please, notice that using *organisational role* to refer to groups is conflicting with term *role* itself. In those cases, providing an explicit definition of the term is convenient.

sons that have specific characteristics, e.g. they are associated to the same department or to a specific type of tasks [113, 126, 132]. They may belong to other organisational unit, and thus form hierarchies [106, 126].

Position (a.k.a. *organisational position*). It appears as a connection element between persons and the units and/or roles to which they are associated [78, 106, 113, 132], so that the organisational characteristics that resources possess relate to the position(s) that they occupy rather than directly to the resource themselves. In languages such as YAWL, people can be associated to roles and positions separately, but in other organisational metamodels, roles are associated to people only by means of positions. In addition, there may be a hierarchy of positions representing the possible lines-of-reporting in the company [9].

Legal Entity. When the permissions or functions associated to a person or organisational unit (respectively) are worldwide, they may be considered a *legal entity*. The rights and responsibilities of a legal entity are recognized in the world at large and by legal jurisdictions in particular, unlike persons and organisational units, which need only to have full recognition within an organisation [126].

Organizational Function (a.k.a. *functions*, *profile*, *person type*, *person class* or *resource type*). This concept refers to the characteristics shared by the members of a group, or just assigned to a specific person, as derived from the definition of organisational function provided by Casati et al. [40]. It groups similar capabilities in a single concept, which can make rise to hierarchies of organisational functions, specially useful in collaborative environments. Please, notice that term *role* is used in approaches such as ARIS to refer to organisational functions [113]. We consider that roles group permissions, whereas organisational functions group capabilities.

Capability (a.k.a. *skill*, *qualification* [78, 93], *competency*, or *knowledge* [78]). A capability is an attribute associated to a function entity or to a person. That is, an organisational functions groups a set of capabilities, which can be individually associated to the persons of the organisation. They may serve as filtering criteria to select and assign people to tasks in the BPs of an organisation. In most cases, this concept has a very broad meaning. For instance, when associated to a person, its scope ranges from information such as the age of a person, his/her gender, or his living town, to details about his/her technical knowledge on a specific field, or the number of years that he/she has occupied a certain position in a company.

Team. In collaborative environments, resources can also be part of hierarchically structured teams for solving collaborational tasks [86].

Team Type (a.k.a. *Team Role*). It can be seen as the role concept extended to teams.

Team Position. It can be seen as the position concept extended to teams.

Team Member. In collaborative environments, persons play the role of team members that produce concrete **contributions** for the completion of the activities assigned to their team(s) [131].

Figure §3.1 illustrates the use of most of the aforementioned concepts. This organisational model is based on a hierarchy of positions related to a hypothetical THEOS project, which correspond to the Project Coordinator, the Account Delegate, the Technician(s), the Administrative Assistant, the people Responsible for Work Package(s), and the PhD Student(s) involved in the project. These positions are occupied by different members of the organisation in question, and have one or more roles associated to them (summarized in the table attached to the hierarchy). Information about the organisational unit(s) in which this structure is used, or the capabilities of each person, could also be part of such an organisational model. They are not depicted in the figure for the sake of readability.

Please, notice that the terms described may have slightly different names in different approaches, and they can also be found in metamodels and ontologies defined in application domains different from BPM [6, 22, 39, 53, 59, 70, 84, 92, 116, 117, 123, 140]. A survey on how to model and simulate organisations have been performed by Nicolae and Wagner [92].

In this section, we have provided a common vocabulary that will be used throughout this thesis to refer to concepts related to organisational models. The synonyms of the terms defined above that are described in approaches that do not belong to the field of WF and BP management, have been left aside.

3.2.2 Task Duties

When speaking about resource assignment, most of the times we implicitly refer to specifying who must *undertake/perform* or *control* the execution of a task. However, there are other duties involved in the development of an activity. For example, there may be the need to *notify* somebody about different milestones related to the activity,

e.g. when it starts, when it finishes, or even when specific goals are achieved along the task execution. Furthermore, it is also common to require the *approval* of the work developed to complete an activity by somebody that is said to be accountable for the task, e.g. a supervisor; or to ask for *assistance* in the form of external help to perform some action required to complete the work. Some developers seem to be aware of this, as they are incorporating features to define some of these functions in the BPMSs. For instance, the so-called *Generic Human Roles* defined in the current version of WS-HumanTask [3] and BPEL4People [2], include support for some of the aforementioned duties. The *RACI roles* defined in the RACI matrices [44, 118] are exactly focused on indicating the performers of several task duties related to the execution of an activity. In addition, extensions of the original definition of RACI matrix (e.g. RASCI), define a greater variety of task duties. However, most of the current systems or approaches dealing with resource specification do not consider the existence of types of duties apart from the resource Responsible for an activity [17, 69, 78, 86, 120, 139].

In this thesis, we are going to take into consideration five different *task duties* that can be specified for a BP activity:

Responsible - R (a.k.a. *performer*, or *(task) executor*). Person who must perform the work, responsible for the activity until the work is finished and approved by an accountable. There must be at least (and there is typically) only one person being responsible for an activity of a BP.

Accountable - A (a.k.a. *approver* or *final approving authority*, *owner*, or *(task) stakeholder*). Person who must approve the work performed by the person responsible for an activity, and who becomes responsible for it after approval. There is typically one and only one resource accountable for each activity.

Consulted - C (a.k.a. *counsel*, or textitconsultant). This duty involves the people whose opinion is sought while performing the work, and with whom there is two-way communication.

Informed - I (a.k.a. *notification recipient* [3, 119]). Person who is kept up-to-date about the progress of an activity and/or the results of the work, and with whom there is just one-way communication. There may be more than one informed person for an activity.

Support - S. People who may assist in completing an activity, i.e., the person in charge of the task can delegate work to them. Unlike *Consulted*, who may provide input

information to the activity (i.e., information helpful to perform some work), *Support* will actively contribute in the completion of the activity.

Please, note that the term *performer* is typically used to refer implicitly to the task duty Responsible for an activity, since it is by default the task duty supported by the current approaches. However, in this thesis we extrapolate the concept to all the task duties, and thus we refer to the *performer(s) of a task duty that is associated to a BP activity*.

Under certain circumstances the person that performs a task duty for an activity may change dynamically, that is, when an activity is ready to be executed but has not yet been allocated to a specific person, or while the activity is being completed (cf. Figure §2.2 for an excerpt of the activity lifecycle). There are two well-known example of this situation:

Delegation: The ability to change the person Responsible for an activity dynamically.

Escalation: The ability to change the person Accountable for an activity dynamically.

In addition, there may be several resources performing the same task duty for a single activity of a process. For instance, there may be two persons notified of the completion of the work once the activity is over. Meyer deals with this situation [86]. Among the contributions of the work, he defines a set of patterns that have not been defined before in literature, which he calls *Advanced Resource Patterns (ARPs)*. Two of the ARPs are related to the specification of the number of resources that can be associated to a task duty in a process either for a specific activity or for the whole BP.

Number of Task Owners Specification - Single Resources: The ability to specify a minimum and maximum number of possible task owners. Only single resources are considered. For instance, between one and three people can provide Support to make the reservations necessary to prepare a trip to a conference.

Number of Task Owners Specification - Multiple Resources: The ability to specify a minimum and maximum number of possible task owners. Only complete organisational units or teams are considered. For instance, a maximum of two teams can be requested for information in order to complete the activity that deals with the filling of the authorization form. This means that people of at most two teams can work as Consultant for the activity.

Please, notice that he refers to the *task owner*, that is, the resource Accountable for the task. However, in this thesis we are going to assume that the following two ARPs can be applied to any task duty defined above.

3.2.3 Assignment Patterns

We need to use some resource assignment language to specify which resources of an organisation can perform each task duty for the activities of a BP. To this respect, it is important to distinguish between two concepts, related to states Allocating and Allocated of the activity lifecycle shown in Figure §2.2 (cf. Section §2.3):

- *Resource assignment* consists of indicating the conditions that the *potential performers* of a task duty must meet in order to be allowed to become *actual performers* of the task duty for the activity. Typically, at run time the task duty for each instance of the activity will be allocated to only one of the candidates. This might not hold for task duties Support, Consulted, and Informed, as there can usually be more than one performer of them associated to an activity.
- *Resource allocation*, thus, is performed when one specific resource from the set of potential performers of a task duty of an activity, is selected as *actual performer* of the task duty for an activity instance.

Although we usually use these terms in the scope of task duty Responsible, they could be applied to the rest of task duties defined in Section §3.2.2. In this section we specifically focus on defining resource assignments, since task allocation is a duty of the BPMS in which the process is executed and, thus, it is not related to resource specification. We refer the reader to Adams [8, 9] for further information about concepts related to resource assignment and allocation.

The types of resource assignments we are able to express depend on the language used. We have identified a set of *assignment patterns* for which the resource assignment languages may provide support, and which can serve as starting point to develop new languages for such purpose. Most of the patterns have been directly extracted from literature, although in some cases they have been renamed or their descriptions have been slightly refined in order to provide more accurate definitions.

Russell et al. [106] defined the well-known *Workflow Resource Patterns (WRPs)* "to capture the various ways in which resources are represented and utilized in WFs".

They emerged to extend the Workflow Patterns Initiative[3], which at that moment focused on issues related to control flow and data management. There are seven groups of WRPs (creation, push, pull, detour, auto-start, visibility and multiple resource patterns) containing a total of forty-three patterns. Using several WRPs together, we should be able to provide support for assignments such as "Activity *Deploy Application* must be undertaken by someone that reports work to the *Project Manager*, preferably the person that carried out activity *Supervise Code*'; or "Activity *Upload Updates* can be done by someone playing role *Technician* or having positions *Web designer* or *Web developer*".

The WRPs have been used in several approaches as a reference model to assess the resource management capabilities of different modelling languages, as well as to extend existing modelling notations [17, 69, 86, 139]. However, not all the WRPs are assignment patterns. Some of them are focused on indicating the order in which tasks can be executed, some refer to properties that the BPMS in which the process is run should have, and so on. The group of WRPs most closely related to the assignment patterns is constituted by the so-called *creation patterns*. In fact, most of the creation patterns have been included in the assignment patterns defined in this thesis.

As stated in Section §3.2.2, Meyer [86] introduced the so-called ARPs as an extension of the support provided by the WRPs. They mainly deal with cardinality, task completion and allocation, and teamwork management. Some of them have to do with resource assignment in BPs. In particular, two ARPs were used when we talked about the task duties in Section §3.2.2, and others are part of the assignment patterns described below.

Next, we describe the twelve assignment patterns that we are going to use as reference framework throughout this thesis when speaking about resource assignment to the task duties associated to a BP activity. Notice the use of term allocation instead of assignment in their names. The reason is that most of them are taken directly from the WRPs, and these are defined as features that a BPMS should have to properly deal with resource management. Therefore, they are mainly oriented to describe required runtime functionalities. However, we believe it is reasonable to assume that in order for a system to allocate tasks to resources based on a specific criteria, assignments based on that criteria need to be allowed. Besides, as aforementioned, other approaches have al-

[3]We refer to `www.workflowpatterns.com` for more information about the Workflow Patterns Initiative in general, and to `http://www.workflowpatterns.com/patterns/resource/` for details about the WRPs in particular.

ready used them for a similar purpose. We have extended the meaning of the patterns to deal with any task duty associated to an activity.

Direct Allocation: This pattern represents the ability to specify at design time the identity of the resource that will be perform a task duty for an activity. For instance, *Anna* is responsible for checking the response received from the *Research Vice-chancellorship*.

Role-Based Allocation: This pattern represents the ability to specify that a task duty can only be executed by resources with a given role. For example, a person with role *PhD Student* in the scope of project THEOS is in charge of submitting the Camera Ready version of an accepted paper to a conference.

Deferred Allocation: This pattern represents the ability to defer specifying the identity of the performer of a task duty until run time. For instance, during execution of the process, instances of the *Send Travel Authorization* task must be undertaken by the person referenced in field *Attendee* of data object *Travel Authorization*.

Separation of Duties (a.k.a. *Segregation of Duties*): This pattern represents the ability to specify that two task duties of two different activities must be allocated to different persons. For example, the travel authorization form must be signed by anybody except the person that filled in the document, in order to prevent conflicts of interests. This pattern belongs to the so-called *access-control constraints* (a.k.a. *entailment constraints*), which are used to specify constraints that take into account the resource assignments defined for two activities of a process. There is *Dynamic SoD (DSoD)* and *Static SoD (SSoD)*. The former applies only in one instance of a BP, i.e. the two activities affected by the constraint must be undertaken by different individuals in a single process instance. If the opposite is not mentioned, this is the default form of this assignment pattern. The latter refers to the whole BP and takes into consideration all the instances run for a process, e.g. the person Responsible for one activity has to be different than all the persons that were Responsible for a specific activity in *any* execution of the process.

Case Handling: This pattern represents the ability to allocate a specific task duty to the same resource for *all* the activity instances of a BP instance. For example, a single person with role *Project's PhD Student* is responsible for undertaking all the activities of the trip management process.

Binding of Duties: This pattern represents the ability to specify that two task duties of two different activities must be allocated to the same person. For instance, the

person that submits the paper is due to register at the conference. It also belongs to the access-control constraint and, thus, it can also be found in similar dynamic (*Dynamic BoD (DBoD)*) and static (*Static BoD (SBoD)*) forms as those described for the *Segregation of Duties (SoD)* pattern. The modality by default is the dynamic one, i.e. the constraint is applied within a single BP instance.

A variant of this pattern is the so-called *Retain Familiar* WRP, which represents the ability to *preferably* allocate a task duty of an activity to the same resource that perform a specific task duty in a preceding activity. That is, it favours the allocation of the task duty to the resource that performed it in a previous activity. For instance, any person playing role *Project's PhD Student* in project THEOS can become responsible for making the reservations required, even if he is not the one attending the conference. However, it is preferable to allocate this task to the individual interested in going to the venue.

Capability-based Allocation: This pattern represents the ability to offer or allocate instances of a task duty to resources based on their specific capabilities. For instance, instances of the *Sign Travel Authorization* task must be allocated to someone holding a degree.

History-based Allocation: This pattern represents the ability to offer or allocate activities to resources on the basis of their previous execution history. For instance, any participant in any instance of the process can check the answer received from the *Research Vice-chancellorship*. This example may make no sense in a real scenario, but it is used to exemplify the kind of assignments that this pattern covers.

Position-based Allocation: This pattern, originally so-called *Organizational Allocation*, represents the ability to offer or allocate instances of a task to resources based on their positions within the organisation and their relations with other resources. For instance, the travel authorization form must be filled in by somebody that reports to Anthony.

Commonality-based Allocation: This pattern represents the ability to allocate a task to a resource based on the characteristics it has in common with another resource. For instance, the *Check Response* task must be allocated to someone that plays exactly the same roles played by Daniel.

Group-based Allocation: This pattern, originally so-called *Single Entity*, represents the ability to consider persons, groups or teams as single resources for alloca-

tion to a task duty. For instance, the *Make Reservations* task can be performed by anybody belonging to the THEOS organisational unit.

Restricted Team Size: This pattern represents the ability to specify a minimum and/or maximum size of a group or team able to allocate a specific task duty. For instance, task *Make Reservations* can be performed by three clerks, at most.

We have left out of the assignment patterns WRPs such as Delegation or Escalation because, although they are somehow related to the assignment of resources to task duties of an activity (cf. Section §3.2.2), these patterns deal with the *ability* of a resource to delegate, escalate, *etcetera*, work to another resource, rather than with the specification of to whom the work can be delegated, to whom the work can be escalated, and so on.

3.2.4 Binding Strategies

As depicted in Figure §3.3, and as explained in Section §3.1, when we address resource specification the starting scenario is composed of the following parts: (1) a resource un-aware BP model, that is, a process model with no information about resources; (2) the organisational model of the company that constitutes the ground for resource allocation, and which is assumed to be accessible; and (3) a way (e.g. a language) to specify the resources that can perform each task duty for each activity of the BP. At this point, we have to decide *how* the resource specification is going to be bound to resource un-aware BP model, that is, the *binding model*. According to the binding model, there are three *binding strategies*, namely:

All-in-one binding. With the all-in-one binding strategy, the binding is performed in the BP model itself by modelling all the resource assignments together with the BP control flow and data perspectives, giving rise to a *resource-aware BP model*. The advantage of choosing this option is that we "just" need to maintain one type of model, and as all the information is modelled together. However, the readability of the resulting BP models may be questionable depending on the type and the size of the process we are working with. Most of the BPMSs and resource specification approaches opt for this binding strategy [2, 3, 69, 86, 94, 119, 139].

Split binding. With the split binding strategy, we create a resource-aware BP model that contains part of the resource specification together with the BP control flow

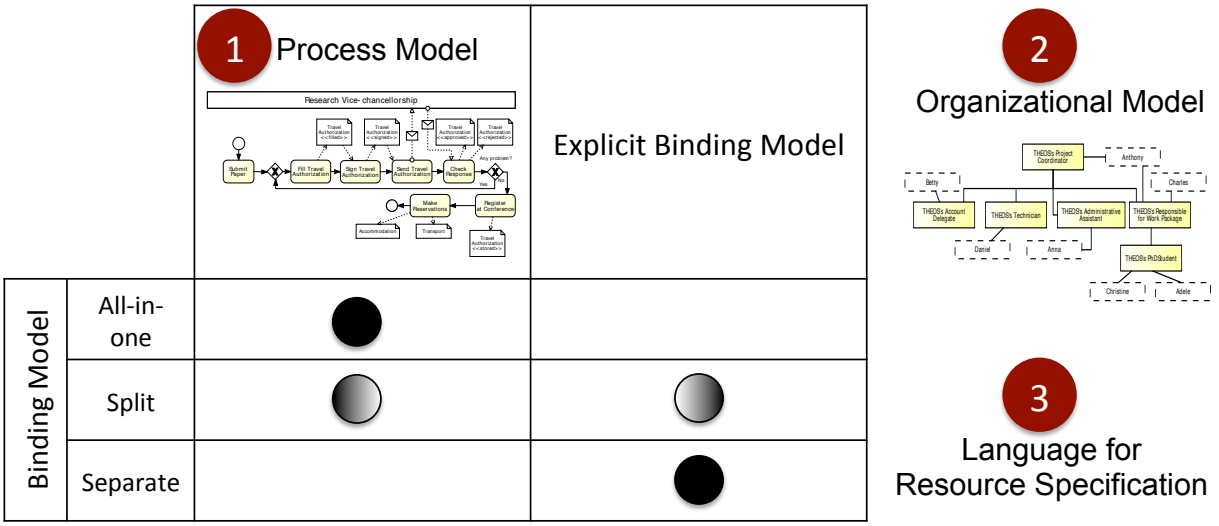


Figure 3.3: Binding strategies for resource specification

and data perspectives, and part of the resource specification is modelled separately, in a external model. For instance, the resource assignment expressions corresponding to task duty Responsible of the activities are included in the BP model, and the assignments for the rest of task duties are defined outside, e.g. in a text file. This helps keep the BP model cleaner, but it requires to keep track of the information represented with external techniques, and to develop a way to take everything into consideration when executing the BP. The Business Activities introduced by Strembeck and Mendling [120] are an example of split strategy.

Separate binding. With the separate binding strategy, the binding model is outside the BP model, that is, all the information related to the assignment of resources to the task duties of the activities is modelled separately. This, on the one hand, allows us to manage each element separately, while maintaining related information together and keeping the process model totally clean as for resource information. But, like in the previous case, mechanisms to jointly treat both "worlds" at run time are required in order to execute the process taking into consideration the human resources that participate in it. Some authors have adopted this binding strategy in their approaches [13, 17, 20, 56].

In any case, we reiterate the need of jointly *processing* BP models and resource models at run time for proper process execution, independently of the binding strategy used to represent BPs and resource specification. This is the only way to make processes aware of the resources involved in them while being executed, easing the auto-

mated allocation of resources to tasks. In addition, please note that the BP modelling notation used to model the BP control flow and data perspectives, can determine the binding strategy used for resource specification. For instance, BPMN [94] requires that all the information about binding resources be included in the BP model.

3.3 CURRENT PROPOSALS

This section outlines some proposals for resource specification found in literature, and extracts their main features with regard to the issues described in Section §3.2. We have mainly focused on the approaches that deal with the assignment of human resources to BP activities, leaving aside those that come from other research fields (e.g. agents, security). A total of twenty-one approaches have been studied.

Please note that, regarding the task duties, some approaches use the term *(potential) owners* for the assignment of resources to activities, without specifying what they mean by (potential) owner. Using the vocabulary we established in Section §3.2.2, in those cases we interpret that task duties Responsible and Accountable are supported, and we assume that the approach associated both duties to the same resource. Furthermore, as the Case Handling pattern can be seen as a generalization of the *Binding of Duties (BoD)* pattern to all the activities of a BP, we assume that the approaches that cover the latter, are provided with support to cover the former as well.

Role-Based Access Control (RBAC). *Role-Based Access Control (RBAC)* is a framework for controlling user access to resources based on roles. It can significantly reduce the cost of access control policy administration and is increasingly used in large organisations [61, 82]. This standard structures an organisation on the ground of organisational roles, which may be organised in a hierarchy in which permissions are inherited from the bottom up. Pattern Role-based Allocation is supported, as well as the specification of SoD.

RBAC-based approaches. Several proposals re-use (part of) the organisational structure of the RBAC model [4, 20, 120].

- The ***EXtensible Access Control Markup Language (XACML)*** profile offers an implementation of the RBAC model with role hierarchies and inheritance [4], according to the ANSI-RBAC [61], though different names are used to refer to the RBAC concepts. This profile enables the specification of the resource Responsible for an activity by using roles and capabilities, so it supports exactly two as-

signment patterns (Role-based Allocation and Capability-based Allocation). The binding strategy to be used is not specified.

- The ***Constraint Specification Language (CSL)*** introduced by Bertino et al. [20] considers an RBAC-like organisational model based on users and roles, where there is a dominance relationship between roles. It covers the assignment patterns dealing with persons, roles, access-control constraints and case handling, with which the resource Responsible for the tasks in a BP can be specified by using the separate binding strategy.

- The **Business Activities** constitute an integrated approach to enable the specification of process flows as well as process-related RBAC models and constraints [120]. It uses the RBAC principles to develop a language that enables the implementation of five assignment patterns (Role-based Allocation, SoD, BoD, Case Handling, History-based Allocation). The approach has been implemented as an extension of the UML 2 Activity Models, using a split binding strategy in which the assignment of tasks to roles is performed in an external model together with the whole configuration of the company, and only the access-control constraints are indicated in the activities of the BP model. Only the Responsible task duty is mentioned in the approach.

Workflow on Intelligent Distributed database Environment (WIDE). The WIDE *Workflow Management System (WfMS)* [40] uses an all-in-one binding strategy in which the resources Responsible for the BP activities are assigned to the activities by means of roles (i.e. Role-Based Distribution). Four more assignment patterns are supported by the system (SoD, BoD, Case Handling, History-based Allocation). The Support task duty is mentioned, although it is not clearly stated to which extend it is supported. The organisational model used for resource specification consists of different types of *agents*[4] that can be organised in hierarchies.

Extended WIDE. Casati et al. [41] present an advanced role-based authorization model for WF processes that uses the same organisational concepts handled in the WIDE WfMS [40] but with some differences introduced to deal with some authorization constraints. The five assignment patterns supported by WIDE as well as the Direct and Group-based patterns, are covered. The approach has been implemented by extending WIDE. The same task duties as in WIDE are considered, and also an all-in-one binding strategy is used.

[4]For Casati et al. [40], an agent can be a person, a role, a group, or an organisational function.

Russell et al.'s organisational metamodel. The work by Russell et al. [105] does not deal with resource assignment in BPs. It introduces an organisational metamodel that has been used as reference point to define the WRPs, as well as to develop languages such as YAWL [132], which makes this work appealing for our study, despite having a different scope than the other proposals. The excerpt of the metamodel related to human resources mainly works with persons, capabilities, a hierarchy of positions, and role and organisational unit levels.

YAWL. YAWL is a WF modelling language developed to extend the Petri Nets with dedicated constructs to deal with the WF patterns[5] [9, 132]. Today, it covers the BP control flow, data and resource perspectives. Regarding resources, it supports eight assignment patterns described in Section §3.2.3 (Direct, Role-based, Capability-based, Position-based, Group-based, and Deferred Allocation, SoD, Case Handling) on the ground of an organisational model very similar to the one presented by Russell et al. [105]. The Retain Familiar sub-pattern is also covered. It implements solely the Responsible task duty by means of an all-in-one strategy.

BPMN 1.x extensions. Several extensions for the varied BPMN 1.x versions released, all following an all-in-one binding strategy, have been proposed [17, 69, 86, 139]. The common aim is to improve the almost-missing support for resource management provided by the standard.

- Grosskopf [69] proposed the introduction of new attributes and associations to the metamodel used by BPMN 1.0, mainly persons and roles, in order to capture missing WRPs. Six assignment patterns (Direct, Role-based and Deferred Allocation, SoD, BoD, Case Handling) plus the Retain Familiar sub-pattern are covered by the approach, in which the Responsible and Accountable task duties are mentioned.
- BPMN 1.0 was also extended in a different way by Wolter and Schaad [139], this time by defining a new organisational metamodel for BPMN, which is similar to hierarchical RBAC (cf. Section §A.1 for information about the RBAC variants). Five assignment patterns are addressed by this approach (Role-based Allocation, SoD, BoD, Case Handling, History-based Allocation), which deals with the modelling of the resources that are Responsible for the BP activities.
- Meyer introduced the ARPs [86] and proposed an extension of the BPMN 1.1 metamodel for handling the concepts of resource, team, profile, role and per-

[5] Workflow Patterns: www.workflowpatterns.com

mission. The implementation of the approach supports six assignment patterns (Direct, Role-based and Group-based Allocation, SoD, BoD, Case Handling) for the definition of the resources that are Responsible for the BP activities.

- Some preliminary results of Meyer's work were published Awad et al. in [17], where they presented a simplified organisational metamodel that covers eight assignment patterns (Direct, Role-based, Capability-based, History-based and Group-based Allocation, SoD, BoD, Case Handling) plus Retain Familiar, and resources can be assigned to task duty Responsible.

BPMN 2.0. This version of BPMN does not work with a specific organisational model [94], so any structure is assumed to be valid. Resource assignment is based on literals that can be expressed with XPath[6] (by default) or with any other language. The default approach supports five assignment patterns (Direct, Role-based, Capability-based, Position-based, and Group-based Allocation) following an all-in-one binding strategy that considers the two task duties Responsible and Accountable, which are assumed to be performed by the same person.

BPEL4People/WS-HumanTask. BPEL4People [2] is an extension of the BPEL notation [1] based on the WS-HumanTask specification [3], which enables the integration of human beings in service-oriented applications. It does not use a pre-defined type of organisational model, nor deals with resource allocation at run time. Resources can be assigned to the BP activities directly or by means of roles, positions, and capabilities. Therefore, four assignment patterns are supported. Task duties Responsible, Accountable, and Informed are supported with the so-called *Generic Human Roles* by using an all-in-one strategy.

ARIS. ARIS [113, 119] is a tool suite that provides support for the management of several BP perspectives. It uses an organisational model mainly composed of organisational units, positions and roles, in which users are considered a special type of organisational unit composed of only one member. It covers a total of seven assignment patterns (Direct, Position-based, Role-based and Group-based Allocation, SoD, BoD and Case Handling). The Responsible, Accountable, and Informed task duties can be represented by means of an all-in-one binding strategy.

Enterprise Ontology (EO). EO [126] includes a great variety of carefully defined terms which are widely used for describing enterprises in general. The organisational metamodel implemented is focused on a hierarchy of organisational units, and on legal

[6] http://www.w3.org/TR/xpath/

entities that range from persons to organisations. The Role-Based Allocation pattern is somehow supported, although EO assumes a different meaning for the term role. We have identified only three assignment patterns supported by the approach (Direct, Role-Based, and Group-based Allocation), which allows assigning the resources that are Responsible and/or Accountable for activities. Being EO an ontology, the binding strategy to be used for resource specification is not a concern of the approach.

Du et al.'s Resource Management System. Du et al. [56] introduce the fundamentals for a resource management system addressing distributed resource management. They present a three-level hierarchy of resource managers, each of which must have an organisational model based on resources, resource types, roles and other attributes. They define three assignment languages that cover four assignment patterns (Direct, Role-Based, Capability-Based, and Position-based Allocation) for the specification of the resource Responsible for the BP activities. Although it is not explicitly indicated, the assignments seem to be done separately from the BP model.

Team-Enabled WF Reference Model. van der Aalst and Kumar [131] present a metamodel that introduces concepts related to team and teamwork. Among others, the BP activities can be assigned to work teams, one or more of whose members are responsible for the tasks. Eight assignment patterns (Role-Based, Position-based, Group-based, and Capability-based, SoD, BoD, Case Handling, and Restricted Team Size) are supported by the approach. Although the authors state that they focus on the performers of the activities and do not deal with duties such as responsible or accountable, following the vocabulary established in Section §3.2.2 *their* performers correspond to task duty Responsible. The binding strategy to use is not specified in the approach.

Tan et al.'s Constrained WF System. Tan et al. [124] use a simple organisational model, composed of persons distributed into role hierarchies, with which one can assign roles to the BP activities, and specify access-control constraints and cardinality constraints. Regarding cardinality constraints, the authors distinguish between local cardinality constraints, and global cardinality constrains. The former are used to specify SoD and BoD between process activities. The latter provides support for the Restricted Team Size pattern by allowing to limit the number of roles that can execute a set of activities of the BP. Five assignment patterns are covered in total (Role-based Allocation, SoD, BoD, Case Handling, and Restricted Team Size), dealing with the resources that are Responsible for the process activities. No binding strategy is specified.

Human Resource Meta-Model (HRMM). The HRMM [78] is a part of the *Resource Modeling Language (RML)* [93] that uses the concepts of skill, competency and knowl-

edge within the organisational model. There is no evidence of the consideration of task duties different from the resource Responsible for a process task in the approach. Only the Role-based Allocation pattern seems to be covered. How to do the binding is not specified because it is out of the scope of the work.

RASCI. Like Russell et al.'s metamodel [105], RACI [44] is another exception in our study because its aim is limited. In this case, it is focused on the definition of a variety of task duties for the activities carried out in an organisation, which are usually assigned to people by means of their roles. It uses a separate binding strategy, as resource assignments are represented by means of a specific type of RACI matrices called RASCI. Task duties Responsible, Accountable, Support, Consulted, and Informed, are inherent in RASCI matrices.

Tables §3.1 and §3.2 collect the information of the proposals assessed with regard to the foundations described in Section §3.2. In particular, the former shows the concepts that are considered in the organisational metamodels used in the proposals, according to the vocabulary we defined in Section §3.2.1. We use symbol ✓if the feature is supported. (h) indicates that the concept can be structured in a hierarchy. Furthermore, an asterisk $(*)$ is used when the meaning given to an element differs a little bit from the standard meaning given by the rest of approaches, e.g. in the case of hierarchies, maybe the entities have a domination relation, or a sub-class relation that is similar to a typical hierarchy but is not exactly defined like so by the authors. Regarding Table §3.2, it presents the support for the rest of issues described in Section §3.2, that is: (i) the task duties considered in the approach, of which the acronyms are shown according to Section §3.2.2); (ii) the assignment patterns covered, where *D* stands for Direct Allocation, *R* stands for Role-based Allocation, *G* stands for Group-based Allocation, and *P* stands for Position-based Allocation (cf. Section §3.2.3); and (iii) the binding strategy used, if specified. In those cases in which the information related to some issue studied is not included, and there are no clear examples of use provided that could serve as evidence of how they are treated in the approach, *N/S* (*non-specified*) shows up in the corresponding cell of the table. The justification of the values in the tables can be found in Appendix §A, where we present a broader description of each proposal and the results of the study.

As depicted in the Table §3.1, there is not an organisational metamodel par excellence, used by most of the approaches. Instead, each approach proposes one according to its needs and the final goal pursued. The concepts handled by a large part of the solutions are Person, Role, Organizational Unit and, to a lesser extent, Position

[4, 9, 17, 20, 40, 41, 56, 78, 82, 86, 105, 113, 120, 124, 126, 131, 139]. Roles, organisational units and positions are sometimes hierarchically structured according to some criteria. The addition of capabilities as well as the consideration of team work, are more and more present in the approaches [9, 56, 78, 86, 105, 131].

Regarding Table §3.2, most of the proposals only consider the management of the Responsible task duty, which is usually referred to as the *performer* or the *executor* of the process activities [4, 9, 17, 20, 41, 56, 86, 120, 124, 131, 139]. The approaches that deal with other task duties, generally do not provide full support for the corresponding functions, which are usually offered as optional. RASCI is an exception to this group.

Taking into consideration both tables, we can notice that the assignment patterns supported by an approach mainly depend on the organisational model used. Thus, if the model contains roles, the approach is likely to provide Role-Based Distribution; if there are capabilities, the Capability-Based Allocation is usually supported, *etcetera*. The most common assignment method is based on organisational roles, which is supported by all the approaches, indeed. The access-control constraints (i.e. patterns SoD and BoD) and, thus, the Case Handling pattern, count on quite a lot of support as well.

Finally, as depicted in Table §3.2, the most used binding strategy is the all-in-one binding [2, 3, 9, 40, 41, 69, 86, 94, 113, 139]. It means that the approaches tend to include all the information related to resources in the BP models along with the rest of BP perspectives, with the pros and cons that it entails (cf. Section §3.2.4 for details about the binding strategies). However, the binding strategy is not often a concept explicitly mentioned in the approaches. In those cases in which the binding strategy used was not specified, we have used the examples provided by the authors in the description of the proposal to classify the approaches as for this issue [17, 20, 56, 120]. In some of the approaches whose strategy has been marked as N/S in the table, we believe that any strategy could be valid.

3.4 SUMMARY

In this chapter, we have introduced the foundations to be considered in the specification of resources in BPs, and we have evaluated the existing proposals dealing with resource specification with respect to four aspects identified, to be named: (i) the organisational model that is used by the approach to assign resources to process activities; (ii) the task duties considered for resource assignment; (iii) the assignment patterns

supported from a list of twelve patterns; and (iv) the binding strategy followed for resource specification and assignment. The result of this study will be used to understand the existing weak points or problems in resource specification in BPs (cf. Section §5.2), and they will thus constitute goals to be achieved with the solutions proposed in this doctoral thesis.

Approach	Organizational Model				
	Person	**Role**	**Org. Unit**	**Position**	**Others**
RBAC [61, 82]	✓	✓(h)			Data, Permission
XACML [4]	✓	✓(h)			Data, Activity, Permission
Bertino et al.'s CSL [20]	✓	✓(h*)			
Business Activities [120]	✓	✓(h)			Data, Permission
WIDE [40]	✓(h)	✓(h)	✓(h)		Organizational Function (h)
Extended WIDE [41]	✓(h)	✓(h)	✓(h)		Organizational Function (h)
[105]'s organisational metamodel [105]	✓	✓(h*)	✓(h*)	✓(h)	Capability
YAWL [9, 132]	✓	✓(h*)	✓	✓(h)	Capability
BPMN 1.0 Ext. by Grosskopf [69]	✓	✓(h)			
BPMN 1.0 Ext. by Wolter and Schaad [139]	✓	✓(h)			
BPMN Ext. 1.1 by Meyer [86]	✓	✓(h)			Team (h), Organizational Function (h), Permission
BPMN 1.1 Ext. by Awad et al. [17]	✓	✓(h)	✓		Organizational Function
BPMN 2.0 [94]					
BPEL4People [2] / WS-HumanTask [3]					
ARIS [113, 119]	✓	✓	✓(h)	✓	Capability, Organizational Function
EO [126]	✓		✓(h)		Legal Entity
Du et al.'s Resource Management System [56]	✓	✓			Organizational Function (h), Capability, Non-capability
Team-Enabled WF Reference Model [131]	✓(h)	✓(h)	✓(h)	✓(h)	Capability, Team, Team Type, Team Position, and Contribution
Tan et al.'s Constrained WF System [124]	✓	✓(h)			
HRMM [78]	✓	✓	✓(h)	✓	Permission, Capability
RASCI [44]					

Table 3.1: Organizational metamodels of the current specification approaches

Approach	Task Duties	Assignment Patterns					Binding Strategy
		D	R	G	P	Others	
RBAC [61, 82]	N/S		✓			SoD	N/S
XACML [4]	R		✓			Capability-based	N/S
Bertino et al.'s CSL [20]	R	✓	✓			SSoD and DSoD, SBoD and DBoD, Case Handling	Separate
Business Activities [120]	R		✓			SoD, BoD, Case Handling, History-Based	Split
WIDE [40]	R-S		✓			SoD, BoD, Case Handling, History-based	All-in-one
Extended WIDE [41]	R-S	✓	✓	✓		SoD, BoD, Case Handling, History-Based	All-in-one
Russell et al.'s org. metamodel [105]	N/S						N/S
YAWL [9, 132]	R	✓	✓	✓	✓	Capability-Based, Deferred, SoD, Case Handling	All-in-one
BPMN 1.0 Ext. by Grosskopf [69]	R-S	✓	✓			Deferred, SoD, BoD, Case Handling	All-in-one
BPMN 1.0 Ext. by Wolter and Schaad [139]	R		✓			SoD, BoD, Case Handling, History-Based	All-in-one
BPMN Ext. 1.1 by Meyer [86]	R	✓	✓	✓		SoD, BoD, Case Handling	All-in-one
BPMN 1.1 Ext. by Awad et al. [17]	R	✓	✓	✓		SoD, BoD, Case Handling, Capability-based, History-based	Separate
BPMN 2.0 [94]	R-A	✓	✓	✓	✓	Capability-based	All-in-one
BPEL4People [2] / WS-HumanTask [3]	R-A-I	✓	✓		✓	Capability-based	All-in-one
ARIS [113, 119]	R-A-I	✓	✓	✓	✓	Case Handling, BoD, SoD	All-in-one
EO [126]	R-A	✓	✓	✓			
Du et al.'s Management System [56]	R	✓	✓		✓	Capability-Based	Separate
Team-Enabled WF Reference Model [131]	R		✓	✓	✓	SoD, BoD, Case Handling, Capability-based, Restricted Team Size	N/S
Tan et al.'s WF System [124]	R		✓			SoD, BoD, Case Handling, Restricted Team Size	N/S
HRMM [78]	R		✓				N/S
RASCI [44]	R-A-S-C-I	✓	✓		✓		Separate

Table 3.2: Resource specification in current approaches

4

Analysis of Resources in Business Processes

"The superior man understands what is right; the inferior man understands what will sell"

Confucius (551 b.C–479 b.C),
Philosopher

The analysis of the BP resource perspective provides information about how resources are being managed in a process-oriented organisation. In this chapter we describe how the analysis of resources in BPs is dealt with in the literature. After a brief introduction to the topic in Section §4.1, in Section §4.2 of this chapter, we introduce the foundations of resource-related BP analysis, that is, the issues to take into consideration when developing a method for analysing the resource perspective in BPs. Next, we study the related approaches, assessing them on the basis of the foundations identified. The results of the evaluation are presented in Section §4.3. Finally, a summary of the content closes the chapter in Section §4.4.

4.1 INTRODUCTION

BP analysis helps ensure the proper operation of BPs. As far as resources are concerned, the resource assignments associated to the process activities must be analysed in order to ensure that the members of the organisations are managed properly, and they are allocated to appropriate tasks according to their skills and duties.

Automating this analysis may bring many benefits. At design time, it allows investigating properties related to resource specification, and thus detect potential allocation problems that can be overcome beforehand by modifying the resource specification. At run time, it enables the automated allocation of activities to the members of an organisation, among other things. Post-mortem analysis of BPs in general helps improve subsequent executions of the BPs by identifying shortcomings and bottlenecks that can be solved by modifying the process models and/or the configuration of the BPMS. Therefore, the automated analysis of the BP resource perspective can be defined as the computer-aided extraction of information from BP models and/or other models that contain information related to the resources associated to the BP activities.

However, automatically analysing BP models implies taking into account several issues. Regarding the resource perspective, there are four main questions to explore:

(1) It is often useful to define the analysis of models in terms of analysis operations that take a set of parameters as input, and return a result as output. It has been successfully done in other fields such as feature models [19]. Thus, we first must come up with the set of *analysis operations* that we aim at supporting.

(2) Then, we must identify which other *BP perspectives* apart from the resource perspective are going to be affected by the analysis, which may depend on several issues, e.g. the kind of resource assignments we are dealing with.

(3) Another issue to consider is *when* the analysis is going to be performed, that is, in which *phase of the BP lifecycle* we plan to execute the analysis operations.

(4) Finally, we need to select or develop a *technique or formalism* to enable the automation of the analysis.

Notice that the foundation for resource specification that we described in Chapter §3 and the aforementioned four issues for BP analysis, are relatively independent from each other. Nevertheless, there may be relations between different issues. For instance,

there are analysis operations that force us to take into account several BP perspectives in the analysis, e.g. in order to find out who can potentially read a data object in a specific state, the control flow and the data perspectives must be considered in combination with resources; other times the involvement of one or other BP perspective directly depends on the resource assignment pattern(s) used for resource specification, e.g. if the language allows specifying access-control constraints, the control flow is likely to be involved in the analysis; the BP lifecycle phase(s) also influences the analysis, e.g. at the Enactment phase, the activities that have already been executed at a certain moment during BP execution and cannot be executed again in the process instance, are likely to be disregarded for the execution of an analysis operation; the same applies to the analysis technique, e.g. mechanisms such as Petri Nets may be necessary in order to execute analysis operations that involve the BP control flow perspective. To sum up, the decisions made with regard to some issues may constrain the available options for other elements. In next section we explain each issue related to resource analysis in BPs in detail.

4.2 FOUNDATIONS

In the following, we define the basic issues to consider when facing the analysis of resources in BPs, according to which we will compare the current approaches.

4.2.1 Analysis Operations

As stated in Section §9.1, analysis can be defined in terms of analysis operations that take a set of parameters as input, and return a result as output. We next describe the two analysis operations we have found in literature from a study we have conducted on the current approaches dealing with resource-related BP analysis. Please notice that we use the concept *solve a resource assignment expression* to refer to the calculation of the resources of the organisation that meet the requirements established in an assignment expression. For instance, solving an expression that states that the person Accountable for activity *Sign Travel Authorization* must play role *Manager* in the company, means finding out which resources of the organisation are associated to role Manager. Also note that the analysis operations can be applied to any $TaskDuty = [R, A, S, C, I]$ (cf. Section §3.2.2) associated to a BP activity, and are assumed to be related to a specific *BusinessProcess*.

Potential Performers (PP). This operation calculates the resources that are candidate to perform a specific task duty for a BP activity, e.g. the persons that can potentially become responsible for an activity. It takes a specific activity and a specific task duty, and it returns the set of persons of the organisation that meet the requirements set in the assignment of the task duty for the activity specified.

$$PP(Activity, TaskDuty) : Person[*]$$

This computation is usually done by *solving* the resource assignments associated to the activity for this task duty. Notice that when mentioning potential performers in general, we might refer to the potential performers of task duty Responsible, as it is the most straightforward and most considered duty in the approaches related to resource specification (cf. Section §3.3). Besides, it is of special concern for the BPMSs, as they may need to automatically solve the resource assignments during process execution [9, 40, 56, 113, 132].

Consistency Checking (CC). An activity is consistent with regard a task duty if there is at least one person that is allowed to perform the task duty for the activity, according to the assignment expression associated to the task duty. This operation takes an activity and a task duty, and returns whether the activity is consistent with regard to such task duty. A BP model is consistent if all its activities are consistent.

$$CC(Activity, TaskDuty) : Boolean$$

This definition is strongly influenced by the definition of consistency provided by Bertino et al. in [20], although we have extended it to deal with task duties. Thus, an inconsistent process may derive in deadlocks at run time, since there might not be anybody to whom some task duty can be allocated in case the activity needs to be executed in a BP instance.

4.2.2 Business Process Perspectives Involved in the Analysis

Several BP perspectives are involved in the analysis of resources in BPs. For some checks, it is required only information coming from the resource assignments of the BP activities and from the organisational model of the company. In this case resources are said to be analysed *in isolation*. For instance, in order to calculate the set of potential

performers for an activity, it is necessary to check which members of the organisation meet the conditions/requirements that enable them to undertake the task, e.g. who occupies a certain position, or has specific skills. For most of the assignments, this can be done by examining the organisational model of the company, comparing the properties associated to each person to those specified in the resource assignment.

However, in approaches that support the Deffered Allocation pattern (cf. Section §3.2.3), the resource assignment language used may consider the specification of assignments in which *run-time* information coming in data objects is required. In these cases, the *BP data perspective* must also be taken into consideration when calculating the potential performers of an activity.

The data perspective can also be involved in some analysis operations. For example, in order to give answer to questions such as "who can write data Authorization Request in the BP?", or "is there any person that can read all the data objects in all their states in a single process instance?", it is required to study the resource assignments of the activities in combination with the data they handle, as well as with the control flow perspective of the process. The need of jointly managing control flow, data and resources has been described by several authors in the scope of BPM [79, 85, 101]. Mendling et al. [85] introduce *Integrated EPC (iEPC)*, a BP modelling language that extends EPCs with a concept of object flow and role assignment. The aim is to address an existing gap in process verification, where most of the approaches only consider the control flow of the BP in the checks. However, to the best of our knowledge, there has not been proposed any implementation for the resolution of analysis operations dealing with resources, data and control flow, yet. Therefore, we did not include their definition in Section §4.2.1.

Notice that the control flow of the process may also need to be taken into account in BPs that contain exclusive decision points (i.e. XOR gateways) and use a resource assignment language that allows expressing dynamic access-control constraints (i.e. patterns DSoD, DBoD), where it is necessary to make sure that two activities that are constrained to each other do not belong to different execution paths (i.e. different branches of an XOR gateway). Otherwise, information necessary to solve the resource assignment expressions of the activities would be missing, as the activity referenced in the assignment is never executed in the same BP instance. This may cause a deadlock in the process at run time, if proper recovery protocols that implement a resolution mechanism for this problem, are not available.

Summing up, depending on several issues, e.g. the type of resource assignment

expression, the operation to be executed, or the phase of the BP lifecycle in which the operation is going to be performed, different BP perspectives are involved in the resolution of the analysis operations.

4.2.3 Business Process Lifecycle Phase Considered in the Analysis

Another issue to take into consideration in resource-related analysis is *when* we are carrying out the analysis of the process, that is, at which phase of the BP lifecycle (cf. Section §2.4). Furthermore, the BP perspectives involved in the resolution of some operations may vary depending on when the operation is performed. As detailed below, in each phase of the BP lifecycle, the analysis has some implications, pros and cons.

Resource analysis in the Design and Analysis phase. Analyzing the BP before running it, helps to check that the process has been designed according to the requirements established, and that it has been modelled properly. In design-time analysis, the context of the analysis operations is all the possible executions of the BP. Therefore, a person is a potential responsible of an activity if there is some process instance in which he or she can be the responsible of such activity. This means that no specific run-time information is available in this type of analysis. Furthermore, at design time we do not usually have critical temporal restrictions to perform the analysis operations, so we can use not-so-efficient techniques and/or more time-consuming algorithms, which may not be appropriate in other phases.

Regarding resource analysis, at design time we can check consistency in order to make sure that there will be at least one resource to which we will be able to allocate each task at run time. However, sometimes the analysis cannot be exhaustive due to the absence of run-time information that can affect the actual resource allocation. For instance in some BPs, although all the activities apparently have potential performers, the fact of allocating a task to a *certain* person at run time makes the set of potential performers of another activity somehow constrained to it, empty for that resource distribution. Two possible measures can be adopted to detect such situations: (i) carry out an exhaustive study of the technique being used for resource specification and analysis, so that we can detect potential problems and try and reduce the risk beforehand; or (ii) use process simulation techniques that allow taking into consideration the resource perspective (and those affected by the resource assignments of the activities) too [95].

Other analysis operations that will be described in Section §5.3 help find out people that never participate in a BP, or identify indispensable resources without which a BP could not be completed. Working at design time, every potential problem detected can be overcome before launching the process.

Resource analysis in the Enactment phase. By analysing the BP at the Design and Analysis phase, we obtain informative results that help us to make decisions before running the process. However, it is at run time when we specially need to be able to perform operations such as solving the resource assignments (i.e. potential performers). Thus, it is necessary to have BP monitoring mechanisms (e.g. BAM techniques) that observe the process during its execution, as well as storage mechanisms in charge of saving generated data that can be useful for subsequent activities and to solve some analysis operations, e.g. to know who has actually performed an activity of the process that has already been executed, i.e. the *actual performer* of a task.

In run-time analysis, the context of the analysis operations is a specific running instance of a BP. Therefore, a person is a potential responsible of an activity if he/she can be the person responsible of such activity in that specific running instance. Note that in this case, it is no longer necessary to analyse the control flow in combination with the resources when calculating the potential performers of the activities because the dynamic behaviour of the process is inherent in its execution. Therefore, having at disposal the information related to the actual task performer of every activity, and every activity instance in case of BPs with loops and/or resource assignments supporting History-Based Distribution (cf. Section §3.2.3), is sufficient to be able to solve at run time any kind of operation that required studying the control flow perspective at design time. Similarly, in order to solve any other type of analysis operation, we can use real information from the process fragment that has already been executed at some point in time, and design-time information for the remaining fragment of the BP.

Thus, at run time we could plan operations beyond those introduced in Section §4.2.1, such as "is there somebody that can be allocated to all the remaining activities of this process?", or "in which activities can David take part from this execution point on?". Notice, however, that optimized algorithms for the resolution of certain analysis operations may be required in order to reduce their execution time at run time, when the response time may be usually critical.

Few current approaches explicitly focus on providing support for the performance of a variety of run-time analysis operations. Some approaches, though,

do provide runtime engines that include the ability of calculating the potential performers at run time, since it is required for automated resource allocation [2, 3, 9, 115].

Analysis of resources in the Evaluation phase. Although this phase of the BP lifecycle is out of the scope of this thesis, it is convenient to mention that analysing the use of resources once the process execution is over, helps in building statistical analyses that are beneficial in two main directions: (i) to improve resource management in later BP executions [47]; and (ii) to automatically generate resource assignments for the process activities of this and/or other process, based on probabilistic analyses performed from the history logs [78].

Process mining techniques have been developed mainly for post-mortem analysis dealing with the BP resource, control flow and data perspectives, both separately [5, 102, 133] and jointly [47]. Analysis algorithms for the resource perspective focused on the Evaluation phase of the BP lifecycle based on Hidden Markov Models have also been introduced [78, 141].

4.2.4 Analysis Technique

Depending on the decisions made with regard to the other elements (e.g. the phase of BP lifecycle), we will have more or less freedom to select the type of automated analysis technique(s) to use. Let us summarize some mechanisms for automated analysis identified in literature:

- To analyse the control flow of BPs, authors usually rely on Petri Nets [89], as there is a plethora of algorithms already developed to perform checks related to the BP control flow perspective [129]. For instance, the *behavioural profiles* introduced by Weidlich et al. [136] categorize fragment of control flow structures in order to enable the automated analysis of the process models, e.g. for BP similarity measurement purposes [54].

  Labeled Transition Systems are used in some approaches to deal with the formalization of complex behaviour that is difficult to express directly in terms of Petri Net constructs, e.g. in YAWL [8, 9, 132] and the Business Activities [120].

  Formal logics and π-calculus have typically been used for the analysis of control flow as well, especially in the area of BP compliance [83, 135]. For instance,

Governatori and Sadiq [66] introduced the *Formal Contract Language (FCL)*, a language based on deontic logic for the definition and checking of rules related to the order in which the activities of a process have to be performed.

- Petri Nets have been extended to deal with the representation and analysis of the BP data perspective in combination with the control flow. This way, analysis related to how data evolve in a process in order to detect anomalies [16, 26, 107, 109, 122] or compliance problems related to rules that deal with the management of information [108], can be automated.

- Regarding resources, we can use logical formalisms to define the semantics of resource assignment languages and implement the operations required to automatically infer information from the resource assignments [27]. Depending on the assignment language and the analysis operations to be implemented, we may develop ad-hoc algorithms that enable the automated extraction of the required information from the BP models and the resource specifications [20].

4.3 Current Proposals

In the following, we briefly summarize some approaches dealing with the analysis of the BP resource perspective, and classify them according to the foundations introduced in Section §4.2. Following the same criteria as for resource specification (cf. Section §3.3), we have limited the study to approaches that somehow address resource analysis in the field of BPs. Notice that the amount of approaches dealing with resource analysis is significantly lower than the number of proposals for resource specification, which were described in Section §3.3. However, some approaches face both issues.

Bertino et al.'s Constraint Specification Language (CSL). Besides introducing a CSL [20] for expressing resource assignments in WFs, other goal of this work was to perform *consistency* analysis and planning, for which the authors developed the so-called *constraint analysis and enforcement module*. The approach checks the consistency of a BP model regarding resource specification, at design time. However, taking into account the control flow of the process is not necessary, as the authors assume they are working with error-free sequential process models. Although it is not explicitly mentioned, some potential performers-like operation must be implemented within the scope of the approach in order to deal with consistency. The formalization has been made with Logic Programming.

Business Activities. Strembeck and Mendling [120] relied on Petri Nets to check the consistency of the Business Activities. As a consequence of this work, several potential conflicts in this type of models were identified. Then, as an extension of the Business Activities, Schefer et al. [115] developed a set of algorithms to detect potential resource-related conflicts at the level of design-time constraint definition, design-time assignment relations, and runtime task allocation. The 2 analysis operations described in Section §4.2.1 can be found in their work.

WIDE. WIDE [40] allows both automatic and manual allocation of tasks to resources. In the former case, the local scheduler module is responsible for dispatching requests for allocation of tasks to agents, and it uses different criteria for agent selection, e.g. workload, availability of agents, and priorities. The only analysis operation mentioned in the specification of WIDE is the calculation of the *potential performers* of the BP activities, which is performed at run time. We assume the only BP dimension involved in the analysis is the resource perspective since nothing is said about needing information coming in data objects to solve the resource assignment expressions. The technique or formalism used is neither mentioned.

YAWL. YAWL 2.0 [9] is equipped with a run-time engine that deals with resource offering and allocation, in such a way that the resource assignments are automatically solved during BP execution. Thus, the *potential performers* of the tasks are automatically calculated at run time. The technique or algorithms used are not specified.

ARIS. Similarly to YAWL, ARIS [113] addresses the resolution of resource assignments at run time, since it is the main feature a BPMS should provide regarding resource management. As for task duty Informed, the person notified is meant to be given in the data flow of the model. Therefore, we assume that data is also taken into consideration in the calculation of the *potential performers*. To the best of our knowledge, design time analysis is outside the scope of ARIS, and no other resource-related analysis operations are supported.

Du et al.'s Resource Management System. The resource engine provided by this resource management system [56], implements the operation in charge of calculating the *potential performers* of the process activities from the resource assignments associated to the tasks. It is done at the Enactment phase of the BP lifecycle, i.e. at run time. Nothing is said about the technique utilized to perform the analysis.

Tan et al.'s Constrained WF System. In this work, Tan et al. [124] focus on solving the *consistency* problem of the resource constraints set in a BP, with the aim of helping

the BP designers to define a sound constrained BP authorization schema. They define consistency rules for constraint-task pairs that guarantee there is no inconsistency, ambiguities and redundancy contained in the set of constraints. "When the constraint-task pairs conform to these rules, then for each user and each role authorized in a task in the process, there is at least one successful BP instance that satisfies all the constraints", as stated in [124]. Like in the CSL developed by Bertino et al. [20], we assume that the operation for calculating the potential performers of the activities is somehow supported by the system. Nothing about the possible existence of exclusive gateways and/or complex process structures is mentioned. Therefore, we assume only the BP resource perspective is taken into account in the approach, which is targeted at design time analysis.

Table §4.1 collects the information of the proposals assessed with regard to the foundations defined in Section §4.2, that is, the analysis operations supported (where *PP* stands for Potential Performers, and *CC* stands for Consistency Checking), the BP perspective(s) involved in the analysis, the BP lifecycle phase in which the analysis is performed, and the analysis technique used, if specified. We use symbol ✓if the feature is supported. Like in Section §3.3, in those cases in which the information related to some issue studied is not included, and there are no clear examples provided that could serve as evidence of how they are treated in the approach, *N/S* (*non-specified*) shows up in the corresponding cell of the table. The justification of the values in the tables can be found in Appendix §B, where we present a broader description of each proposal and the results of the analysis.

As depicted in the table, the analysis operation most implemented is the calculation of the *potential performers*. All the proposals support this operation, indeed. Consistency checking seem not to be a first necessity operation for the authors and for the BPMS developers. In particular, only three BPMSs in the table provide support for analysis operations further than the potential performers of the activities [20, 115, 124]. We believe the reason may be that the expressiveness of the underlying language for resource specification is limited, as many BPMSs focus on assigning persons, roles and/or groups, indeed, e.g. jBPM, ProcessMaker, Activiti.

It is noteworthy though the small number of analysis operations supported by the current approaches. In particular, the fact that checking validity and consistency in BPs is not a first-level need for the authors, especially regarding the existing BPMSs (notice that none of the three BPMSs in the table implement analysis operations different from the potential performers) [20, 115, 124].

Approach	Analysis Operations		BP Perspectives	BP Lifecycle Phases	Analysis Technique
	PP	CC			
Bertino et al.'s CSL [20]	✓	✓	Resources	Design and Analysis	Logic Programming
Business Activities [115]	✓	✓	Resources, Control Flow	Design and Analysis, Enactment	Petri Nets
WIDE [40]	✓		Resources	Enactment	N/S
YAWL [9, 132]	✓		Resources	Enactment	N/S
ARIS [113]	✓		Resources, Data	Enactment	N/S
Du et al.'s Resource Management System [56]	✓		Resources	Enactment	N/S
Tan et al.'s Constrained WF System [124]	✓	✓	Resources	Design and Analysis	N/S

Table 4.1: Evaluation of current approaches dealing with BP resource analysis

Regarding the BP perspective(s) taken into account in the analysis, five out of the seven approaches studied analyse the *resources in isolation* [9, 20, 40, 56, 124]. There are mainly three causes for this:

- The resource assignment language they use does not consider the possibility to use information that comes from data objects handled by the process activities, e.g. the role of the person that must perform task duty Responsible for an activity is indicated in a data object accessed by the activity. Therefore, data are outside the scope of the analysis.
- The authors assume that the BPs are sets of *sequential* activities [20]. Thus, the location in exclusive branches of activities that implement the SoD or BoD patterns, is no longer a problem. As a consequence, taking into consideration the control flow of the process may not be necessary to solve certain analysis operations.
- The analysis is performed at run time. In this case, the step that checks whether the BP is valid is skipped, and the approaches work with real data for the analysis [9, 40, 56]. However, this has one risk: when there are exclusive bifurcation structures and the resource assignments are not properly defined, deadlocks may occur during BP execution. Only in the approach introduce by Schefer et al. [115] to deal with the analysis of the Business Activities, it seems to be considered the

control flow of the process together with the resource perspective, both at design time and at run time.

As far as the BP lifecycle phase is concerned, there is an interesting observation. The approaches that come from academia and have not been deployed in real BPMSs [20, 124], focus the analysis on the Design and Analysis phase. The only one exception is the approach proposed by Schefer et al. [115], which defines the analysis operations both for design time and run time, as stated above. On the contrary, the BPMSs deal with BP analysis only at the Enactment phase [9, 40, 56, 113].

In the last column of Table §4.1, the N/S symbol is abundant because most of the approaches do not give details about the implementation or the technique specifically used to solve the resource assignments of the activities.

4.4 SUMMARY

In this chapter, we have introduced the foundations to be considered in the analysis of resources in BPs, and we have evaluated the existing proposals dealing with resource analysis with respect to those foundations, to be named: (i) the analysis operations supported; (ii) the BP perspective(s) considered in the analysis; (iii) the BP lifecycle phase(s) at which the analysis is carried out, especially focusing on the Design and Analysis, and Enactment phases; and (iv) the formalism or technique used to perform the analysis. The result of this study will be used to understand the existing weak points or problems in resource analysis in BPs (cf. Section §5.3), and they will thus constitute goals to be achieved with the solutions proposed in this doctoral thesis.

Part III

OUR CONTRIBUTION

5

Motivation

"A creative man is motivated by the desire to achieve, not by the desire to beat others"

Ayn Rand (1905–1982),
Writer

The analysis of literature allows us to identify deficiencies that need to be overcome. The goal of this chapter is to define the problems we are going to address in this thesis, and to briefly introduce the contributions developed, as stated in Section §5.1. The problems derived from the study of the state of the art on resource specification and analysis, are defined in Sections §5.2 and §5.3, respectively. In Section §5.4 we present the results of the assessment of the current approaches with respect to the problems described, in order to detect the most highlighting shortcomings. Then, in Section §5.5 we show the overview of the contributions of the thesis, along with a summary of each one of them. Finally, the chapter is closed in Section §5.6, sidestepping to the detailed explanation of the contributions, provided in the following chapters.

5.1 Introduction

In Chapters §3 and §4, we described the foundations of the specification and analysis of resources in BPs, respectively, and we featured the related approaches according to them. In this chapter we explicitly define the problems related to resource specification and analysis that we have identified and which will be addressed in this thesis. After that, we review the literature on the basis of the identified problems, in order to figure out the support provided by the current approaches with regard to every problem, and we show an overview of the solution proposed in this thesis to deal with the problems, providing a brief description of the modules that make up the solution. Each module will be individually described in the next chapters.

5.2 Problems Related to Resource Specification

Chapter §3 introduced the foundations related to the specification of resources in BPs, namely: (i) the organisational model that is used to assign resources; (ii) the task duties involved in each process activity; (iii) the assignment patterns supported from a list of twelve patterns; and (iv) the binding strategy followed to link the resource specification to the BP model. From the result of the study of the current approaches that we carried out, described in Section §3.3, we have drawn three main conclusions. First, the need to provide *traceability* between BP models and organisational models. Second, the advantages of having an *expressive* resource specification method in terms of both the capability to specify resource assignments for the different task duties that can be associated to the activities, and the support provided for the specification of the assignment patterns. Finally, the convenience of providing *flexibility* to allow the user to choose the desired binding strategy to use for the specification of resources in BPs. In the following, we provide further descriptions of these aspects.

5.2.1 Traceability

We define traceability as the capability to express resource assignments using concepts from an organisational model. It bridges a well-known gap between BP models and organisational models [86, 120, 131], allowing the use of concepts from one model in the other. Thus, the resource assignments related to the activities of a process model can make reference to real data contained in the organisational model of the company,

e.g. roles, positions, persons, capabilities, and so on.

Having resource-aware BP models that are traceable to an organisational model enables the automation of work in different directions. On the one hand, it makes it possible to infer interesting information from a resource-aware BP model, such as: (i) the *potential performers* of the task duties for every BP activity (cf. Section §4.2.1), both at design time and at run time; or (ii) the *potential set of activities* each person of an organisation can be allocated at run time. We refer to Chapter §4 for details on resource analysis in BPs.

On the other hand, traceability between BPs and organisational models enables the design-time detection of inconsistencies between the resource assignments associated to the activities of a BP and the structure of the organisation where it is used, e.g. non-existent roles or persons that have been assigned to process tasks.

5.2.2 Expressiveness

We define expressiveness as the capability to offer a wide catalogue of resource assignment expressions to the user. The more variety of options provided, the more expressive the resource specification approach is, and thus, it is more likely to fulfill the needs and/or expectations of more organisations.

Although there is not a standardized way to evaluate the expressiveness of an approach dealing with resource specification, the use of the WRPs as an evaluation framework for such a purpose has been spread in recent years [17, 69, 86, 106, 120, 124, 131]. We assess expressiveness in terms of the *task duties* and the *assignment patterns* supported for resource specification (cf. Sections §3.2.2 and §3.2.3, respectively). Thus, our expressiveness assessment criteria are more complete, as they include the concept of task duty.

5.2.3 Binding Flexibility

We define binding flexibility as the capability of allowing a user to choose the binding strategy to use for resource specification in an organisation, or for a specific BP (cf. Section §3.2.4 for a description of several binding strategies). This implies that the resource specification solution must provide several alternatives among which the user can select which one to use. It is also desirable providing support to automatically switch between different strategies in order to adapt the specification to the new needs

or requirements of the company, as these may change through time. To the best of our knowledge, current approaches do not provide flexible solutions.

5.3 Problems Related to Resource Analysis

In Section §4.3 we studied the current approaches dealing with the analysis of the BP resource perspective according to some analysis foundations, to be named: (i) the analysis operations supported by the approach; (ii) the BP perspective(s) considered in the analysis; (iii) the BP lifecycle phase(s) at which the analysis is carried out, especially focusing on the Design and Analysis, and Enactment phases; and (iv) the formalism or technique used to perform the analysis.

We have identified some problems related to resource analysis in BPs, which can be defined in terms of eleven analysis operations for which support should be provided both at design time and at run time[1] (i.e. at the Design and Analysis, and Enactment phases of the BP lifecycle). Two of the operations were already defined in Section §4.2.1, because implementations of them are provided by some existing approaches. Other operations have already been mentioned and/or described in literature, but support for them is missing, e.g. those related to data access control, defined by Künzle and Reichert [79]. The rest have been defined by us for the first time, to the best of our knowledge.

The operations have been classified in two groups, as depicted in Table §5.1. Within each group, there is a core analysis operation (in *cursive*), on which the other members of the group rely to perform the analysis required. The operations have been defined to be as reusable as possible. Furthermore, *compositions* of them can be implemented. For instance, typical operations for set comparison used in Set Theory [58] can be applied to figure out whether the *potential performers* of two given activities are exactly the same, or whether the set of activities that a specific resource can be allocated (i.e. the Potential Activities operation) is a subset of the set of activities potentially allocated to other resource.

[1] We remind the reader that post-mortem analysis is outside the scope of this thesis.

Group	Analysis Operations
Person-Activity Operations	*Potential Performers (PP)* Potential Activities (PA) Consistency Checking (CC) Non-participants (NP) Permanent Participants (PePa) Critical Participants (CP) Indispensable Participants (IP)
Person-Data Operations	*Potential Accessors to Data States (PADS)* Potential Accessors to Data Objects (PADO) Potential Data States Allowed for a Person (PDSAP) Potential Data Objects Allowed for a Person (PDOAP)

Table 5.1: Classification of resource-related analysis operations

5.3.1 Person-Activity Operations

The operations in this group deal with the relationship between resources and the activities they can participate in. They can be applied to any task duty associated to a BP activity. Thus, in the definitions below, it is assumed that $TaskDuty = [R, A, S, C, I]$ (cf. Section §3.2.2). Furthermore, we assume that the operations are executed in a specific BP, whose BP model is syntactically correct (a.k.a. valid BP) regarding the resource perspective, i.e. all the resource assignments associated to the activities can be solved.

Potential Performers (PP): as stated in Section §4.2.1, this operation calculates the resources that are candidate to perform a specific task duty for a BP activity. It takes a specific activity and a specific task duty, and it returns the set of persons of the organisation that meet the requirements set in the assignment of the task duty for the activity specified.

$$PP(Activity, TaskDuty) : Person[*]$$

Potential Activities (PA): this is the opposite operation of Potential Performers, that is, it lists the activities that may be allocated to a specific resource with regard to a specific task duty during a process instance execution. It takes a resource-aware BP model, the identity of a specific person, and the task duty to be checked, and it returns the activities that can be potentially allocated to the person regarding that task duty.

$$PA(Person, TaskDuty) : Activity[*]$$

Consistency Checking (CC): an activity is consistent with regard a task duty if there is at least one person that is allowed to perform the task duty for the activity, according to the assignment expression associated to the task duty. This operation receives an activity and a task duty, and returns whether the activity is consistent with regard to such a task duty. A BP model is consistent if all its activities are consistent.

$$CC(Activity, TaskDuty) : Boolean$$

Non-participants (NP): this operation checks if there is any person that never participates in a BP for a specific task duty. It takes a set of activities of a business process and a task duty, and it returns the set of persons that can never participate in them performing such a task duty, if any.

$$NP(Activity[*], TaskDuty) : Person[*]$$

Permanent Participants (PePa): this operations checks if there is somebody that *can* participate in all the activities of a BP for a specific task duty. It takes a set of activities of a business process and a task duty, and it returns the set of persons that can participate in all of those activities performing such a task duty, if any.

$$PePa(Activity[*], TaskDuty) : Person[*]$$

Note that this operation does not force a resource to actually perform the task duty for all the activities in all the BP instances. Indeed, there may be process instances in which the resource does not participate because the task duty is allocated to other resources at run time.

Critical Participants (CP): this operations takes a set of activities and a task duty, and returns the critical participants for them with regard to the given task duty. An activity is a critical activity for a given task duty if it has only one potential performer for such a task duty. A critical participant is the potential performer of a critical activity.

$$CP(Activity[*], TaskDuty) : Person[*]$$

Indispensable Participants (IP): this operations takes a set of activities and a task duty, and returns the indispensable participants for them with regard to the given task duty. A person is an indispensable participant if it is the critical participant of a mandatory activity, i.e., an activity that always takes place in the BP. Therefore, the indispensable participants are those people without whom the BP always gets blocked.

$$IP(Activity[*], TaskDuty) : Person[*]$$

5.3.2 Person-Data Operations

We assume that a person has access to a data object if he/she participates in activities that need the data object. Considerting this issue, the operations in this group aim at inferring information about the permissions that the resources of an organisation have to access the information handled in a process, where $AccessType = [Read, Write, Read/Write]$, and at which *states* in the lifecycle of a data object it can be manipulated by a specific resource, if allowed. Thus, in order to automate these analysis operations, we assume that the BP model being analysed is *data-aware*, that is, it contains information about the data objects accessed by the activities. Data states are also assumed to be modelled for the execution of some of the activities.

Futhermore, in this thesis we assume that the person-data operations are applied to four task duties at the same time, specifically $TaskDuty = R \bigcup A \bigcup S \bigcup C$ (cf. Section §3.2.2). Other criteria on the task duties involved could also be acceptable. Finally, we assume that the operations are executed in a specific BP, whose BP model is syntactically correct (a.k.a. valid BP) regarding the resource and data perspectives.

Potential Accessors to Data States (PADS): this operation analyses the BP model to figure out who has a specific access type to one or more specific data states. It takes a list of data states and an access type, and it returns the persons that has that type of access on data objects that are in the states indicated.

$$PADS(DataState[1..*], AccessType) : Person[*]$$

Potential Accessors to Data Objects (PADO): this operation analyses the BP model to figure out who has a specific access type to one or more specific data objects. It takes a set of data objects and an access type, and it returns the persons that are allowed to access all of the given data objects regardless of their state.

$$PADO(DataObject[1..*], AccessType) : Person[*]$$

Potential Data States Allowed for a Person (PDSAP): this analysis operation returns the states of a specific data object that a given person can access in a process instance. It takes a person, a data object and an access type, and it returns the states of the data object that the person can access with the given access type. This operation can only be applied to data objects that have data states defined.

$$PDSAP(Person, DataObject, AccessType) : DataState[*].$$

Potential Data Objects Allowed for a Person (PDOAP): this analysis operation calculates the data objects that a given person can access in a process instance. It takes a person and an access type, and it returns the data objects that he/she can access with the given access type. If the data objects have data states, it returns the data objects that can be accessed by in at least one data state.

$$PDOAP(Person, AccessType) : DataObject[*]$$

The implementation of the four operations defined above requires taking into consideration the BP control flow, data, and resource perspectives. In particular, when dealing with specific states of data objects, it is necessary to know in which state the data is accessed by the activity. This often requires, among other things, taking into account the dynamic behaviour of the process, that is, the control flow. The lifecycles of the data objects of a BP could be seen as a *data-centered view of the process*, as will be detailed in Section §10.2.

5.4 Analysis of Current Solutions

In this section, we review the approaches dealing with resource specification and analysis in BPs that we studied in Chapters §3 and §4, using the problems described in Sections §5.2 and §5.3 as comparison framework.

5.4.1 Solutions Dealing with Resource Specification

Table §5.2 collects the approaches studied in Section §3.3, indicating for each of them whether (i) it is traceable with an organisational model; (ii) it is expressive regarding the assignment patterns and the task duties; and (iii) it is flexible, that is, it allows one to use more than one binding strategy for resource specification.

We use symbol ✓ if the feature is supported, and blanks when it is not. Symbol *N/S* (non-specified) indicates that the corresponding information could not be extracted from the specification of the approach.

Expressiveness is treated in a special way, since it is susceptible to subjectiveness. In order to make the evaluation as objective as possible, expressiveness criteria has been established. Specifically, the expressiveness of an approach is:

6 - *Very high* if it supports more than 7 assignment patterns and several task duties.

5 - *High* if it supports more than 7 assignment patterns for task duty Responsible.

4 - *Medium-high* if 4-7 assignment patterns and several task duties are supported.

3 - *Medium* if 4-7 assignment patterns for task duty Responsible are supported.

2 - *Low-medium* if it supports less than 4 assignment patterns and several task duties.

1 - *Low* if it supports less than 4 assignment patterns for task duty Responsible.

An asterisk (∗) accompanying the expressiveness degree indicates that by supporting one more assignment pattern, the approach would upgrade to the next level. Blanks in the table indicate that the information for that feature could not be extracted, either because it was out of the scope of the approach, or because it was not well specified in the proposal.

As shown in Table §5.2, most of the approaches dealing with resource specification use an organisational model as ground for the definition of the resource assignments for the process activities. However, standards such as BPMN [94] and BPEL [2] do not provide traceability.

As far as expressiveness is concerned, there is much variety, but more than the 50% of the approaches do not exceed the medium degree of expressiveness, standing out 1s and 3s. It means that as average, the current approaches support less than eight assignment patterns out of those defined in Section §3.2.3, and provide support only for the resource Responsible for the BP activities.

Regarding flexibility, the existing approaches are not flexible with regard to the binding strategy offered. This was to be expected, since when we address a problem we usually develop a solution that provides the functionality required to meet the

Approach	Traceability	Expressiveness	Flexibility
RBAC [61, 82]	✓	1 - L	N/S
XACML [4]	✓	1 - L	N/S
Bertino et al.'s CSL [20]	✓	3 - M	
Business Activities [120]	✓	3 - M	
WIDE [40]	✓	4 - MH	
Extended WIDE [41]	✓	4* - MH*	
Russell et al.'s [105] Organisational Metamodel	✓	N/S	N/S
YAWL [9, 132]	✓	5 - H	
BPMN 1.0 Ext. by Grosskopf [69]		4 - MH	
BPMN 1.0 Ext. by Wolter and Schaad [139]	✓	3 - M	
BPMN 1.1 Ext. by Meyer [86]	✓	3 - M	
BPMN 1.1 Ext. by Awad et al. [17]	✓	5 - H	
BPMN 2.0 [94]		4 - MH	
BPEL4People [2] / WS-HumanTask [3]		4 - MH	
ARIS [113]	✓	4* - MH*	
EO [126]	✓	2* - LM*	N/S
Du et al.'s Resource Management System [56]	✓	3 - M	
Team-Enabled WF Reference Model [131]	✓	5 - H	N/S
Tan et al.'s Constrained WF System [124]	✓	3 - M	N/S
HRMM [78]	✓	1 - L	N/S
RACI [44]		2* - LM*	

Table 5.2: Solutions dealing with resource specification in BPs

requirements in a single way, and we tend not to consider offering several options or configuration features to the user.

Assessing the three properties jointly, the best proposals are YAWL [9], the BPMN 1.1 extension described by Awad et al. [17], and the WF reference model dealing with teamwork introduced by van der Aalst and Kumar [131]. Please, note that the second one is very closely related to the extension of BPMN 1.1 described by Meyer in [86]. In fact, we stated in Section §3.3 that Meyer's proposal somehow constituted and improvement or extension of the work headed by Awad et al. [17]. However, we have taken into consideration only the implementation provided by Meyer for the evaluation of the approach, hence its features are reduced with respect to the theoretical contribution.

5.4.2 Solutions Dealing with Resource Analysis

Tables §5.3 and §5.4[2] collect the approaches studied in Section §4.3, indicating for each of them the analysis operations supported, from those defined in Section §5.3. The operations are identified with the acronyms shown in Table §5.1 and in the definitions of the operations. Furthermore, we use acronym *DT* to indicate that support for the design-time execution of the operation is supported (i.e. in the Design and Analysis phase of the BP lifecycle), and *RT* when it is supported at run time (i.e. in the Enactment phase). A blank in a table 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 believe that for the approaches supporting some of the principal operations of some group of analysis operations, support for the other operations of the group could be developed by extending the approach, at a not very high cost (regarding time and effort).

Approaches	Person-Activity Operations						
	PP	**PA**	**CC**	**NP**	**PePa**	**CP**	**IP**
Bertino et al.'s CSL [20]	DT		DT				
Business Activities [120]	DT & RT		DT				
WIDE [40]	RT						
YAWL [9, 132]	RT						
ARIS [113]	RT						
Du et al.'s Resource Management System [56]	RT						
Tan et al.'s Constrained WF System [124]	DT		DT				

Table 5.3: Current support for Person-Activity Operations

	Person-Data Operations			
	PADS	**PADO**	**PDSAP**	**PDOAP**
Bertino et al.'s CSL [20]				
Business Activities [120]				
WIDE [40]				
YAWL [9, 132]				
ARIS [113]				
Du et al.'s Resource Management System [56]				
Tan et al.'s Constrained WF System [124]				

Table 5.4: Current support for Person-Data Operations

[2]Blanks are left intentionally.

As shown in the tables, the only operation actually supported by all the approaches is that in charge of calculating the potential performers of the activities of a process. As stated in Section §4.3, this is not very significant, since it is the most basic operation in which an organisation may be interested in order to deal with resource allocation at run time. Enabling automated run-time allocation is exactly the purpose of the BPMSs in the table [9, 40, 56, 113, 120]. Other proposals implement Potential Performers at design time, since it may also be convenient to know the potential performers for an activity before task execution, so that changes derived from last-minute decisions can be made [20, 115, 120, 124].

Consistency (or ad-hoc variants of it) has been addressed often [20, 120, 124], being the support provided always focused on the Design and Analysis phase of the BP lifecycle. It is reasonable, since it is before launching a process that we should make sure that the process does not contain inconsistencies regarding resource management and, thus, there will always be somebody to which every activity will be able to be allocated during the execution of the process.

To the best of our knowledge, none of the approaches in the literature on resource management in BPs, has yet developed support to automate the performance of the analysis operations dealing with data access control, even though some of them have been pointed out in literature [79], as stated in Section §5.3.

5.5 Overview of Our Solution

This doctoral thesis is aimed at enhancing the current support for the management of the BP human resource perspective. Specifically, we focus on resource specification and analysis in BPs. In the following, we outline our overall solution to overcome the shortages found in current approaches regarding the problems defined in Sections §5.2 and §5.3. It is depicted in Figure §5.1, where the contributions related to resource specification are placed in the left column, and those dealing with analysis problems occupy the right column. The modules within each column are separated into layers, so that modules in one layer rely on or use the functionality provided by the modules that are just below them. Let us sum up each contribution of this thesis according to the modules shown in the figure.

C1. RAL. A new language, called RAL, has been developed with the aim of providing a traceable, expressive and user-friendly way to define resource assignments in

BP models. The underlying organisational metamodel is an excerpt of the one described by Russell et al. [105] (cf. Section §3.3 or Appendix §A for further details). It supports 10 assignment patterns, and it can be used to specify resource assignments independently of the specific task duty. Furthermore, RAL syntax is close to natural language in order to increase the readability of the expressions.

C2. Flexible Resource Specification. With the aim of providing a flexible solution for resource specification, we offer two different ways of assigning resources to the BP activities. One is an all-in-one approach in which RAL is directly used within a BPMN model (C2.1). The other one implements the separate binding strategy. Specifically, starting with a resource-unaware BP model, it enables the automated definition of the 5 task duties defined in Section §3.2.2 for the activities of the process within the process model, by complementing a RASCI matrix with the so-called *binding information* specified with RAL (C2.2). Furthermore, a set of transformations have been developed to enable switching from the separate specification model to an all-in-one binding that involves the 5 task duties, automatically (C2.3).

C3. RAL Semantics. RAL has been endowed with a formal semantics that enables the automated execution of analysis operations on a RAL-aware BP model. The semantics is defined in Description Logics (DLs), and it is the ground for the implementation of the analysis operations that we introduce.

C4. Automated Person-Activity Operations. We introduce the extensions that need to be applied to the DL-based Knowledge Base (KB) that defines RAL semantics in order to automate the resolution of the 11 analysis operations described in Section §5.3. Thus, these extensions prepare the KB to automatically deal with operations that analyse the BP resource perspective in isolation, and those that require information from the control flow and/or data perspectives, both at design time and at run time. In particular, to deal with control flow we rely on concepts defined by Weidlich et al. when they introduced the so-called Behavioural Profiles [136], and on functionality provided by BPMN-Q [15], a language to query the control flow and data perspectives of BPs. Based on such extensions, we provide a reference implementation of the operations that deal with the relationship between resources and BP activities at design time and run time.

C5. Automated Person-Data Operations. As an intermediate step for the development of the extensions of the DL-based KB, we introduce a procedure to automatically

generate a data-centered view of BPs, called the BP2OLC procedure. This, together with the rest of features of the extensions, enables the automation of the 4 analysis operations related to the relationship between resources and data that we described in Section §5.3. We provide a reference DL-based implementation to perform the operations at design time and at run time.

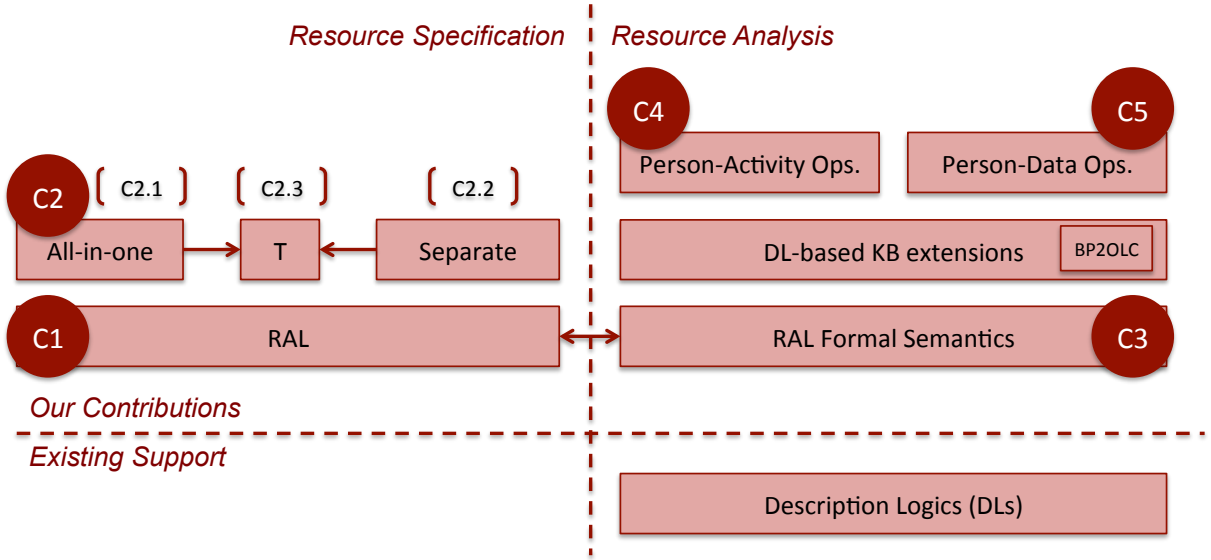


Figure 5.1: Overview of our solution

All these contributions have been implemented together in a system we have called CRISTAL (Collection of Resource-centrIc Supporting Tools And Languages). Details about them and about CRISTAL can be found in the next chapters.

5.6 SUMMARY

In this chapter, we have provided a definition of each problem extracted from the study of the literature related to resource specification and analysis in Chapters §3 and §4, and we have reviewed the approaches we regard to the specific problems identified. As a result, we have figured out the real support provided by nowadays solutions, which has been used as starting point to develop our own solution, briefly summarized in Section §5.5.

RAL: Resource Assignment Language

"A successful economic development strategy must focus on improving the skills of the area's workforce, reducing the cost of doing business and making available the resources business needs to compete and thrive in today's global economy"

Rod Blagojevich (1956–),
Politician

From the literature review on the specification of resources in BPs, we concluded that there is not yet a traceable, highly expressive and flexible solution for resource specification. In this chapter, we introduce a resource assignment language that has its ground on a well-known organisational metamodel (traceability), covers eleven out of the twelve assignment patterns for any task duty we have used in the study (expressiveness), and is not conceived to be used with a specific binding strategy (flexibility). The goal of the language and the introduction to the contents of the chapter are presented in Section §6.1. The organisational and BP metamodels imported in the language and their connection, are defined in Sections §6.2 and §6.3, respectively. An application scenario that will be used throughout this thesis is also presented in these section exemplifying the concepts introduced. The specification of the language is then described in Section §6.4, accompanied by examples of use based on the application scenario. An example of how it can be used to express each assignment pattern is included In Section §6.5 and, finally, a summary of the chapter is presented in Section §6.6.

6.1 INTRODUCTION

In Chapter §3, we described the binding model as the connection between the BP model and the organisational model aimed at assigning resources to activities. Furthermore, in Chapter §5, we identified problems of current binding models concerning traceability, expressiveness, and flexibility regarding the binding strategy. In this chapter we aim at overcoming these problems by means of *Resource Assignment Language (RAL)*. RAL is a *Domain Specific Language (DSL)* developed to specify resource assignments in BPs. Specifically, RAL allows formulating expressions that can be used to define who can perform an activity in the BP such as:

```
RAL 1: IS Samuel
RAL 2: NOT (IS PERSON ACCOUNTABLE FOR ACTIVITY CreateRP IN ANOTHER INSTANCE)
RAL 3: (HAS ROLE DocumentWriter) OR (HAS POSITION ACSigner)
RAL 4: SHARES SOME ROLE WITH PERSON IN DATA FIELD Document.DueSigner
RAL 5: (HAS UNIT ISAgroup) AND (IS PERSON RESPONSIBLE FOR ACTIVITY CreateDoc)
```

Looking at the expressions that can be defined with RAL, it is possible to find references to elements that belong to an organisational model such as Samuel, DocumentWriter or ISAgroup, and elements that belong to a BP model such as CreateRP or Document.DueSigner. Thus, as depicted in Figure §6.1, RAL metamodel needs to import both an organisational metamodel and a BP metamodel. Specifically, RAL imports an organisational metamodel that is strongly inspired in the metamodel described by Russell et al. [105], and a BP metamodel highly inspired in BPMN [94], which abstracts the most important characteristics of the current approaches of such types of meta models.

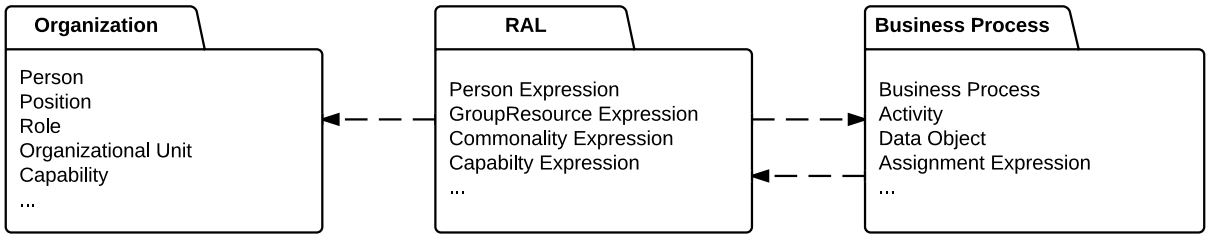


Figure 6.1: Overview of the RAL metamodel

As a result of RAL metamodel, RAL expressions are *traceable* with organisational concepts. Furthermore, resource assignments for any of the five task duties described

in Section §3.2.2 can be defined with the language, which covers eleven out of the twelve assignment patterns defined in Section §3.2.3. These two characteristics make RAL a *highly expressive* language, according to the expressiveness criteria we established in Section §5.4.1. Furthermore, RAL is independent of the binding strategy used, so it can be used for all-in-one, split, and separate modelling of resources in BPs.

6.2 Organisational Metamodel Used in RAL

The organisational metamodel used with RAL is depicted in gray in Figure §6.2. It is part of the metamodel described by Russell et al. [105] that we included in our study on resource specification in Chapter §3. In a nutshell, it consists of persons, positions, roles and organisational units. A *person* might have a set of *capabilities*, such as his/her professional experience. The metamodel is extensible to include new capabilities and does not provide a specific meaning for this term. Each person occupies one or more *positions* within an organisation, which in turn participate in one or several *roles*, and can belong to at most one *organisational unit*, e.g. an organisational team. Organisational units have a *unit type*, which classify them. For instance, regarding the hypothetical project THEOS, we can have the THEOS organisational unit, associated to unit type Project. Another example could be a department called Computer Systems and Languages, which would be an organisational unit associated to the unit type Department. Then, within each organisational unit, there may be a hierarchy of positions representing the lines-of-reporting in an organisational unit, that is, work can be *reported* and/or *delegated* among the members of an organisation according to their positions in the organisational units.

Please, notice that we might use the term *group resource* to refer to concepts that represent groups of people according to this organisational metamodel, i.e., positions, roles and organisational units. Persons can also be named *individual resources*, *individuals* or *resources*, as stated in Section §3.2.1.

A complete example of instantiation of this organisational metamodel can be found in Figure §6.3. Let us assume that we belong to the ISA research group of the University of Seville (Spain), and we participate in a project called *THEOS*. The model shown in the figure corresponds to the hierarchy of *positions* that are involved in such project[1]. Thus, we have an *OrganizationalUnit* THEOS that is of *UnitType* Project. As depicted, there are six positions (THEOS's Project Coordinator, THEOS's Account Del-

[1]Please notice that all the values (roles, positions, persons...) are fictitious.

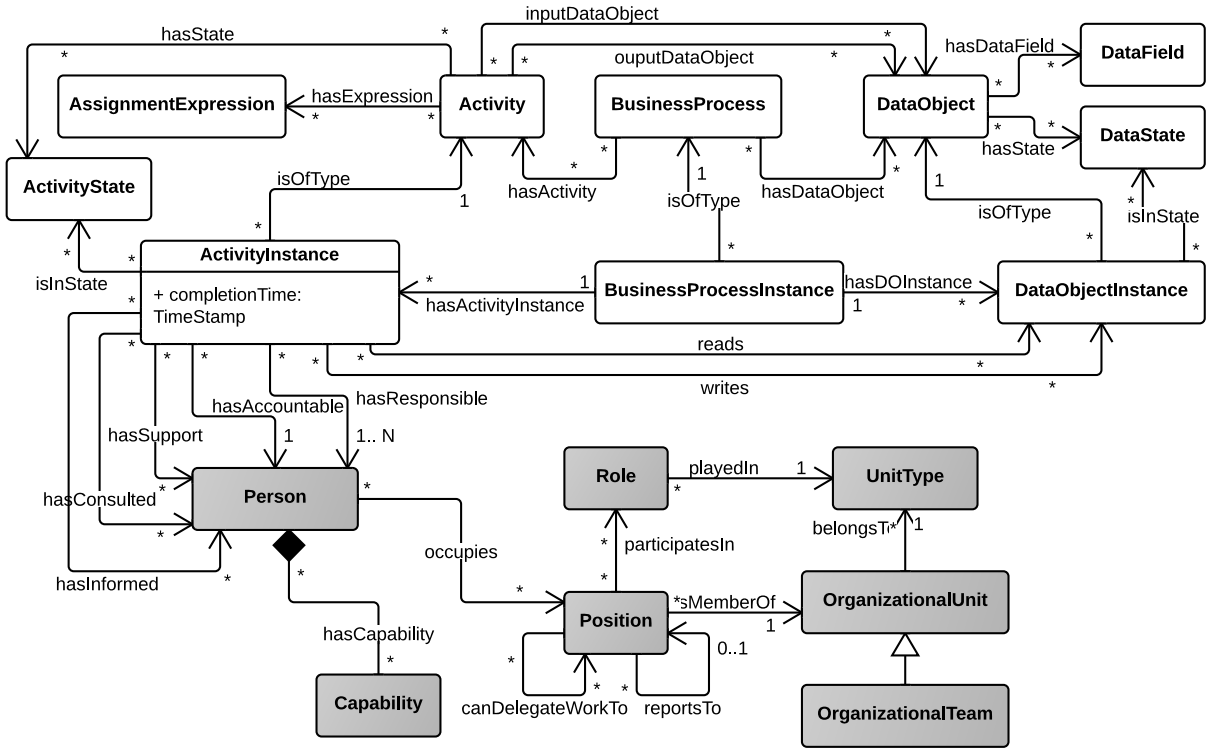
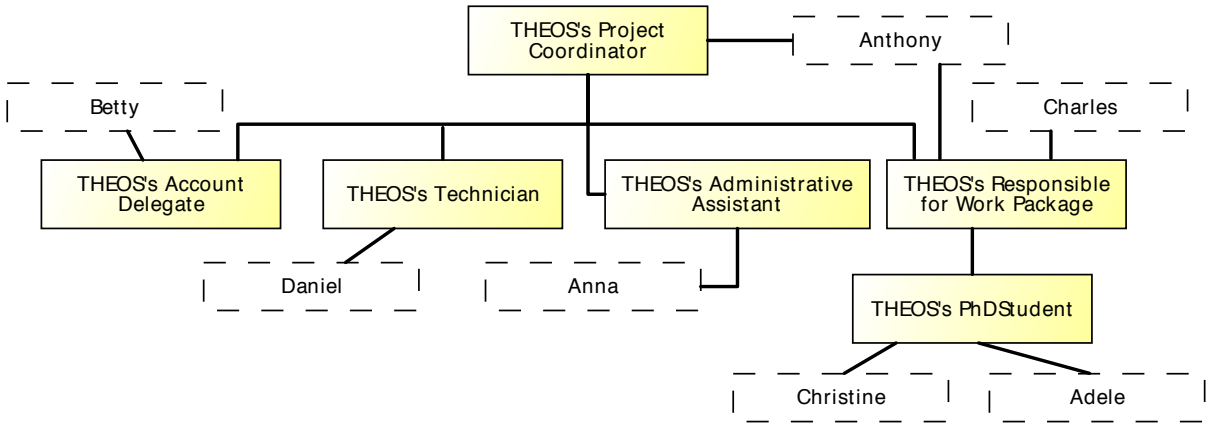


Figure 6.2: RAL metamodel excluding RAL expressions

egate, THEOS's Technician, THEOS's Administrative Assistant, THEOS's Responsible for Work Package, and THEOS's PhD Student), and seven *persons* occupying them. The people that occupy a position can delegate work to those who occupy a position that is below in the hierarchy, and they can report work to people that occupy the immediately upper position. The *participatesIn* relation of the organisational metamodel is summarized in the table attached to the figure. For instance, *Anthony* occupies positions *THEOS's Responsible for Work Package* and *THEOS's Project Coordinator* in the scope of the project. As a responsible for a Work Package he has two roles, and his other position participates in 3 roles according to the model, so Anthony has five roles in total in the organisational unit THEOS. Furthermore, he cannot report work to any member of the company, but he can delegate work to everybody, since he is directly associated to all the positions in the immediately lower level of the hierarchy. A table with the *hasCapability* relation should also be specified, but it has been omitted because it is not necessary to understand the examples provided in this thesis.



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

Figure 6.3: Excerpt of the organisational model of the ISA group for project THEOS

6.3 BUSINESS PROCESS METAMODEL USED IN RAL

The classes of the metamodel related to BPs used in RAL are coloured in white in Figure §6.2. Please, notice that this BP metamodel does not intend to cover all the elements involved in BPM, but to illustrate a fragment of it that is relevant for resource specification.

As depicted in the figure, a *BusinessProcess* can have a set of *DataObjects*, which in turn can contain one or more *DataFields*, and can have different *DataStates* throughout the process execution. The data objects in a process are handled by the *Activities* of the process, which can read and/or write them to consult and/or to modify their content, i.e. the values of their data fields. As stated in Section §2.3 and shown in Figure §6.2, an activity goes through different *ActivityStates* according to the activity lifecycle. During state Allocating, the task duties of the activity are allocated to specific resources. The potential performers from which the actual performers are selected, are specified with *AssignmentExpressions* associated to the activity. There is where RAL fits, as the mechanism to specify the conditions that the potential performers of the task duties must fulfill.

At run time, when a BP is instantiated, instances of the activities and the data ob-

jects used in the BP, are created. Thus, the *DataObjectInstances* are handled by the BP *ActivityInstances*. Thus, the task duties of every *ActivityInstance* are allocated to specific persons, who become the *actual performers* of the task duties for the activity in that BP instance. Hence the link with the organisational metamodel, by means of relations *hasResponsible*, *hasAccountable*, *hasSupport*, *hasConsulted*, and *hasInformed*, representing the five task duties considered (cf. Section §3.2.2). Finally, the completion time of each *ActivityInstance* is recorded in order to enable the automated analysis of history-based assignment expressions, i.e. assignments that are constrained to the assignment of another activity that may have been executed in a previous BP instance.

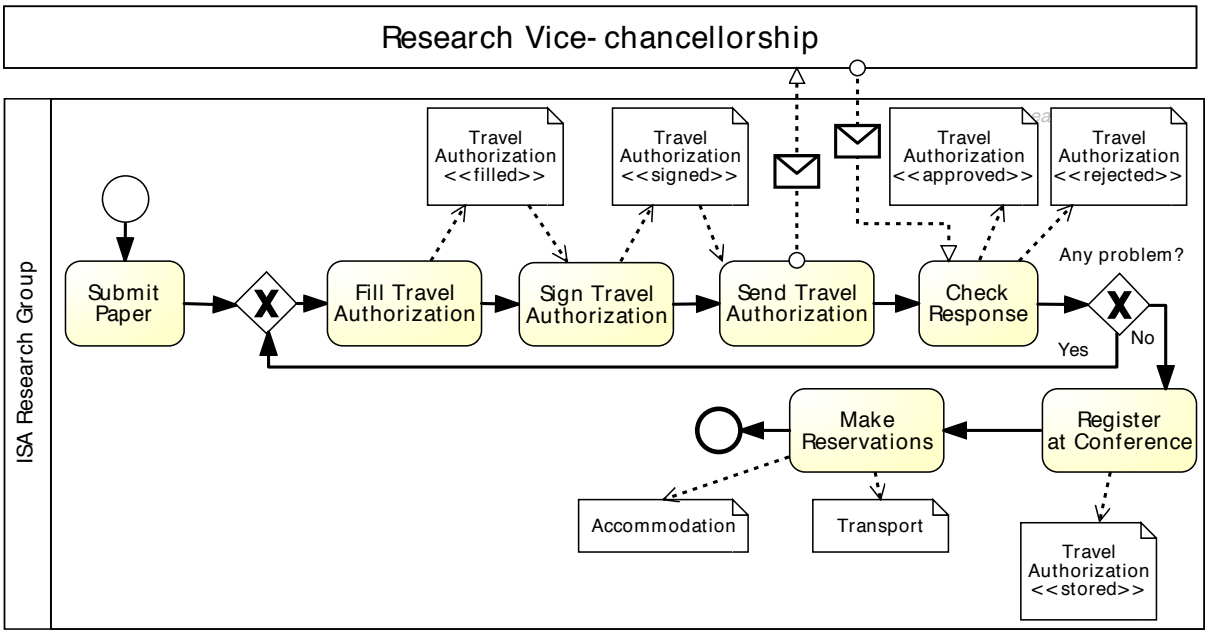


Figure 6.4: BP to manage the trip to attend a conference

Let us instantiate the BP metamodel with an example. The procedure illustrated in Figure §6.4 represents a *collaboration* between two BPs modelled in BPMN 2.0 [94]: one BP is developed at pool *Research Vice-chancellorship* and the other at pool *ISA Research Group*[2]. It consists of a simplified version of the procedure to manage the 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 somebody submits the Camera Ready version of a paper that is assumed to have been accepted for publi-

[2]We remind the reader that 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. All the process-oriented concepts used from now on in this dissertation are taken from BPMN 2.0 [94].

cation at a conference. Then, one of the authors must fill in a document (i.e. the *Travel Authorization*) requesting for authorization both to travel to the venue place and to take funds from some funding source. This document must be approved by some person allowed to authorize the applicant to attend the conference and take funds from the funding source specified in a data field of the document, e.g. the project coordinator in case the funding source is a research project. Once signed, it is sent for revision to an external entity of the *Research Vice-chancellorship*, where someone evaluates the request, and checks whether all the conditions required are met. If so, the authorization is approved. Otherwise, it is denied. In the former case, the author that has been authorized to attend the conference, must register at the conference and make the reservations required (i.e. accommodation and transport). In the latter case, the authorization must be filled in again and the evaluation process is repeated until it is finally approved.

6.4 RAL Specification

RAL is a modular language to define resource assignments. It is composed of a set of basic expressions that form RAL Core, and several extensions that enable the definition of advanced types of expressions. Due to its modularity, RAL is open to future extensions dealing with more advanced human resource management. In particular:

- *RAL Core* contains resource assignment expressions directly based on organisational concepts (e.g. positions, roles) and on relations between them derived from the organisational model.
- *RAL Data* extends RAL Core to take into account information that can come from data handled in the process, i.e. part of the information used in a resource assignment expression is specified in a field of a data object.
- *RAL AC* adds to the basic expressions the possibility of specifying assignment patterns SoD and BoD which correspond to two specific types of the so-called access-control constraints: DSoD and DBoD (cf. Section §3.2.3).
- *RAL History* extends RAL AC to enable the specification of the static form of the access-control constraints addressed by RAL AC, i.e. SSoD, SBoD, allowing us to make reference to previous execution instances of a BP to assign resources to the BP activities.

The aforementioned RAL extensions are grounded on the addition of new types of expressions, or on the extension of the types of constraints that can be specified with RAL Core. Furthermore, RAL extensions can be composed with each other, in a way that for instance RAL AC can be used in conjunction with RAL Data. In Figure §6.5, we show how the extensions ground on each other, being RAL Core the basis for all of them. We next describe RAL Core and the extensions, accompanied with examples that use the application scenario introduced in Sections §6.2 and §6.3. Please, notice that from now on we might use term *RAL DACH* to refer to RAL Core with all the aforementioned extensions as a whole. For the sake of simplicity, we show RAL syntax only in *Extended Backus-Naur Form (EBNF)*.

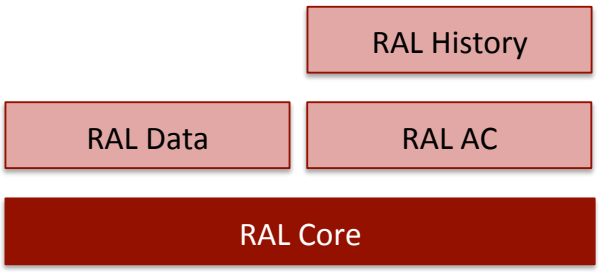


Figure 6.5: RAL DACH: RAL Core and three RAL extensions

6.4.1 RAL Core

RAL Core is composed of eight types of expressions that allow selecting people based on their organisational information. These expressions rely on a set of constraints that constitute the mechanism to reference resources and group resources of an organisation. As RAL has been developed from an excerpt of an organisational metamodel (cf. Section §6.2), many concepts of the metamodel can be explicitly found in the language. Language 6.1 shows RAL Core specification, which includes the following types of expressions and constraints.

RAL Core Expressions

PersonExpr (*line 6*) allows assigning an activity to a specific person, who is indicated by means of a *PersonConstraint*.

GroupResourceExpr is used to assign an activity to a specific group resource, which can be done is several different ways:

HAS (POSITION | UNIT) GroupResourceConstraint (*line 8*) assigns an activ-

Language 6.1 RAL Core specification in EBNF

```
1  Expr := PersonExpr                    | IsAssignmentExpr
2        | GroupResourceExpr             | HierarchyExpr
3        | CommonalityExpr               | NegativeExpr
4        | CapabilityExpr                | CompoundExpr
5
6  PersonExpr := IS PersonConstraint
7
8  GroupResourceExpr := HAS (POSITION | UNIT) GroupResourceConstraint
9        | HAS ROLE GroupResourceConstraint [IN UNIT GroupResourceConstraint]
10
11 CommonalityExpr := SHARES Amount (POSITION | UNIT) WITH PersonConstraint
12       | SHARES Amount ROLE [IN UNIT GroupResourceConstraint] WITH PersonConstraint
13
14 CapabilityExpr := HAS CAPABILITY CapabilityConstraint
15
16 IsAssignmentExpr := IS ASSIGNMENT IN ACTIVITY activityID
17
18 HierarchyExpr := ReportExpr  | DelegateExpr
19
20 ReportExpr := Depth REPORTS TO PositionRef  | IS Depth REPORTED BY PositionRef
21
22 DelegateExpr := CAN DELEGATE WORK TO PositionRef  | CAN HAVE WORK DELEGATED BY PositionRef
23
24 NegativeExpr := NOT '('PersonExpr | GroupResourceExpr | CommonalityExpr | CapabilityExpr')'
25
26 CompoundExpr := '('Expr')' OR '('Expr')'  | '('Expr')' AND '('Expr')'
27
28 PersonConstraint := personName
29
30 GroupResourceConstraint := groupResourceName
31
32 CapabilityConstraint := capabilityID  | CapabilityRestriction
33
34 PositionRef := POSITION GroupResourceConstraint  | PersonConstraint
35
36 Amount := SOME  |  ALL
37
38 Depth := DIRECTLY | λ
```

ity to a specific position or organisational unit, indicated by a *GroupResourceConstraint*.

HAS ROLE GroupResourceConstraint [IN UNIT GroupResourceConstraint] (*line 9*) enables role-based assignment. Optionally, the role can be constrained to a specific organisational unit. We included this option because roles are usually played within an organisational unit (cf. Section §6.2).

CommonalityExpr deals with the Commonality-based Allocation pattern, that is, the assignment of activities to people based on the features they have in common

with other people:

SHARES Amount (POSITION | UNIT) WITH PersonConstraint (*line 11*) assigns an individual that has *some* or *all* position(s) or organisational unit(s) in common with a specific person, indicated by a *PersonConstraint*. In the former case, i.e. people sharing *something*, the concrete group resources that have to be shared are not specified, that is, any position, role or organisational unit in common with the individual indicated makes a resource meet the condition. Thus, we assume that every potential performer of the activity possess some privilege/permission that makes him/her a good candidate to execute it. The latter case, i.e. people sharing *everything*, is more intuitive: only those people sharing all their positions or organisational units with the person specified will be part of the potential performers of the activity at hand.

SHARES Amount ROLE [IN UNIT GroupResourceConstraint] WITH PersonConstraint (*line 12*) is the combination of the two previous assignment expressions. It allows assigning an activity to somebody that has some or all the roles in common with a specific person, who is given by means of a *PersonConstraint*. Optionally, the organisational unit in which the roles have to be searched can be indicated with a *GroupResourceConstraint*.

CapabilityExpr (*line 14*) allows expressing assignments based on the capabilities of individual resources that can be allocated to a task duty for an activity, e.g. *years of experience* or *reputation*[3].

IsAssignmentExpr (*line 16*) indicates that an activity has the same RAL expression as another activity. This avoids having to re-write several times the same assignment, at the same time as it helps saving time and effort and prevents writing mistakes derived from the replication of RAL expressions.

HierarchyExpr allows playing with the positional hierarchy of the organisational model in two different ways:

ReportExpr (*line 20*) allows expressing constraints like "Activity *Request external resolution* must be performed by someone that reports to the *Business Manager*". Attribute *directly* is used for stating whether we want to move up more than one reporting level by transitivity ($directly = false$), or not ($directly = true$).

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

DelegateExpr (*line 22*) is similar to *Report* but using organisational relation *can delegate work to*, i.e., this time we move down in the positional hierarchy. In this case transitivity is implicit.

NegativeExpr (*line 24*) allows expressing the negative form of *PersonExprs*, *GroupResourceExprs*, *CommonalityExprs*, and *CapabilityExprs* in order to state that an activity has to be allocated to somebody that does not have a certain property. The negation is not applied to the other 3 types of expressions mainly because difficulties arise to define their semantics following the same procedure as for the rest of RAL expressions (cf. Chapter §8 for a description of RAL semantics). Besides, their meaning becomes tricky in the negated form and/or the same can be expressed with other types of expressions.

In particular, negating an *IsAssignmentExpr* lacks of coherent meaning. Furthermore, it would involve permitting us to negate any expression type, as the assignment expression of the referenced activity could be any. That would imply allowing the negation of *HierarchyExprs* and *CompoundExprs*, which is contradictory with what we are stating here. The negation of *HierarchyExprs* causes problems due to the transitivity in the *ReportExpr*. Nevertheless, we argue that an organisation is not likely to need to define expressions on the ground of people that cannot report work to a position or person, but based on who can actually do it. Finally, the negative form of a *CompoundExpr* can be expressed by negating the expressions of the compound expression and modifying the operators in question as required.

CompoundExpr (*lines 26*) allows specifying multiple conditions in the assignment, connecting RAL expressions with the OR/AND operators.

RAL Core Constraints

PersonConstraint (*line 28*) limits the scope of the persons that are mentioned in some RAL expressions, e.g. *PersonExpr*. In RAL Core the only one *PersonConstraint* considered consists of indicating the specific identity of the person in question.

GroupResourceConstraint (*line 30*) indicates a group resource required in some types of expressions, e.g. *GroupResourceExpr*. In RAL Core it consists of specifying the identity of the position, role or unit in question.

CapabilityConstraint (*line 32*) consists of either having a certain capability, or meeting a certain condition on the value of a capability. We do not detail term *Capa-*

bilityRestriction in Language 6.1 because it is based on mathematical and logical operators and, thus, its use is quite straight, i.e. it serves for instance to express that an activity has to be carried out by someone with more than three years of experience.

Submit Paper. A *PhD Student* of project *THEOS* is in charge of submitting the paper to the conference.

```
HAS ROLE ProjectsPhDStudent IN UNIT THEOS
```

Fill Travel Authorization. The authorization form must be filled in by somebody that reports to Anthony.

```
REPORTS TO Anthony
```

Figure 6.6: RAL Core expressions for some BP activities

A sample of the use of the RAL Core is shown in Figure §6.6, where we present assignments for activities *Submit Paper* and *Fill Travel Authorization* of the BP shown in Figure §6.4, given the organisational model in Figure §6.3.

6.4.2 RAL Data

RAL Data adds expressiveness related to the Deferred Allocation assignment pattern by allowing us to express that the resource that is assigned to an activity is indicated in a data field, whose value is unknown until run time. Specifically, RAL Data extends *PersonConstraint* and *GroupResourceConstraint* to incorporate this functionality, as shown in Language 6.2 in bold letters. The rest expressions that are not affected by this change, remain exactly like in RAL Core. An example of RAL Data applied to our use case is shown in Figure §6.7.

Language 6.2 RAL Data specification in EBNF. RAL Core elements ommitted

```
1 PersonConstraint := personName
2                   | PERSON IN DATA FIELD dataObject.fieldID
3
4 GroupResourceConstraint := groupResourceName
5                   | IN DATA FIELD dataObject.fieldID
```

Send Travel Authorization. This task must be undertaken by the person referenced in field *Attendee* of data object *Travel Authorization*.

`IS PERSON IN DATA FIELD TravelAuthorization.Attendee`

Figure 6.7: RAL Data expression for activity *Send Travel Authorization*

6.4.3 RAL AC

Language 6.3 shows in bold the extension introduced by RAL Access-Control (a.k.a. RAL AC) to deal with two types of access-control constraints that belong to the assignment patterns, namely DBoD and DSoD (cf. Section §3.2.3 on assignment patterns). The rest of RAL expressions do not vary with respect to RAL Core.

Language 6.3 RAL AC specification in EBNF. RAL Core elements ommitted

```
1  PersonConstraint := personName
2                    | PERSON TaskDuty ACTIVITY activityID
3
4  TaskDuty := RESPONSIBLE FOR
5              | ACCOUNTABLE FOR
6              | PROVIDING SUPPORT FOR
7              | CONSULTANT OF
8              | INFORMED ABOUT
9              | INVOLVED IN
```

In RAL AC, only the *PersonConstraint* is modified. In this case, it is extended to express that the person specified in the constraint must be the performer of one of the five task duties defined in Section §3.2.2 for another activity of *the same* BP instance. In other words, there is a DBoD with another activity. If we do not want to limit the scope to a specific task duty, the option `INVOLVED IN` can be used to express that it does not matter the specific task duty the person performed in the activity referenced.

Notice that in order to express the opposite constraint, i.e. DSoD, it is necessary to use the negation construct provided by RAL (cf. *NOT (Expr)* in RAL Core §6.4.1). Thus, we would state that an activity has to be performed by somebody that is *not* the person who performs a specific task duty for another activity. An example is shown in Figure §6.8, and more examples are provided in Section §6.5 when explaining how RAL can be used to express the assignment patterns.

Sign Travel Authorization. The authorization form can be signed only by a person that can supervise the work developed by the person who filled in the document. Furthermore, this person has to be different from the one that filled in the form, in order to prevent conflicts of interests.

```
(IS REPORTED BY PERSON RESPONSIBLE FOR ACTIVITY FillTravelAuthorization)
AND (NOT (IS PERSON RESPONSIBLE FOR ACTIVITY FillTravelAuthorization))
```

Figure 6.8: RAL AC expression for activity *Sign Travel Authorization*

6.4.4 RAL History

RAL History extends RAL AC by permitting us to make reference to other BP instances, that is, to support SBoD and SSoD too. The new elements added to RAL (specifically to the *PersonConstraint*) are depicted in bold in Language 6.4. In particular, as an extension of the access-control constraints addressed in RAL AC, now we can specify the concrete BP instance in which the referenced activity is executed, among the following options:

- (*Line 5*) The same BP instance currently running. It is similar to RAL AC.
- (*Line 6*) Any instance of the process, including the ongoing one.
- (*Line 7*) Any previous process instance, excluding the ongoing one.
- (*Line 8*) The process instances in which the activity has been completed between two given dates, regardless of if the process instance itself is over or not.

We can also omit the specification of the activity referenced, and indicate only the BP instance in which the person in question has participated. We can limit the scope to people that have performed a specific task duty for any activity of the process, or refer to any person involved in the process without specifying the task duty performed. The constructs shown in lines 10-13, allow one to indicate whether it is necessary to search in (i) the ongoing process instance, (ii) any process instance, (iii) a previous process instance, or (iv) any process instance, provided that the activity was completed between two given dates (similarly to expression in line 8, it is not required the whole process instance being over by that moment).

Several RAL History expressions applied to our application scenario are shown in Figure §6.9.

Language 6.4 RAL History specification in EBNF. RAL Core elements ommitted

```
1  PersonConstraint := personName
2          | PERSON TaskDuty ACTIVITY activityID [HistoryExpr]
3          | PERSON WHO HAS BEEN TaskDuty ANY ACTIVITY IN BPHistoryExpr
4
5  HistoryExpr := IN CURRENT INSTANCE
6          | IN ANY INSTANCE
7          | IN ANOTHER INSTANCE
8          | FROM startDate TO endDate
9
10 BPHistoryExpr := CURRENT PROCESS INSTANCE
11         | ANY PROCESS INSTANCE
12         | ANOTHER PROCESS INSTANCE
13         | A PROCESS INSTANCE BETWEEN startDate AND endDate
14
15 TaskDuty := RESPONSIBLE FOR
16         | ACCOUNTABLE FOR
17         | PROVIDING SUPPORT FOR
18         | CONSULTANT OF
19         | INFORMED ABOUT
20         | INVOLVED IN
```

Check Response. Any participant in any instance of the BP can check the answer received from the *Research Vice-chancellorship*, as long as he/she has some organisational unit in common with the person that submitted the paper in the current process instance.

```
(IS PERSON WHO HAS BEEN INVOLVED IN ANY ACTIVITY IN ANY PROCESS INSTANCE) AND
(SHARES SOME UNIT WITH PERSON RESPONSIBLE FOR ACTIVITY SubmitPaper IN CURRENT
INSTANCE)
```

Register at Conference. The person responsible or accountable for task *Submit Paper* in the ongoing instance is due to register at the conference.

```
(IS PERSON RESPONSIBLE FOR ACTIVITY SubmitPaper IN CURRENT INSTANCE) OR
(IS PERSON ACCOUNTABLE FOR ACTIVITY SubmitPaper IN CURRENT INSTANCE)
```

Make Reservations. Somebody with role *Clerk* or any person that has previously been responsible for this activity can become responsible for making the reservations again, even if he is not the one attending the conference.

```
(HAS ROLE Clerk) OR (IS PERSON RESPONSIBLE FOR ACTIVITY MakeReservations
IN ANOTHER INSTANCE)
```

Figure 6.9: RAL History expression for some activities of the BP

6.5 Assignment Pattern Specification with RAL

In the following, we provide examples of how the assignment patterns we introduced in Section §3.2.3 can be expressed with RAL. Please, notice that all the examples are referred to task duty Responsible of the activities. Assignments for the rest of task duties could be expressed similarly.

Direct Allocation: This pattern represents the ability to specify at design time the identity of the resource that will be perform a task duty for an activity. For instance, *Anna* is responsible for checking the response received from the *Research Vice-chancellorship*, which can be expressed using the following RAL expression in activity Check Response:

```
IS Anna
```

Role-Based Allocation: This pattern represents the ability to specify that a task duty can only be executed by resources with a given role. For example, a *PhD Student* of project *THEOS* is in charge of submitting the paper to the conference (cf. Figure §6.6), which can be expressed using the following RAL expression in activity Submit Paper:

```
HAS ROLE ProjectsPhDStudent
```

Deferred Allocation: This pattern represents the ability to defer specifying the identity of the performer of a task duty until run time. For instance, during execution of the process, instances of the *Send Travel Authorization* task must be undertaken by the person referenced in field *Attendee* of data object *Travel Authorization* (cf. Figure §6.7), which can be expressed using the following RAL expression in activity Send Travel Authorization:

```
IS PERSON IN DATA FIELD TravelAuthorization.Attendee
```

Separation of Duties (a.k.a. *Segregation of Duties*): This pattern represents the ability to specify that two task duties of two different activities must be allocated to different persons. For example, the travel authorization form must be signed by anybody except the person that filled in the document, in order to prevent conflicts of interests (cf. Figure §6.8). This can be expressed using the following RAL expression in activity Sign Travel Authorization:

```
(NOT (IS PERSON RESPONSIBLE FOR ACTIVITY FillTravelAuthorization))
```

Case Handling: This pattern represents the ability to allocate a specific task duty to the same resource for *all* the activity instances of a BP instance. For example, a single person with role *Project's PhD Student* is responsible for undertaking all the activities of the trip management process, which can be expressed using the following RAL expression in all the activites of the process:

```
RAL expression for all the activities of the process:
(HAS ROLE ProjectsPhDStudent) AND
(IS PERSON RESPONSIBLE FOR ACTIVITY SubmitPaper)
```

Notice that the second part of the composition is not necessary for the first activity assigned the *Project's PhD Student* role (in our application scenario, activity *Submit Paper*).

Binding of Duties: This pattern represents the ability to specify that two task duties of two different activities must be allocated to the same person. For instance, the person that submits the paper is due to register at the conference, which can be expressed using the following RAL expression in activity Register at Conference:

```
IS PERSON RESPONSIBLE FOR ACTIVITY SubmitPaper
```

The variant of this assignment pattern named **Retain Familiar** (cf. Section §3.2.3) is not supported by RAL, as we consider this pattern related to the configuration of assignment *preferences*, which are out of the scope of this thesis.

Capability-based Allocation: This pattern represents the ability to offer or allocate instances of a task duty to resources based on their specific capabilities. For instance, instances of the *Sign Travel Authorization* task must be allocated to someone holding a degree.

```
RAL expression for activity Sign Travel Authorization:
HAS CAPABILITY Degree
```

History-based Allocation: This pattern represents the ability to offer or allocate activities to resources on the basis of their previous execution history. For instance, any participant in any instance of the process can check the answer received from the *Research Vice-chancellorship* (cf. Figure §6.9).

```
RAL expression for activity Check Response:
IS PERSON WHO HAS BEEN INVOLVED IN ANY ACTIVITY IN ANY PROCESS
INSTANCE
```

Position-based Allocation: This pattern, originally so-called *Organisational Allocation*, represents the ability to offer or allocate instances of a task to resources based on their positions within the organisation and their relations with other resources. For instance, the travel authorization form must be filled in by somebody that reports to Anthony (cf. Figure §6.6).

```
RAL expression for activity Fill Travel Authorization:
REPORTS TO Anthony
```

Commonality-based Allocation: This pattern represents the ability to allocate a task to a resource based on the characteristics it has in common with another resource. For instance, the *Check Response* task must be allocated to someone that plays exactly the same roles played by Daniel.

```
RAL expression for activity Check Response:
SHARES ALL ROLE WITH Daniel
```

Group-based Allocation: This pattern, originally so-called *Single Entity*, represents the ability to consider persons, groups or teams as single resources for allocation to a task duty. For instance, the *Make Reservations* task can be performed by anybody belonging to the THEOS organisational unit.

```
RAL expression for activity Make Reservations: HAS UNIT THEOS
```

Restricted Team Size: This pattern represents the ability to specify a minimum and/or maximum size of a group or team able to allocate a specific task duty. For instance, task *Make Reservations* can be performed by three clerks, at most. This pattern is not supported by RAL, since RAL does not deal with team work.

As shown by means of the examples provided, RAL supports eleven out of the twelve assignment patterns considered in this thesis (cf. Section §3.2.3). Besides, RAL could fit in some WRPs [106] that were excluded from the assignment patterns because they are not stuck to resource specification, although they are close to it. For instance, RAL expressions could be used to define restrictions when specifying the recipients for task delegation (WRP Delegation), or task re-allocation (WRPs 30-Stateful Reallocation and 31-Stateless Reallocation), provided that the BPMS used allowed the specification of such restrictions for these actions.

6.6 SUMMARY

In this chapter we have presented RAL, a language for resource specification in BPM. In particular, RAL DACH contains RAL Core and three extensions, namely RAL Data, RAL AC and RAL History. In general, RAL Core and the extensions can be used separately or altogether, and other extensions could be developed for the language. RAL DACH specification in EBNF is depicted in Language 6.5.

The RAL expressions of RAL DACH are *traceable* to the organisational model of the company where the language is used, as long as it complies with the organisational metamodel that is imported in RAL (cf. Section §6.2). This means that every concept that appear in a RAL expression refers to a single concept of the organisational model, e.g. in expression `HAS ROLE ProjectsPhDStudent IN UNIT THEOS` it is perfectly indicated that *ProjectsPhDStudent* is a role, and *THEOS* is an organisational unit. By tracing the concepts of the language with organisational concepts, the gap between BP models and organisational models that is present in some resource specification approaches (e.g. in BPMN 2.0 [94]), is bridged.

RAL has a syntax that is close to natural language, which increases its readability, and can be used to specify the performers of any task duty associated to a process activity. Indeed, task duties are considered in the specification of the language in the RAL AC and RAL History extensions. Furthermore, the wide variety of expressions that can be defined with RAL DACH, which supports eleven assignment patterns out of the twelve patterns used as evaluation framework for expressiveness, makes the language an *expressive* alternative to specify resources in BP models.

Language 6.5 RAL DACH specification in EBNF

```
1  Expr := PersonExpr                          | IsAssignmentExpr
2          | GroupResourceExpr                 | HierarchyExpr
3          | CommonalityExpr                   | NegativeExpr
4          | CapabilityExpr                    | CompoundExpr
5
6  PersonExpr := IS PersonConstraint
7
8  GroupResourceExpr := HAS (POSITION | UNIT) GroupResourceConstraint
9          | HAS ROLE GroupResourceConstraint [IN UNIT GroupResourceConstraint]
10
11 CommonalityExpr := SHARES Amount (POSITION | UNIT) WITH PersonConstraint
12         | SHARES Amount ROLE [IN UNIT GroupResourceConstraint] WITH PersonConstraint
13
14 CapabilityExpr := HAS CAPABILITY CapabilityConstraint
15
16 IsAssignmentExpr := IS ASSIGNMENT IN ACTIVITY activityID
17
18 HierarchyExpr := ReportExpr  | DelegateExpr
19
20 ReportExpr := Depth REPORTS TO PositionRef  | IS Depth REPORTED BY PositionRef
21
22 DelegateExpr := CAN DELEGATE WORK TO PositionRef  | CAN HAVE WORK DELEGATED BY PositionRef
23
24 NegativeExpr := NOT (PersonExpr | GroupResourceExpr | CommonalityExpr | CapabilityExpr)
25
26 CompoundExpr := (Expr) OR (Expr)  | (Expr) AND (Expr)
27
28 PersonConstraint := personName
29         | PERSON IN DATA FIELD dataObject.fieldID
30         | PERSON TaskDuty ACTIVITY activityID [HistoryExpr]
31         | PERSON WHO HAS BEEN TaskDuty ANY ACTIVITY IN BPHistoryExpr
32
33 TaskDuty := RESPONSIBLE FOR                  | CONSULTANT OF
34         | ACCOUNTABLE FOR                   | INFORMED ABOUT
35         | PROVIDING SUPPORT FOR              | INVOLVED IN
36
37 HistoryExpr := IN CURRENT INSTANCE     BPHistoryExpr := CURRENT PROCESS INSTANCE
38         | IN ANY INSTANCE                       | ANY PROCESS INSTANCE
39         | IN ANOTHER INSTANCE                   | ANOTHER PROCESS INSTANCE
40         | FROM startDate TO endDate             | A PROCESS INSTANCE BETWEEN startDate AND endDate
41
42 GroupResourceConstraint := groupResourceName  | IN DATA FIELD dataObject.fieldID
43
44 CapabilityConstraint := capabilityID  | CapabilityRestriction
45
46 PositionRef := POSITION GroupResourceConstraint  | PersonConstraint
47
48 Amount := SOME  |  ALL
49
50 Depth := DIRECTLY | λ
```

7

Flexible Resource Specification in Business Processes with RAL

"When a person acts without knowledge of what he thinks, feels, needs or wants, he does not yet have the option of choosing to act differently"

Clark Moustakas (1923–2012),
Psychologist

"Wealth is not about having a lot of money; it's about having a lot of options"

Clark Moustakas (1965–),
Comedian, actor and producer

Flexible resource specification involves offering the user the support required to let him/her select the binding strategy to use for the specification of resource in BPs, among the three options presented in Section §3.2.4. In this chapter, we present two resource specification approaches for two of the binding strategies, and we define a procedure that allows us to automatically move from one approach to the other. In Section §7.1, we detail of what our proposal for flexible resource specification consists. In Section §7.2, we present an all-in-one binding approach based on RAL and BPMN. Section §7.3 explains how to use the RACI matrices and RAL to implement the separate binding strategy, and in Section §7.4 we define transformation rules to automatically switch from the separate model to the all-in-one model presented in the previous sections. A summary of the contributions and the conclusions drawn from the work performed are presented in Section §7.5.

7.1 INTRODUCTION

In Section §3.2.4 we described the three binding strategies for resource specification that we identified, namely the all-in-one, split, and separate binding strategies. On top of that, we consider *binding flexibility* as the capability of allowing a user to choose the binding strategy to use for resource specification in an organisation, or for a specific BP. Thus, in order to provide a flexible proposal, approaches for several binding strategies have to be developed and offered to the organisation, so that it can select the one that best meets its needs for resource specification.

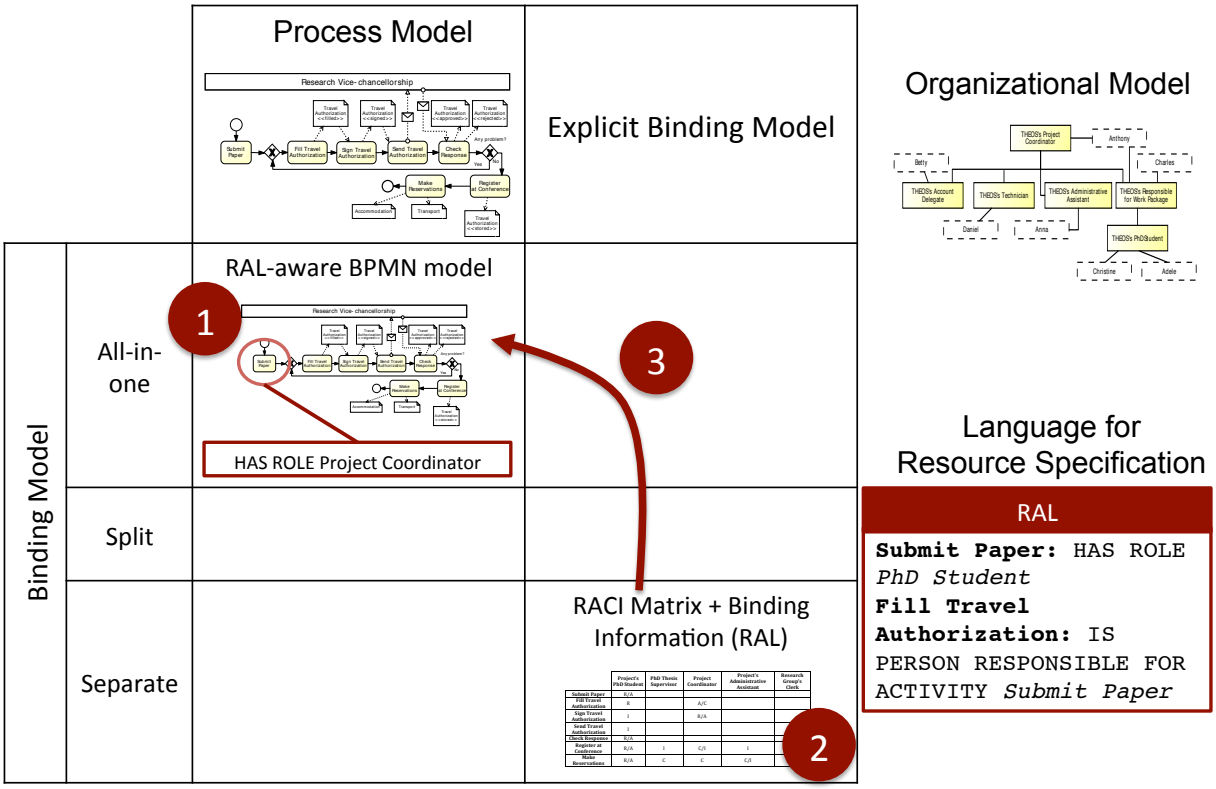


Figure 7.1: Our proposal for flexible resource specification

Figure §7.1 illustrates the contributions of this chapter. Specifically, we present an all-in-one approach by using RAL (cf. Chapter §6) with BPMN 2.0 [94] that generates RAL-aware BPMN models (point one in the figure). Furthermore, we introduce a separate binding approach (point two in the figure) in which the binding model is composed of the so-called RACI matrices [44], extended with what we have called *binding information*, which we specify with RAL. Finally, we provide an automated transformation from our separate binding approach to our all-in-one approach (point three in

the figure), that is, given a binding model based on RACI matrices and RAL, we automatically generate a RAL-aware BPMN model with all the information involved in the separate binding approach.

Although we use BPMN as process modelling language, we would like to emphasize that the approaches presented in this chapter are not constrained to BPMN. Instead, they could be applied to other BP modelling languages by adapting the details of the approaches to their features. Similarly, where we use RAL we could use other language for resource specification, as long as its expressiveness is similar to that provided by RAL.

7.2 All-in-One Binding Approach

Our approach for all-in-one resource specification is grounded on using RAL with the standard for process modelling. Thus, in this section we explain how RAL expressions can be used in BPMN models, that is, how RAL can be integrated into the the BP notation. First, it is necessary to understand how BPMN 2.0 [94] manages the resource perspective.

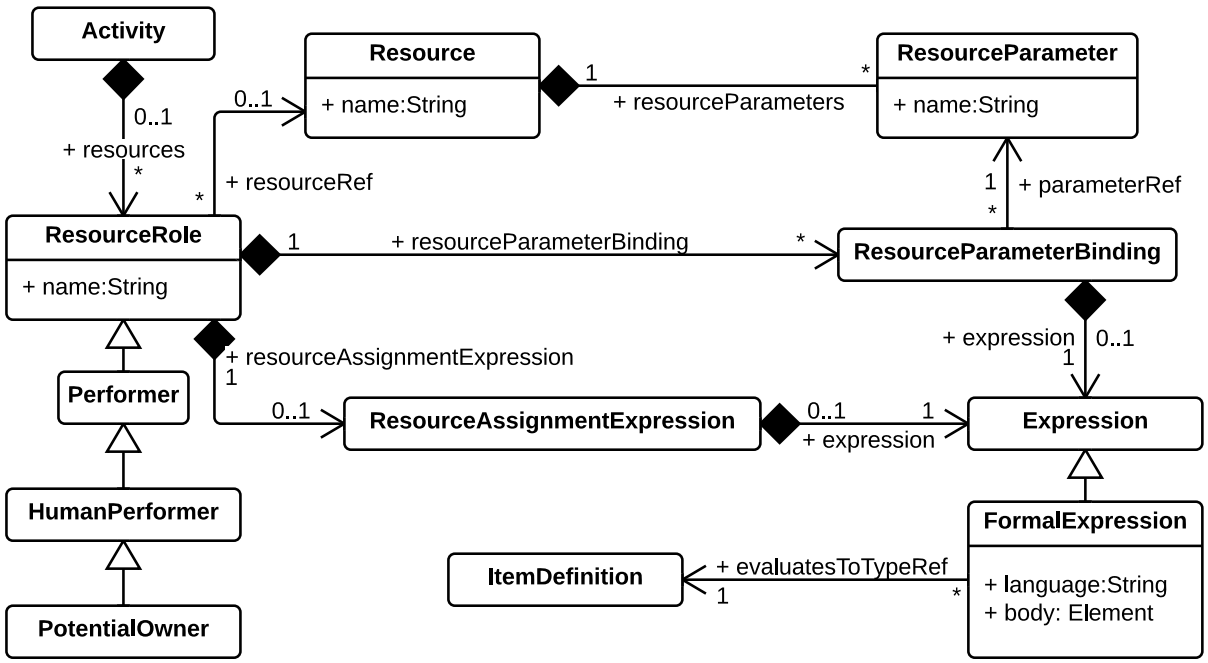


Figure 7.2: Excerpt of the BPMN 2.0 metamodel regarding resource specification

Figure §7.2 shows the excerpt of the BPMN 2.0 metamodel that addresses the assignment of resources to BP activities [94]. Each *Activity* can have zero or more instances of *ResourceRole* associated. BPMN uses class *PotentialOwner* to refer to the resources that are allowed to execute the task. Following the criteria defined in Section §3.2.2, we interpret it as support for the Responsible and Accountable task duties for an activity, being both duties associated to the same resource. Then, the metamodel shows two alternatives to assign the potential owners to the activities.

The first one consist of performing queries over a specific *Resource*. BPMN allows the definition of *Resources*, but as BPMN is not traceable to an organisational model, the term *Resource* has a general meaning. Indeed, as stated in [94], "a *Resource* can be *Human Resources* as well as any other resource assigned to activities during process execution time. The definition of a resource is *abstract* [...]". Thus, a resource can be anything (a person, a role, an organisation, *etcetera*) and there is no way to figure out the type of resource we are referring to. Using this option, we place the name of the resource that we want to assign in class *Resource*, e.g. Project's PhD Student, which is a role in our application scenario, as depicted in Figure §6.3. Once selected a resource, we can reduce the set of potential owners for the activity by establishing filtering conditions, e.g. we can indicate a specific country or age, a certain number of years of experience, and the like. These conditions are defined by means of class *ResourceParameterBinding*. Specifically, this class contains an *Expression* that allows us to freely define properties that must be fulfilled by the potential owners. Notice that according to BPMN specification, Class *ResourceParameterBinding* can only be used if in conjunction with *Resource* [94].

BPMN proposes by default the use of XPath[1] to define the *Expressions*, a language to query *EXtensible Markup Language (XML)*[2] mainly based on performing XSL transformations[3]. However, XPath present several shortcomings:

- Even though it is is a standard, XPath is barely supported by current BPMSs, which usually implement resource assignments in an ad-hoc fashion.
- XPath is not conceived specifically for resource specification, so it is difficult to use it to define resource assignment expressions.
- Derived from the previous point, XPath does not allow to express some resource-

[1] http://www.w3.org/TR/xpath/
[2] http://www.w3.org/TR/1998/REC-xml-19980210
[3] http://www.w3.org/TR/xslt

related information such as SoD constraints or capability-based assignments.

The second approach for resource specification offered by BPMN, consists of allowing free resource assignments, not constrained to a specific *Resources*. It is done by means of class *ResourceAssignmentExpression*, using XPath by default, too. The advantage of this alternative is that no constraints are set beforehand. However, the difficulty of specifying resource assignments with XPath remains.

Note the two methods are incompatible with each other, i.e., the selection of potential owners is made either with the mechanism based on filtering people on the basis of a *Resource*, or with a *ResourceAssignmentExpression*.

```
<userTask name="Approve task Submit Paper" id="id-23">
<potentialOwner>
<resourceAssignmentExpression>
<formalExpression language="RAL">HAS ROLE PhDStudent</formalExpression>
</resourceAssignmentExpression>
</potentialOwner>
</userTask>
```

Figure 7.3: RAL with BPMN 2.0. Example

Our all-in-one approach for resource binding, constitutes an alternative to the use of XPath in the second resource assignment method described above. In our case, RAL assignments would be added like shown in Figure §7.3, that is, in class *FormalExpression* attribute *language* takes value *RAL* and the RAL expression is set in attribute *body*. Thus, the use of RAL with BPMN is straightforward, and we can make use of all RAL's advantages for resource specification without modifying the BPMN metamodel.

Regarding the rest of task duties (Support, Consulted, Informed), the way to add them to BPMN is by adding one new type of *ResourceRole* for each of them. This extension is implicitly supported by BPMN [94], despite not being explicitly mentioned in the BPMN specification.

Table §7.1 presents a summary of the support provided by the main versions of BPMN as for the assignment patterns described in Section §3.2.3. In particular, BPMN 1.0, BPMN 2.0, and BPMN 2.0 with RAL have been evaluated. The patterns supported are represented with symbol ✓and those unsupported are left in blank. The support that RAL provides for the assignment patterns was described in Section §6.5.

Assignment Pattern	BPMN 1.0	BPMN 2.0	BPMN 2.0 with RAL
Direct Allocation	✓	✓	✓
Role-Based Allocation	✓	✓	✓
Deferred Allocation			✓
Separation of Duties (SoD)			✓
Case Handling			✓
Binding of Duties (BoD)			✓
Capability-based Allocation		✓	✓
History-based Allocation			✓
Position-based Allocation		✓	✓
Commonality-based Allocation			✓
Group-based Allocation		✓	✓
Restricted Team Size			

Table 7.1: Assignment patterns supported by BPMN/RAL

7.3 Separate Binding Approach

Languages like RAL can be used to complement resource specification approaches that use a separate binding strategy. The so-called RACI matrices constitute an example of such type of approaches (cf. Sections §3.2.2 and §3.3). They provide advance resource assignment capabilities by associating a number of task duties (called *RACI roles* in RACI[4]) to the process activities [13, 24, 118]. Specifically, the variant of the RACI matrices called RASCI copes with the five task duties considered in this thesis, namely: Responsible, Accountable, Support, Consulted, Informed. However, there are some drawbacks with regard to the type of assignments allowed for the task duties. Let us explain it with an example.

Let us suppose that the RASCI matrix shown in Table §7.2 has been defined considering a company with the organisational model depicted in Figure §6.3 and the BP shown in Figure §6.4. Please, note that although the application scenario is the same as the one we used when we introduced RAL in Chapter §6, the resource assignments defined in this chapter are independent from those used in Chapter §6. Then, with the information from the organisational model, the BP model and the RACI matrix, the process can be described as follows. "The procedure to manage the trip to attend a conference starts when a PhD student submits to a conference the final version of a paper that has been accepted for publication. Then, *that* student fills in an authorization request to attend and present the paper at the conference. The coordinator

[4]In this chapter, we will use terms *task duty* and *RACI role* indistinctly.

Activity \ Role	Project's PhD Student	PhD Thesis Supervisor	Project Coordinator	Project's Administrative Assistant	Research Group's Clerk
Submit Paper	R/A				
Fill Travel Authorization	R		A/C		
Sign Travel Authorization	I		R/A		
Send Travel Authorization	I				R/A
Check Response	R/A				
Register at Conference	R/A	I	C/I	I	
Make Reservations	R/A	C	C	C/I	S

Table 7.2: RASCI matrix for the process at pool *ISA Research Group*

of *the project that will finance the trip expenses* must sign the authorization and inform the student when it is done. The clerk of *the research group the PhD student belongs to* is in charge of delivering the form for approval. This form will be sent back by the Vice-chancellorship some time later, together with an approval or refection notification, which might be checked by *the PhD student*. In absence of problems, *the student* must register at the conference and inform *his/her* PhD thesis's supervisor, as well as *the project coordinator* and *the* administrative assistant of *the project*. Finally, *the PhD student* books the tickets needed, assisted by the clerk of *his/her* research group, if required."

The description of the BP above is different from the one we provided in Section §6.3 due to two main reasons: (i) we already have at hand information about the BP resource perspective, as the RACI matrix in Table §7.2 informs about the roles that can be assigned to the process activities; (ii) with that information, it is possible to infer knowledge that is not explicitly specified in the matrix, but which responds to the natural way of proceeding in BPs. For example, we can assume that the project coordinator in charge of signing the travel authorization is the coordinator of the project that provides the funds for the trip, and not any coordinator of any project that is developed in the company.

In general, the RACI matrices bring information about the BP resource perspective, but the specific details or conditions that the individuals have to fulfill in order to become potential performers of a task duty, remain unspecified. This is a shortcoming that reduces the expressiveness of the RACI approach[5]. In order to overcome this

[5]RACI only supports Role-based Allocation when the columns of the matrix represent roles, Position-

drawback, the missing information required to complements the resource specification provided by RACI matrices and, thus, provide more accurate resource specification, has to be somehow defined. We have called this information *binding information*, and it can be related to the organisational unit context where the role is used, or to any other restriction that may be necessary to indicate:

- *Organizational unit context.* Indicating only the organisational role for a RASCI role is usually insufficient, since it does not limit the context of application. Let us show an example. According to the RASCI matrix in Table §7.2, role *Project Coordinator* is responsible for activity *Sign Travel Authorization*. However, a project coordinator can sign only those forms related to the project(s) he/she coordinates, so *not any* project coordinator can perform this task in *any* process instance. Therefore, it is necessary to indicate either directly the concrete data required (e.g. name of the project we refer to in the current process instance), or *where* this information can be found, e.g. in our BP the name of the project is likely to appear in the *Travel Authorization* form filled in by the student (activity *Fill Travel Authorization* in Figure §6.4), since it is there where the funding source is indicated.
- *Additional restrictions.* Further information may be necessary in order to constrain the set of people that can be assigned certain RASCI role. For example, sometimes it is essential that two activities of the same BP be performed not only by the same organisational role, but by the same person, i.e. BoD. For example, in the scenario at hand we have assumed that the PhD student that submits the paper and the one that fills in the travel authorization form, are the same persons. Other times, exactly the opposite may be necessary, i.e. SoD, in order to avoid conflicts of interests between individuals. Restrictions concerning specific skills required to perform a certain task are also commonly needed. All these additional restrictions might be taken into account for resource allocation at run time.

Notice that the binding information must be provided at cell level in the RACI matrix, that is, for each task duty associated to each BP activity.

7.3.1 Specification of Binding Information with RAL

As foreseen above, RAL can be used to specify the binding information required to complement RACI matrices. Let us see some examples:

based Allocation in case of using positions, Group-based Allocation if the type of resource used are organisational uni, and so on.

- *Organizational unit context.* To state that one task duty has to be performed by a role that is related to *a specific* organisational unit, we could use RAL expression `HAS UNIT UnitName`, or expression `HAS UNIT IN DATA FIELD DataObj.DataField` to complement the role-based information provided in the RACI matrix.

- *Additional restrictions.* RAL offers expressions to specify many types of restrictions, such as expression `IS PERSON RESPONSIBLE FOR ACTIVITY ActName` to indicate BoD constraints with respect to the performer of another activity; or expression `NOT(IS PERSON ACCOUNTABLE FOR ACTIVITY ActName)` to indicate SoD with respect to the resource accountable for another activity. Expressions such as `SHARES SOME ROLE WITH PersonName` and `HAS CAPABILITY CapabilityName` allow the specification of other kinds of restrictions. We refer the reader to Chapter §6 for a detailed description of RAL language.

7.3.2 RASCI Metamodel with Binding Information

Figure §7.4 shows a possible metamodel of a RASCI matrix with binding information, taking all the aforementioned aspects into consideration.

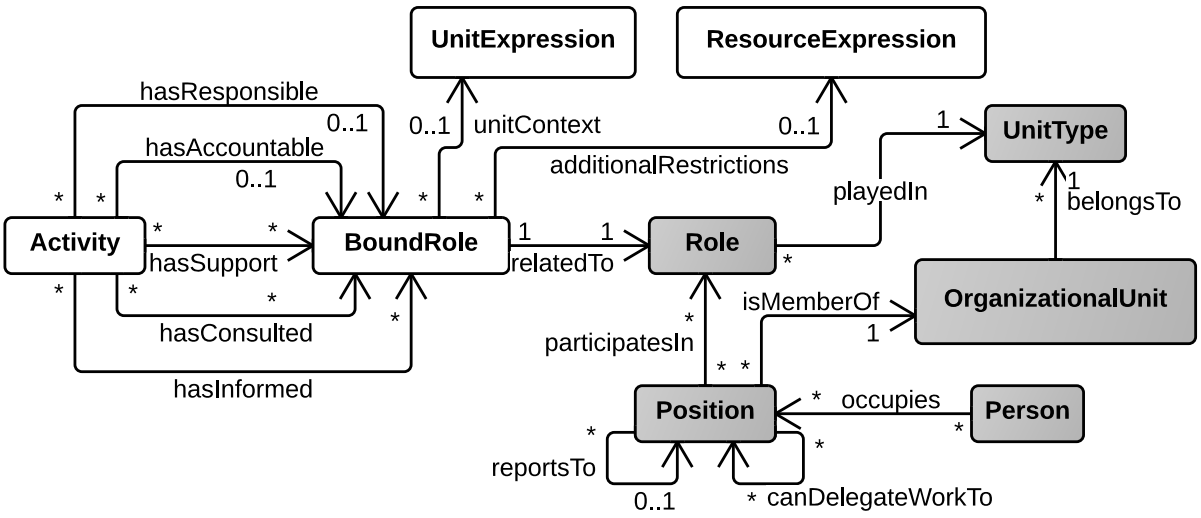


Figure 7.4: RASCI metamodel with binding information

- Class *Activity* represents the activities of the BP the RASCI matrix is associated to, and in which we aim to insert the responsibility-related information necessary to make it work according to the matrix.

- Five relations between *Activity* and *BoundRole* represent the five RASCI roles to be distributed among the members of the organisation. The role is bound because it may have binding information associated. The expressions to specify the organisational unit context and any other additional restriction can be defined in classes *UnitExpression* and *ResourceExpression*, respectively.

  We have added the following conditions between *Activity* and *BoundRole* in order to define some existence relations between RASCI roles. We use *Object Constraint Language (OCL)*[6] to specify the following invariants:

  – When there is not a resource responsible for an activity (e.g. an automatic task executed directly by the system), the other RASCI roles cannot exist, except RASCI role I. We exclude the information function (I) because there may be automatic activities in the process consisting of a notification message automatically sent by the system, but whose destination can be a resource indicated in the RASCI matrix.

  ```
  context Activity inv:
  if self.hasResponsible->isEmpty()
  then
  self.hasAccountable->isEmpty() and
  self.hasSupport->isEmpty() and
  self.hasConsulted->isEmpty()
  endif
  ```

  – When RASCI role R is in, then there must be an accountable, since this role is mandatory according to RACI definition[7].

  ```
  context Activity inv:
  if not(self.hasResponsible->isEmpty())
  then not(selt.hasAccountable->isEmpty())
  endif
  ```

- The classes in gray in the figure represent the part of the organisational metamodel described by Russell et al. [105] we have relied on to build the structure of an organisation, which was also used in RAL (cf. Chapter §6). In particular, each *BoundRole* is associated to a *Role* of the organisational structure of the company. However, as aforementioned, a person has a role in the context of an organisational unit (e.g. coordinator of a certain project, or research assistant in a specific

[6] http://www.omg.org/spec/OCL/2.0/

[7] The lack of A in the table is interpreted as R and A being assigned to the same role.

research group). This relation is modelled by means of class *Position*. A position, thus, represents a collection of roles in one specific organisational unit.

Let us take as example *Activity Sign Travel Authorization* of our use case (cf. BP model in Figure §6.4 and RASCI matrix in Table §7.2) to instantiate the RASCI metamodel shown in Figure §7.4. The organisational roles that participate in this activity (i.e. *Project's PhD Student* and *Project Coordinator*) fit in class *Role*. As aforementioned, every role is related to an organisational unit. In this case, it is a *project* (class *UnitType*) called *THEOS* (class *OrganizationalUnit*). There is a positional hierarchy for each organisational unit. For project THEOS it is shown in Figure §6.3 and was described in Section §6.1.

The rest of classes of the RASCI metamodel (i.e. *BoundRole*, *UnitExpression* and *ResourceExpression*) are specified at cell level. For RASCI roles R and A, *BoundRole* contains the assignment to role *Project Coordinator*, together with a *UnitExpression* stating that the name of the project can be found in file *Travel Authorization* (handled in the process) during execution. For RASCI role I, *BoundRole* is role *Project's PhD Student* plus a *ResourceExpression* indicating that it has to be the same person who performed activity *Submit Paper*.

7.4 From Separate to All-in-One Binding

As aforementioned, it may be convenient to have a mechanism to automatically switch between different binding strategies. We next introduce a procedure to model the information contained in a RASCI matrix extended with binding information, into a BPMN diagram that initially had no information related to resources. That is, we are changing a separated model into an all-in-one model, specifically into a *RASCI-aware BPMN model*. Notice that this is beneficial not only because it allows us to manage the information separately and then execute them jointly (as stated in Section §7.1), but also because we manage to model all the task duties in BPMN, something that the standard originally does not provide.

The approach introduced next is based on a set of generic transformations that can be automated. The BP model generated as a result is BPMN-compliant and has the required information to be used in existing BPMSs that support BPMN model execution.

First, the BP activities that appear in the RASCI matrix are changed into a subprocess with the name of the activity. All the RASCI information will thus be contained

in the sub-process. We will sometimes refer to such a sub-process as *RASCI sub-process*. Within it, for each RASCI role it is necessary to indicate:

- The control flow elements required, with the *name convention pattern* we will use to make the transformation as automatic and generic as possible.
- The proper resource assignment expression associated to each new task. This expression comes from class *BoundRole* of the RASCI metamodel (cf. Figure §7.4). Using RAL, the expression there is a *UnitExpression* is

  ```
  (HAS ROLE Role IN UNIT Unit_In_UnitExpression) AND
  (ResourceExpression),
  ```

  Otherwise, it is

  ```
  (HAS ROLE Role) AND (ResourceExpression)
  ```

We assume that there is only *one person responsible and one accountable* for each activity (cf. metamodel in Figure §7.4). Furthermore, the approval action (RASCI role A) takes place after the completion of the work developed for the activity, and only then the notification action (RASCI role I) can be performed. We could opt for a different order of the task duties or, even, for allowing them at different phases of the activity lifecycle (for instance, to inform also before the start of the task or during execution). However, in the latter case, changes should be made in the RASCI matrix, and it is out of the scope of the thesis.

The overview of a RASCI sub-process is depicted in Figure §7.5. BPMN groups define the process fragments related to RASCI roles. In case a RASCI role does not participate in the activity, the corresponding process fragment will be omitted in the sub-process. If, on the contrary, there are several roles performing a RASCI role, the associated process fragment will be added for everyone of them. We will use activity *Register at Conference* of our RASCI matrix (c.f. Table §7.2) to explain the transformations. Figure §7.6 shows the RASCI sub-process for it.

Responsible (R) This is the only RASCI role whose resource assignment expression is associated to the RASCI sub-process itself, i.e. for activity *Register at Conference*, the new sub-process has the following RAL expression: `(HAS ROLE ProjectsPhDStudent) AND (IS PERSON RESPONSIBLE FOR ACTIVITY SubmitPaper)`.

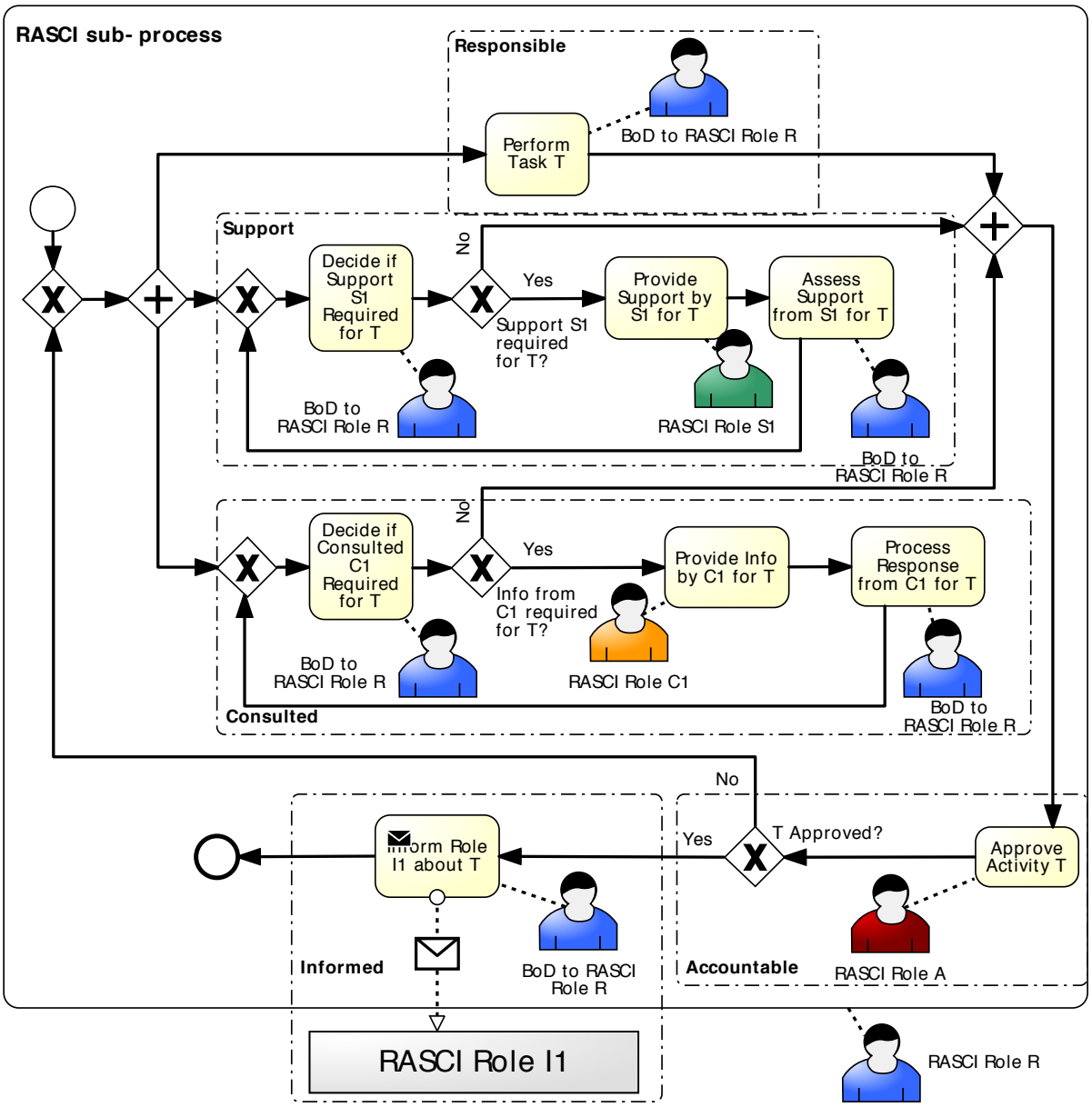


Figure 7.5: Overview of a RASCI sub-process

Nonetheless, task *Perform Task ActivityName* is introduced in the RASCI sub-process to represent the actual work to be completed for the activity. This task is directly assigned to the performer of the sub-process, i.e. the RAL expression for task *Perform Task Register at Conference* is `IS PERSON RESPONSIBLE FOR ACTIVITY RegisterAtConference`. This allows every element within the subprocess to make reference to the performer of the activity being sure that there is already a potential performer allocated[8]. Note that

[8]According to BPMN [94], the allocation related to the sub-process is made before starting the activ-

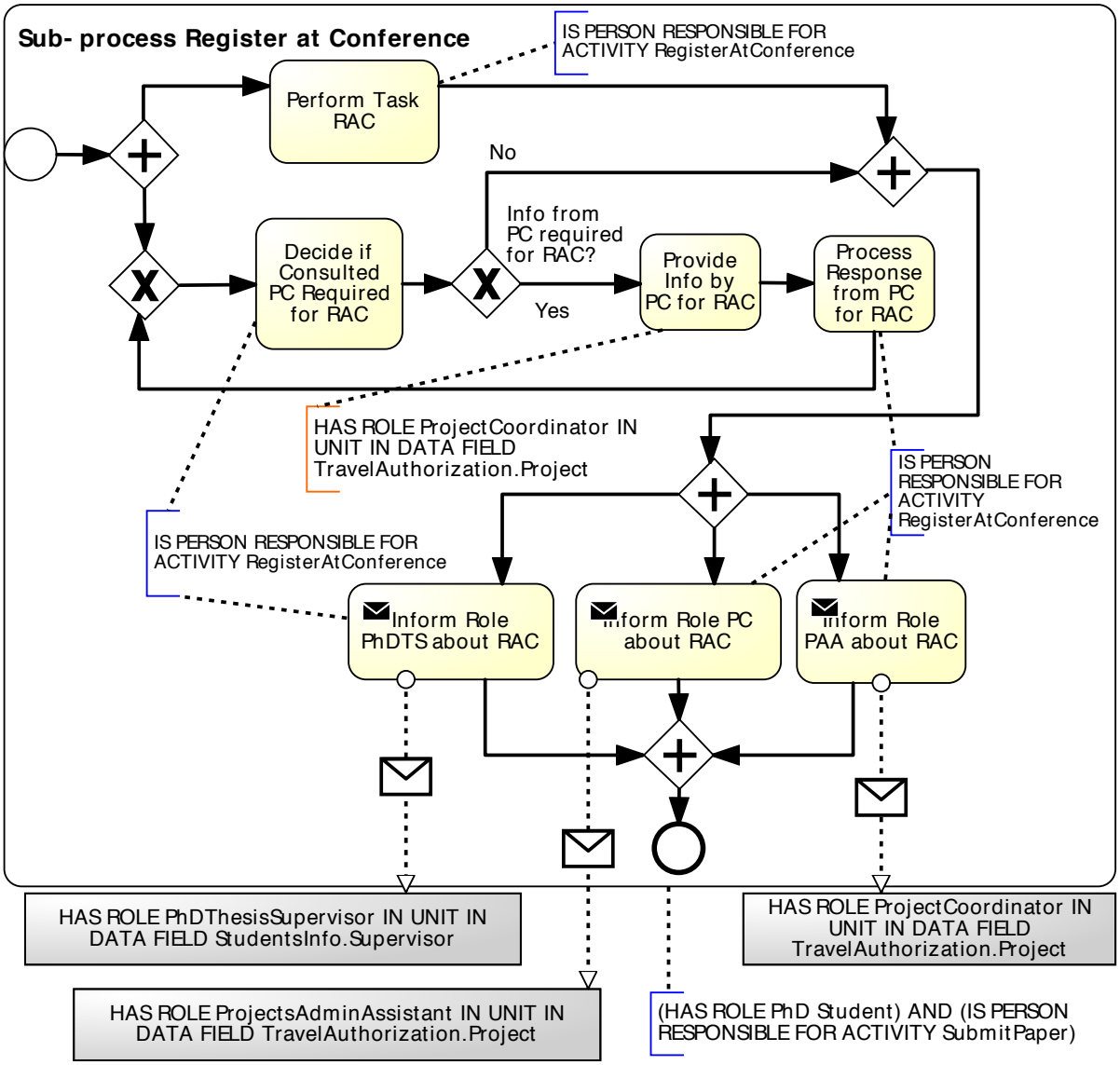


Figure 7.6: RASCI sub-process for activity *Register at Conference*

if the activity itself were a sub-process, then *Perform Task Register at Conference* would be *that* sub-process.

Accountable (A) To model this RASCI role we insert a new task into the RASCI sub-process named *Approve Activity ActivityName*, in charge of approving the work developed in the activity at hand. Moreover, we have to add the control flow required to go back to the beginning of the sub-process in case the activity was not approved,

ities that compose it.

by means of an *XOR join* gateway. The assignment expression of RASCI role A will be assigned to the new task.

This process fragment can be omitted only if R and A are assigned to the same organisational role in the RASCI matrix, and the binding information for A consists of a BoD with respect to R (i.e. BoD between A and R). For size reasons in Figure §7.6, we assume this is satisfied in our example activity. Note that if the previous condition is met and no other RASCI role participates in the activity, then the whole RASCI subprocess can be omitted and, thus, the result of the transformation is the same initial activity with the resource assignment expression corresponding to R.

Support (S) Inserting this RASCI role is not as straightforward for several reasons: (i) support is not mandatory. It is a decision of the person in charge of the task whether support is required to complete the work; (ii) it is said nowhere that support cannot be requested more than once to the same organisational role associated to RASCI role S in the matrix; (iii) it is inherent to term support that the work performed by the person "external" to the task must be evaluated by the resource in charge of the task (R) in order to decide whether the goal of the support has been achieved and/or whether more support is required; and (iv) it is not evident at which moment in task execution, support can be requested. In this sense, we have to make some decisions. So, we propose the control flow structure depicted in Figure §7.5 for RASCI role S, basically composed of tasks *Decide if Support Role Required for ActivityName*, *Provide Support by Role for ActivityName* and *Assess Support from Role for ActivityName*, and a couple of XOR gateways. The task targeted at providing the required support is assigned to the organisational role performing RASCI role S in the matrix with the appropriate resource assignment expression. The rest of new tasks belong to the person that is responsible for the activity. Thus, the resource assignment expression for these tasks is a BoD constraint in RAL: `IS PERSON RESPONSIBLE FOR ACTIVITY ActivityName`. According to our RASCI matrix, activity *Register at Conference* does not need support.

Consulted (C) The translation of this RASCI role into BPMN language is very similar to RASCI role S. As depicted in the figure, the structure introduced in the subprocess is the same, as well as the considerations to be made. The only difference between the application of the two RASCI roles is in the name of the tasks involved, and in the semantics of tasks *Provide Support by Role for ActivityName* and *Provide Info by Role for ActivityName*. The person in charge of the latter is not so much compromised with the global activity, as his/her involvement is limited to providing certain informa-

tion. For activity *Register at Conference*, the RAL expression associated to the task performed by the project coordinator could be `HAS ROLE ProjectCoordinator IN UNIT IN DATA FIELD TravelAuthorization.Project`. For the rest of tasks of this process fragment, the associated RAL expression is `IS PERSON RESPONSIBLE FOR ACTIVITY RegisterAtConference` (cf. Figure §7.6).

Informed (I) This RASCI role has a very special difference with respect to the others. The organisational role indicated in the matrix is the target person of the notification action, not the performer like in the rest of cases. The problem is that in BPMN we do not have a way to specify the resource "affected" by a task. However, there is a mechanism based on message interchange to communicate information to people working on other processes. The key point here is whether the informed person can be considered an external participant or not. In case that role does not participate in any other activity of the BP, it is undoubtedly somebody external to the process. Thus, we could use messages to send the notification. Otherwise, that person may have his own assigned tasks in the process, and we do not have a way to notify something to that person without interrupting his work flow. We believe that, given the RACI definition for I, it is reasonable to consider it an external participant of the process under any circumstances, due to the absence of collaboration from his part. Independently of his responsibilities with respect to other BP activities, for *that* activity in question he is a target, not an executor. Therefore, we will introduce task *Inform Role about ActivityName* to represent an activity that sends a message to a collapsed pool representing RASCI role I (cf. Figure §7.5). This task is assigned to the person in charge of the main activity, like in RASCI roles S and C. For activity *Register at Conference*, there are three informed people. The control flow and the RAL expressions for them are directly shown in Figure §7.6.

Following these rules we can convert the initial BP model into another one with the information necessary to implement RASCI in our organisation. Furthermore, the resulting model is very clean from the visualization perspective, in the sense that it is very similar to the initial one. So, the overall understandability of the initial process is maintained. The real complexity related to the RASCI information is found only when the RASCI sub-processes are opened.

7.5 SUMMARY

In this chapter we have introduced an approach for all-in-one binding using RAL with BPMN, we have presented a proposal for separate binding extending the RACI matrices with binding information expressed with RAL, and we have defined a set of transformations to automatically shift from the latter model to the former one. Therefore, we have introduced a *flexible* binding approach that, integrated in a BPMS, would have several advantages, to be named:

- Several binding strategies can be used as required within the same organisation. For example, for those processes in which we need only to manage the resources Responsible and Accountable for the activities, RAL could be used directly with BPMN (all-in-one strategy). When other task duties are required, the company could opt for using the separate binding approach we provide.
- Even for a single BP model, the organisational could model the resource perspective separately and treat it together with the control and the data flows at any moment, by performing the transformations required. Therefore, there is a decoupling of BP management and resource management at design time, but they can be automatically mixed together to be executed in combination at run time. Furthermore, the appearance of the resulting BPMN model is very similar of the initial resource-unaware BP model, so readability is not a problem at first sight.

As stated in Section §7.1, other BP modelling notation and other resource specification language could be used similarly to how we have done with BPMN and RAL, as long as they have the features required to deal with all the aspects involved in the approaches.

8

RAL Formal Semantics

"Do you wish me a good morning, or mean that it is a good morning whether I want it or not; or that you feel good this morning; or that it is a morning to be good on?"

J.R.R. Tolkien (1892–1973),
Writer, poet, philologist and university professor

The goal of this chapter is to introduce our approach to endow RAL with a formal semantics that allows giving a precise meaning to the assignment expressions defined with the language, and enables the automation of the analysis of the BP resource perspective. In Section §8.1 we justify the formalization of RAL semantics and summarize the procedure we have followed to define it in Description Logics (DLs). In Section §8.2 we explain how to map into DLs the elements related to the organisational metamodel imported by RAL. In Section §8.3 we describe the similar mapping applied to the BPM metamodel used by RAL. In Section §8.4 we define the mapping to DLs of each RAL expression, grouped by RAL Core and each RAL extension. We accompany the explanations with examples based on the RAL expressions used as example throughout Chapter §6. This chapter concludes in Section §8.5 with a brief summary of its content.

8.1 INTRODUCTION

The primary objective of formalizing RAL is to establish a sound basis for a sophisticated automated support. Therefore, according to the formalization principles defined by Hofstede and Proper [74], the style and target domain should be chosen accordingly (*Primary Goal Principle*). In our case, we propose a semantic mapping to DLs [87] to formalize RAL. DLs is a decidable subset of *First Order Logic (FOL)* [76] that serves primarily for formal description of *concepts*, *roles* (relations between the concepts) and *individuals* (instances of the concepts)[1]. A brief introduction to DLs and their use can be found in Appendix §C.

DLs have been chosen because of two reasons. First, RAL expressions can be seen as a way to specify a subset of the people of an organisation by giving a set of constraints that they must satisfy (e.g., *HAS ROLE PhD Student*). This way of approaching RAL expressions fits nicely into the way DLs express their concepts and, hence, they provide a very natural way to describe the problem, which allows following the *Semantics Priority Principle* while keeping close to the RAL metamodel. Furthermore, this makes it easier to avoid unnecessary representational choices as suggested by the *Conceptualization Principle*. As a consequence, we can define RAL Core semantics and, then, extend it to RAL Data, RAL AC and RAL History without modifying the essence of RAL Core semantics.

The second reason for choosing DLs is that they are best known for providing a logical formalism for ontologies and the semantic web. As a matter of fact, the semantics of the W3C recommendation OWL 2 [88] to express ontologies for the semantic Web is defined in DLs. A consequence is that there is a plethora of off-the-shelf DLs reasoners that can be used to automatically analyse RAL expressions efficiently and, hence, to automatically infer information from them.

Before going into details about the mapping of RAL into a DLs, some basic DL-related concepts must be introduced. A DL-based *Knowledge Base (KB)* is a finite set of terminological and assertional sentences. It has 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 ontology. Thus, every model related to RAL has to be mapped into DL elements either in the TBox or in the ABox. To this regard, although there is a sig-

[1] *Web Ontology Language (OWL)* terms *classes*, *properties* and *objects* will also be used to refer to DL terms *concepts*, *roles* and *individuals*, respectively.

nificant number of concepts in the problem domain, we have tried to keep the number of concepts in the formalization as small as possible as suggested by the *Orthogonality Principle*, which in most cases translates into a one-to-one relationship between semantic concepts and domain concepts. In particular, regarding the organisational and the BP model and metamodel, the mapping is performed as depicted in Figure §8.1:

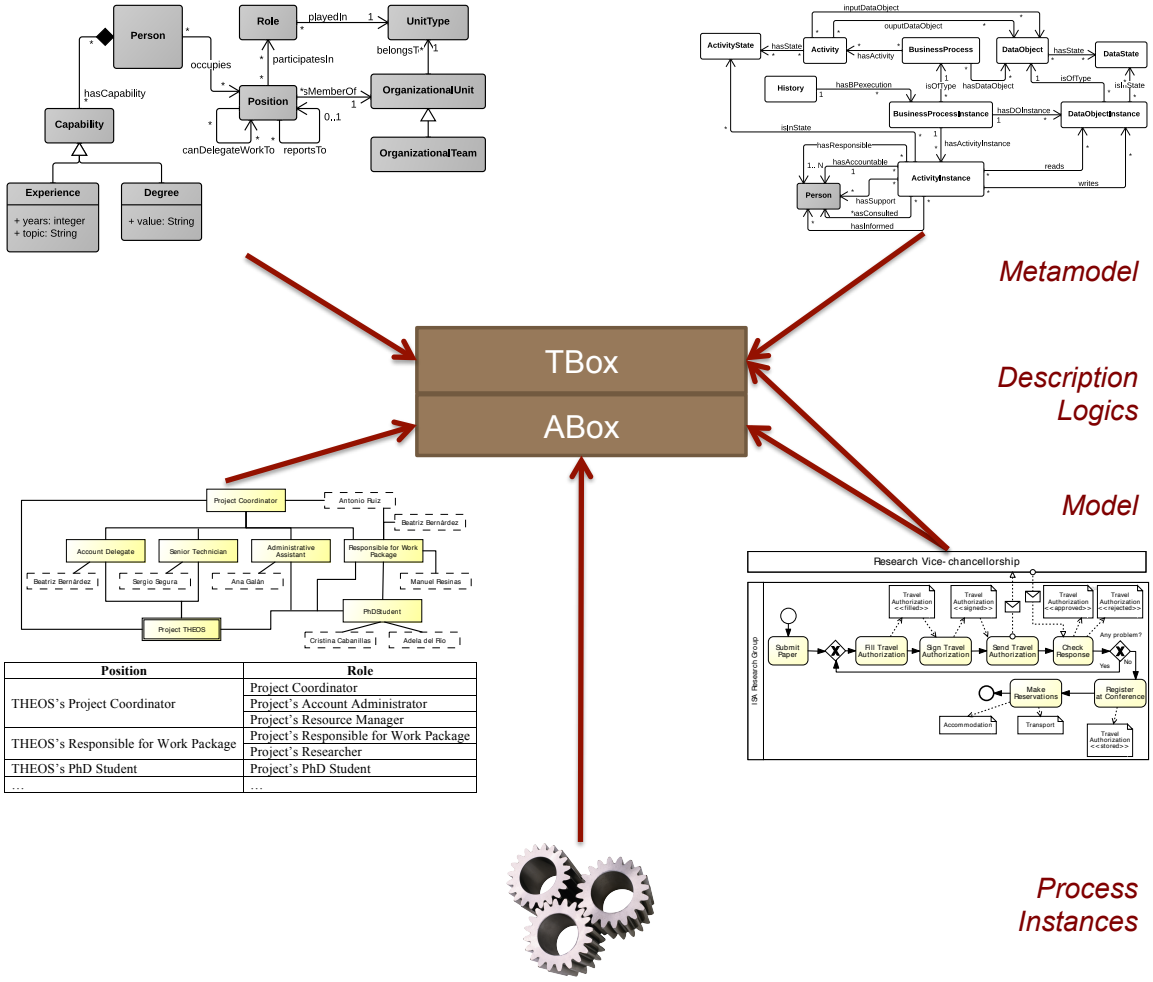
- The elements included in the metamodels have to be mapped into elements of the TBox, so that classes of the metamodel are mapped into *concepts* of the TBox, and relations in the metamodel are mapped into *properties* of the TBox, together with the required configuration, e.g. cardinality restrictions.
- The instances of the classes of the metamodel, i.e., the elements of the models, are mapped into *individual assertions* in the ABox of the DL-based KB, together with the required configuration, like before. Furthermore, for the BP model we add a previous step that refines the mapping of the metamodel by grouping the instances of the same type into new concepts that are added to the TBox.
- The process instances generated during BP execution are mapped into the ABox as instances of the concepts introduced in the aforementioned refinement, adapting the configuration of the KB.

Having the information from the organisational model and de BP model in the KB, RAL expressions can be mapped into DL expressions that represent subconcepts of concept *Person* mapped from the organisational metamodel. This mapping is the basis for the definition of several relationships that are introduced in Chapter §9 to make the formulation of analysis operations easier as suggested by the *Bottom Up Principle*.

In the following three sections we describe every mapping in detail, divided into three groups: the mapping required for the organisational elements that participate in RAL, the similar mapping applied to the BPM elements, and the mapping of the RAL expression into DL queries. For all the DL expressions, a syntax commonly used for DLs [18] is utilised.

8.2 Mapping the Organisational Information

As stated above and depicted in Figure §8.1, the first step to model all the information related to RAL in a DL-based KB, is to map the elements of the metamodels



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

Figure 8.1: Mapping of RAL into DLs

imported in RAL metamodel, together with the elements required for their instantiation, into DL elements. Regarding the organisational metamodel, one *class* or *concept* is added to the TBox of the KB for each and every class included in the metamodel, which (depicted in gray in Figure §6.2). We keep the same names for the sake of understanding. Hierarchies are also included in the TBox by using the *subclassOf* axiom that DL provides. Data properties are added to the classes that contain attributes. For example, the 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, with type *xsd:int*, and topic, with the *xsd:string* type.

Regarding the explicit relations between the classes of the metamodel, all the relations between classes of the metamodel shown in Figure §6.2 are mapped into *properties* of the TBox, and the corresponding *cardinality restrictions* are configured. That is, properties *hasCapability*, *occupies*, *reportsTo*, *participatesIn* and the like, are added to

the TBox, as depicted in Table §8.1. The relations in hierarchies in the metamodel are mapped as *sub-properties* of the corresponding properties. Besides, *inverse properties* have been created for all the properties in order to ease the use of the KB. For example, properties *isOccupiedBy* and *isReportedBy* have been defined as the inverse properties of *occupies* and *reportsTo*, respectively. *Cardinality* must be configured for all the properties according to the relations in the organisational metamodel. For instance, to configure that "a Position is member of exactly one OrganizationalUnit", axiom $Position \sqsubseteq = 1\ isMemberOf.OrganizationalUnit$ is added. In case of cardinality less or equal to 1, an alternative is to set the property type to *functional*. The inverse properties and the information about cardinality has not been included in Table §8.1 for the sake of readability.

Object property	Sub-property of	From	To	Property Type
occupies		Person	Position	
participatesIn		Position	Role	
isMemberOf		Position	OrganizationalUnit	
reportsTo	extendedReportsTo	Position	Position	Functional
extendedReportsTo				Transitive
canDelegateWorkTo		Position	Position	Transitive
hasCapability		Person	Capability	
hasDegree	hasCapability			
hasExperience	hasCapability			

Table 8.1: Properties in the TBox related to organisational information

As can be seen in the table, the relations derived from RAL *HierarchyExprs* (cf. Language 6.1) have received a special treatment. Specifically, a *super-property extendedReportsTo* has been created to make the property *transitive*, as according to the metamodel the *reports to* relation can be propagated from position to position of an organisation in order to represent lines-of-reporting. Thus, we aim at enabling stating assignments such as "activity *Sign Travel Authorization* can only be performed by a person that is reported by somebody reported by a *PhDStudent*". Similarly, property *extendedCanDelegateWorkTo* is transitive. However, there is not a functional variant of this property because the corresponding relation in the organisational metamodel is N:M.

Once the metamodel is mapped, it is possible to map specific organisational models into the KB. The elements of an organisational model are defined as *individual assertions* in the ABox (cf. Figure §8.1). Thus, each *specific* person, role, position, organisational unit and capability, is added to the ABox, associated to the corresponding class of the TBox. For instance, to specify that *Project Coordinator* (ABox) is a *Role* (TBox), we use the following DL expression:

$Role(ProjectCoordinator)$

Then, the relations between instances in the ABox are defined as *property assertions* in the ABox. For instance, "Position *THEOS's Project Coordinator* participates in role *Project Coordinator*" is translated as follows:

$participatesIn(THEOS'sProjectCoordinator, ProjectCoordinator)$

Finally, some technical details are applied to the instances of the organisational model. First, all instances are defined as different from each other, since DLs do not assume it. For instance:

$ProjectCoordinator \neq ProjectsAccountAdministrator \neq ... \neq ProjectsResourceManager$

Second, to avoid unintuitive effects of the *open world assumption* in DLs [18] (cf. Appendix §C), the specific number of instances of each concept must be always kept up to date in the KB: $THEOSsProjectCoordinator \in= 3\, participatesIn$.

8.3 Mapping the Business Process Information

The procedure to map the fragment of the BPM metamodel imported in RAL (cf. Figure §6.2) into DLs, differs a bit from the procedure used for the organisational model. First, according to the schema shown in Figure §8.1, and similarly to the previous case, the classes of the BP metamodel (e.g. BusinessProcess, Activity, DataObject, *etcetera*) are mapped into *concepts* in the KB, with two exceptions: (i) the *DataField* class is mapped in a different way, as explained later on in this section; and (ii) class *AssignmentExpression*, which represents the different RAL expressions, is mapped as detailed in Section §8.4. Furthermore, extra information is required in order to deal with the negation form (i.e. operator NOT) of the history-aware RAL expressions (cf. RAL History specification in Section §6.4.4). Specifically, class *History* has been created to represent the overall history of the executions of the BPs of the organisation, as depicted in Figure §8.2.

Regarding the relations between the classes of a metamodel, like before, they are mapped into *properties* of the TBox together with the corresponding *cardinality restrictions*. That is, object properties *hasBPexecution*, *hasActivity*, *hasActivityInstance*, *hasDOInstance*, *reads*, and the rest of relations shown in Figure §8.2 are added to associate the concepts created. Super-property *hasParticipant* has been defined in order to deal with the RAL AC and RAL History expressions aimed at defining access-control

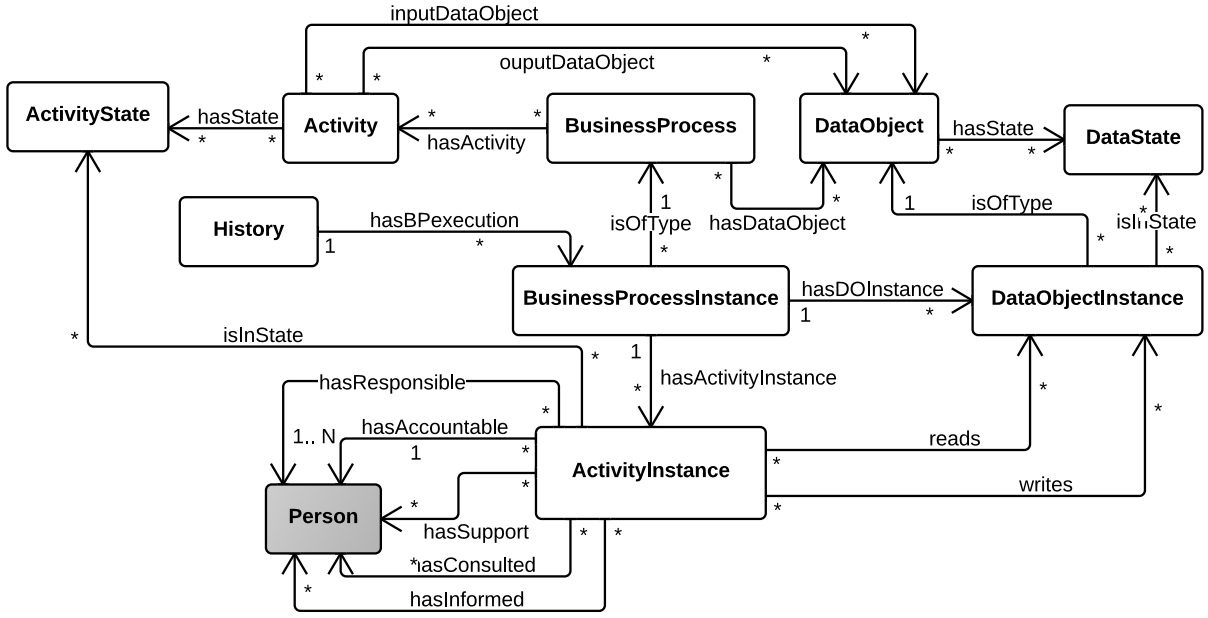


Figure 8.2: DL concepts and properties related to BPM of RAL metamodel

constrains, which allow specifying the task duty we are interested in, or refer to any participant of an activity without specifying the task duty performed (expression `IS INVOLVED IN`).

Properties isBPexecutionOf, *isActivityInstanceOf*, *isDOInstanceOf* and the like, have been defined as the *inverse properties* of the object properties for the relations in the model. An excerpt of the configuration of the TBox regarding the aforementioned relations is shown in Table §8.2. Although not shown in the table, class *ActivityInstance* has a *data property* called *wasCompleted* of standard type *xsd:dateTime*, to store the completion date of the activity.

Regarding the mapping of the BP model, we create instances of the concepts and relations and add them to the ABox of the KB. Thus, instances of *BusinessProcess*, *Activity*, *DataState*, *DataObject*, and *DataState*, are added. For example, for activity *Sign Travel Authorization* of the BP depicted in Figure §6.4, some of the new elements in the ABox are the following:

$TripManagement \in BusinessProcess$

$SignTravelAuthorization \in Activity$

$hasActivity(TripManagement, SignTravelAuthorization)$

Object property	Sub-property of	From	To	Property Type
hasActivity		BusinessProcess	Activity	
hasDataObject		BusinessProcess	DataObject	
hasState		Activiy ⊔ DataObject	ActivityState ⊔ DataState	
isInState		ActiviyInstance ⊔ DataObjectInstance	ActivityState ⊔ DataState	
inputDataObject		Activity	DataObject	
outputDataObject		Activity	DataObject	
hasBPexecution		History	BusinessProcessInstance	
hasActivityInstance		BusinessProcessInstance	ActivityInstance	
hasDOInstance		BusinessProcessInstance	DataObjectInstance	
reads		ActivityInstance	DataObjectInstance	
writes		ActivityInstance	DataObjectInstance	
hasParticipant		ActivityInstance	Person	
hasResponsible	hasParticipant			
hasAccountable	hasParticipant			
hasSupport	hasParticipant			
hasConsulted	hasParticipant			
hasInformed	hasParticipant			

Table 8.2: Properties in the TBox related to the BP metamodel

$DOTravelAuthorization \in DataObject$

$inputDataObject(SignTravelAuthorization, DOTravelAuthorization)$

$SignTravelAuthorization \in= 1\ inputDataObject$

$SignTravelAuthorization \neq SubmitPaper \neq \cdots \neq MakeReservations$

Furthermore, there is an intermediate step that was not necessary in the case of the organisational model, as depicted in Figure §8.1. Specifically, to support the inclusion of run-time information of the BP, we perform a refinement of the KB based on grouping the instances of the same type. In particular, the refinement consists of creating sub-classes of the classes representing the element instances, and introducing some required configuration. This is added to the TBox. Let us explain it by using as example activity *Sign Travel Authorization* of the BP in Figure §6.4.

A sub-class of *BusinessProcessInstance* has to be defined to represent the BP instances related to the conference trip management process. Similarly, sub-classes of *ActivityInstance* and *DataObjectInstance* are introduced for each activity and data object *types* of the process, respectively.

Finally, two axioms are added to state that the process instance is composed of all

of its activity instances and to indicate that all the activity instances are disjoint. The definition in DLs is done as follows:

$$TripManagementInstance \sqsubseteq BusinessProcessInstance$$

$$SignTravelAuthorizationInstance \sqsubseteq ActivityInstance$$

$$DOTravelAuthorizationInstance \sqsubseteq DataObjectInstance$$

$$SignTravelAuthorizationInstance \sqsubseteq = 1\ reads.DOTravelAuthorizationInstance$$

$$SignTravelAuthorizationInstance \sqsubseteq = 1\ writes.DOTravelAuthorizationInstance$$

$$TripManagementInstance \sqsubseteq \exists hasActivityInstance.(SubmitPaperInstance \sqcup ... \sqcup MakeReservationsInstance)$$

$$SignTravelAuthorizationInstance \sqsubseteq \neg\{SubmitPaperInstance, ..., MakeReservationsInstance\}$$

$$TripManagementInstance \sqsubseteq \exists hasDOInstance.(DOTravelAuthorizationInstance \sqcup ... \sqcup DOTransportInstance)$$

$$DOTravelAuthorizationInstance \sqsubseteq \neg\{DOAccommodationInstance, DOTransportInstance\}$$

$$DOTravelAuthorizationInstance \equiv \exists isInState.\{filled, signed, approved, rejected, stored\}$$

Furthermore, in each type of *DataObjectInstance* we create one *data property* for each data field contained by the instances of such a data type, e.g. the data type *TravelAuthorizationInstance* created in the previous example could contain a data field called *Attendee* indicating the person for whom the authorization is requested. The type of the data property depends on its content. For *Attendee*, it could be the standard type *xsd:string*.

Finally, the real instances of the BPM elements that are created at run time, are added to the KB during process execution as instances of the concepts created in the refinement explained above, as depicted in Figure §8.1. Specifically, we need one and only one instance of class *History*, which is related to the specific elements stored in the KB. Then, every time a new process instance is started, DL instances of the corresponding classes or sub-classes previously described have to be added to the KB, and the appropriate property associations have to be configured also at real instance level. For example, for activity *Sign Travel Authorization*, assuming that *hist* is an instance

of *History*, tm_1 is an instance of the BP depicted in Figure §6.4, $sta^1_{tm_1}$ is an instance of the *Sign Travel Authorization* activity in a BP instance, $dota^1_{tm_1}$ is an instance of data object *Travel Authorization* in that process instance, and *signed* is a state of the *Travel Authorization* data object, the mapping to DLs is as follows:

$History(hist)$

$TripManagementInstance(tm_1)$

$hasBPexecution(hist, tm_1)$

$SignTravelAuthorizationInstance(sta^1_{tm_1})$

$hasActivityInstance(tm_1, sta^1_{tm_1})$

$DOTravelAuthorizationInstance(dota^1_{tm_1})$

$hasDOInstance(tm_1, dota^1_{tm_1})$

$reads(sta^1_{tm_1}, dota^1_{tm_1})$

$writes(sta^1_{tm_1}, dota^1_{tm_1})$

$isInState(dota^1_{tm_1}, signed)$

Once a person $person_i$ is selected from the set of potential performers of a task duty, he/she must be added as performer of the task duty for *that* instance of the activity. Similarly, when the activity instance is complete, the completion time is set.

$hasResponsible(sta^1_{tm_1}, person_i)$

$wasCompleted(sta^1_{tm_1}, timestamp_1)$

Like for the organisational metamodel, all instances are defined as different from each other.

$tm_1 \neq tm_2 \neq ... \neq tm_n$

$sp^1_{tm_1} \neq sp^1_{tm_2}, \neq ... \neq, sp^1_{tm_n}$

Similarly to the organisational model, the specific number of instances of each type of element must be always kept up to date. Regarding BPs, this information might change a lot because each time an activity instance starts, new execution information is added to the KB, e.g. a new activity instance $sta^i_{tm_j}$ of type *SignTravelAuthorization-Instance* is initiated in process tm_j. Then, the instance tm_j is updated to indicate that it

has exactly i activities of type *SignTravelAuthorizationInstance*. The same applies to new data object instances, process instances and the history instance:

$tm_j \in= i \quad hasActivityInstance.SignTravelAuthorizationInstance$

$tm_j \in= i \quad hasDOInstance.DOTravelAuthorizationInstance$

$hist \in= j \quad hasBPexecution.TripManagementInstance$

8.4 Mapping RAL Expressions and Constraints

Each RAL expression determines the subset of all the people in the organisation that meet the requirements established in the assignment. For example, a RAL expression stating that a certain activity can only be undertaken (i.e. task duty Responsible) by someone occupying a specific position, limits the set of potential performers to the persons that occupy that position. Thus, we are interested in solving the RAL expressions associated to a task duty TD of an activity a, defined as $expr_a^{TD}$. In case the task duty is not explicitly indicated, task duty Responsible is assumed. For that purpose, we need to know how to map the RAL expression to DLs. The mapping to DLs of a RAL expression is represented as $map_e(expr) : RALExpression \mapsto DL$. Similarly, the mapping of the constraints that may be part of the expressions are referenced to as $map_c(expr) : RALConstraint \mapsto DL$.

Please note that map_e and map_c are not DL constructs, but two auxiliary functions that we use outside the context of DLs to make the description of the mapping more readable. Next, we explain the mapping of each RAL expression (divided into RAL Core, RAL Data, RAL AC, and RAL History), that is, the value of $map_e(expr)$ and $map_c(constr)$ for all the types of expressions and all the types of constraints. In addition, following the same structure as when describing RAL specification in Section §6.4, we provide the DL queries for the example assignment expressions of the application scenario. Notice that syntax $property^-$ is used to represent inverse properties, and that we use *pc* to refer to *PersonContraint*, *grc* to refer to *GroupResourceConstraint*, and *cc* to refer to *CapabilityConstraint* for readability reasons.

8.4.1 Mapping RAL Core Expressions and Constraints

RAL Core involves all the types of expressions addressed by RAL its person-based and group resource-based constraints are limited to referring to a specific person or

group resource, respectively. The part of the KB involved in this extension is (i) all the concepts and relations from the organisational model and (ii) the activity instances of a process regarding the BP model, since the resource assignment expressions are associated to the process activities. In the following we show the mapping of each type of RAL Core expression.

PersonExpr. This expression allows assigning an activity to a specific person, who is indicated by means of a *PersonConstraint*. The mapping is straightforward. The DL query is equivalent to the mapping of the *PersonConstraint* referenced.

$$map_e('IS\ pc') \equiv map_c(pc)$$

GroupResourceExpr. This type of expression is used to assign an activity to a specific group resource, which can be done is several different ways:

HAS (POSITION | UNIT) GroupResourceConstraint. It assigns people occupying a specific position or belonging to a specific organisational unit.

$$map_e('HAS\ POSITION\ grc') \equiv \exists occupies.map_c(grc)$$
$$map_e('HAS\ UNIT\ grc') \equiv \exists occupies.(\exists isMemberOf.map_c(grc))$$

HAS ROLE roleID [IN UNIT unitID]. In case only the role is indicated, the mapping is straightforward. If also the unit is specific, it consists of an intersection between the people that participate in the role indicated and the people that belong to the organisational unit indicated.

$$map_e('HAS\ ROLE\ grc') \equiv \exists occupies.(\exists participatesIn.map_c(GRC))$$
$$map_e('HAS\ ROLE\ grc\ IN\ UNIT\ grc') \equiv$$
$$\equiv \exists occupies.(\exists participatedIn.map_c(grc) \sqcap \exists isMemberOf.map_c(grc))$$

CommonalityExpr. These expressions deal with the Commonality-based Allocation pattern, that is, assign activities to people based on the features they have in common with other people, and their mappings are as follows:

SHARES Amount (POSITION | UNIT) WITH PersonConstraint. The mapping for an assignment that indicates that a task duty has to be performed by an individual that has *some* position in common with somebody specified

in a *PersonConstraint*, and the mapping for an assignment that indicates that a task duty has to be performed by an individual that has *all* the organisational units in common with somebody specified in a *PersonConstraint*.

$$map_e('SHARES\ SOME\ POSITION\ WITH\ pc') \equiv \exists occupies.(\exists occupies^{-}.map_c(pc))$$
$$map_e('SHARES\ ALL\ UNIT\ WITH\ pc') \equiv \exists occupies.(\exists isMemberOf.(\forall isMemberOf^{-}.(\exists occupies^{-}.map_c(pc))))$$

SHARES Amount ROLE [IN UNIT GroupResourceConstraint] WITH PersonConstraint. Next, we show the mapping for an assignment that indicates that a task duty has to be performed by an individual that has *some* role in common with somebody specified in a *PersonConstraint*, and the mapping for an assignment that indicates that a task duty has to be performed by an individual that has *all* the roles within an organisational units specified in a *GroupResourceConstraint* in common with somebody specified in a *PersonConstraint*.

$$map_e('SHARES\ SOME\ ROLE\ WITH\ pc') \equiv \exists occupies.(\exists participatesIn.(\exists participatesIn^{-}.(\exists occupies^{-}.map_c(pc))))$$
$$map_e('SHARES\ ALL\ ROLE\ IN\ UNIT\ grc\ WITH\ pc') \equiv \exists occupies.(\exists participatesIn.(\exists participatesIn^{-}.isMemberOf.(map_c(grc)) \sqcap \forall participatesIn^{-}.(\exists occupies^{-}.map_c(pc))))$$

CapabilityExpr. It allows expressing assignments based on the capabilities of individual resources that can be allocated to a task duty for an activity.

$$map_e('HAS\ CAPABILITY\ cc') \equiv \exists hasCapability.map_c(cc)$$

IsAssignmentExpr. It is used to indicate that an activity has the same RAL expression as another activity. The mapping consists of indicating the assignment expression of the activity referenced.

$$map_e('IS\ ASSIGNMENT\ IN\ ACTIVITY\ activityID') \equiv map_e(expr_{activityID})$$

HierarchyExpr. It allows playing with the positional hierarchy of the organisational model in two different ways:

ReportExpr. It is defined either as the persons who occupy a position that has an *extendedReportsTo* relation with a given position name or with the positions occupied by a given person (in case directly is *false*); or as the persons who occupy a position that has a *reportsTo* relation with a given position name or with the positions occupied by a given person (in case of *direct reports*):

$$map_e('REPORTS\ TO\ POSITION\ grc') \equiv \exists occupies.(\exists extendedReportsTo.\ map_c(grc))$$
$$map_e('DIRECTLY\ REPORTS\ TO\ pc') \equiv \exists occupies.(\exists reportsTo.\ (\exists occupies^{-}.(map_c(pc))))$$

DelegateExpr. It is similar to *Report* but using organisational relation *can delegate work to*, i.e., this time we move down in the positional hierarchy. In this case transitivity is implicit. Let us show the mapping for the opposite direction of the expression, that is, to state that the work has to be assigned to somebody that can have work delegated from another person according to his/her position.

$$map_e('CAN\ HAVE\ WORK\ DELEGATED\ BY\ pc') \equiv \exists occupies.(\exists extendedCanHaveWorkDelegated.(\exists occupies^{-}.map_c(pc)))$$

NegativeExpr. This kind of expression is mapped to the following DL expression.

$$map_e('NOT\ (expr)') \equiv Person \sqcap \neg map_e(expr)$$

CompoundExpr. This kind of assignments has a straight mapping.

$$map_e('(expr1)\ AND\ (expr2)') \equiv map_e(expr1) \sqcap map_e(expr2)$$
$$map_e('(expr1)\ OR\ (expr2)') \equiv map_e(expr1) \sqcup map_e(expr2)$$

Regarding RAL Core Constraints, the mapping involves the following elements:

PersonConstraint. This type of constraint limits the scope of the persons that are mentioned in some RAL expressions, e.g. *PersonExpr*. In RAL Core the only one *PersonConstraint* considered consists of indicating the specific identity of the person in question. The DL query for map_c(PersonConstraint) is direct.

$$map_c(personName) \equiv \{personName\}$$

GroupResourceConstraint. It is used to indicate a group resource required in some types of expressions, e.g. *GroupResourceExpr*. In RAL Core it consists of specifying the identity of the position, role or unit in question, so map_c(GroupResourceConstraint) is similar to map_c(PersonConstraint).

$$map_c(groupResourceName) \equiv \{groupResourceName\}$$

CapabilityConstraint. These constraints consist of either having a certain capability, or meeting a certain condition on the value of a capability. We next show the mapping for map_c(CapabilityConstraint) when it refers to possessing a specific capability, and when it is about having a capability with a specific value in some of its attributes (the *equal* operator). The rest of operations required to deal with other types of value comparison should also be mapped into DLs.

$$map_c(capabilityID) \equiv \{capabilityID\}$$
$$map_c(capabilityID.attribute = value) \equiv \{capabilityID\} \sqcap \exists attribute.\{value\}$$

Figure §8.3 shows the DL queries for activities *Submit Paper* and *Fill Travel Authorization* of the BP of our use case, according to the RAL assignments in Section §6.4.1.

Submit Paper. A *PhD Student* of project *THEOS* is in charge of submitting the paper to the conference.
RAL: `HAS ROLE ProjectsPhDStudent IN UNIT THEOS`
DL: $\exists occupies.(\exists participatesIn.\{ProjectsPhDStudent\} \sqcap \exists isMemberOf.\{THEOS\})$

Fill Travel Authorization. The authorization form must be filled in by somebody that reports to Anthony.
RAL: `REPORTS TO Anthony`
DL: $\exists occupies.(occupies^{-}.(\exists extendedReportsTo.\{Anthony\}))$

Figure 8.3: DL queries for the assignments of some BP activities

8.4.2 Mapping of RAL Data Constraints

As explained in Section §6.4.2, RAL Data extends RAL Core to allow the assignment of activities to resources that are indicated in a data object, that is, information that is not available until run time. In RAL specification, we extended the *PersonConstraint*

and the *GroupResourceConstraint* to address the two options, i.e. the Direct Allocation or the Deferred Allocation pattern (cf. Section §3.2.3). The mapping to DLs of the constraints is shown below, followed by the DL query for activity *Send Travel Authorization* of the BP of our use case in Figure §8.4. Notice that the mapping to DLs is the same for *PersonConstraint* and for *GroupResourceConstraint*.

$$map_c(\text{'PERSON IN DATA FIELD } do.field\text{'}) \equiv \exists field^{-}.(doDataObjectInstance \sqcap \exists hasDOInstance^{-}.\{bp_{currentInstance}\})$$

Send Travel Authorization. This task must be undertaken by the person referenced in field *Attendee* of data object *Travel Authorization*.
RAL: `IS PERSON IN DATA FIELD TravelAuthorization.Attendee`
DL: $\exists Attendee^{-}.(\exists TravelAuthorizationDataObjectInstance \sqcap \exists hasDOInstance^{-}.\{bp_{currentInstance}\})$

Figure 8.4: DL query for the assignment of activity *Send Travel Authorization*

8.4.3 Mapping of RAL AC Constraints

RAL AC extends RAL Core to deal with dynamic access-control constraints: DBoD and DSoD. Thus, the ABox must always contain the specific instances of all the elements that have already been instantiated, given a certain point in time in the execution of a process instance.

In particular, in RAL AC one new constraint is added to *PersonConstraint* to indicate that the task duty in question for that activity has to be performed by the performer of one of the task duties for another activity (in the same BP instance). The mapping of map_c(PersonConstraint) when it refers to the person Responsible for another activity is shown below for instance $bp_{currentInstance}$ of a BP, together with the mapping in case of using the `INVOLVED IN` expression. In the latter case, any participant of the activity is valid for allocation, so we could choose from the union of the potential performers for all the task duties. Property *hasParticipant* was added to the KB to provide this functionality (cf. Section §8.3). Furthermore, value $AfterAllocation$ represents any state of the activity once the participants have been allocated.

$$map_c('PERSON\ RESPONSIBLE\ FOR\ ACTIVITY\ activityID') \equiv \exists hasResponsible^{-}.$$
$$(\exists hasActivityInstance^{-}.\{bp_{currentInstance}\} \sqcap \{activityIDinstance\} \sqcap$$
$$\exists isInState.AfterAllocation)$$
$$map_c('PERSON\ INVOLVED\ IN\ ACTIVITY\ activityID') \equiv \exists hasParticipant^{-}.$$
$$(\exists hasActivityInstance^{-}.\{bp_{currentInstance}\} \sqcap \{activityIDinstance\} \sqcap$$
$$\exists isInState.AfterAllocation)$$

A special situation is that with cyclic structures in the BP. In case of loops, we consider that any of the performers of the corresponding task duty (if specified) in the instances of the referenced activity in the same process instance, is a potential performer of the concrete task duty for the activity at hand.

Sign Travel Authorization. The authorization form can be signed only by a person that can supervise the work developed by the person who filled in the document. Furthermore, this person has to be different from the one who filled in the form.

RAL: `(IS REPORTED BY PERSON RESPONSIBLE FOR ACTIVITY FillTravelAuthorization) AND (NOT (IS PERSON RESPONSIBLE FOR ACTIVITY FillTravelAuthorization))`

DL: $\exists occupies.(\exists extendedReportsTo.(\exists occupies^{-}.(\exists hasResponsible^{-}.$
$(\exists hasActivityInstance^{-}.\{bp_{currentInstance}\} \sqcap \{FillTravelAuthorizationInstance\}))))\sqcap$
$\neg(\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.\{bp_{currentInstance}\} \sqcap$
$\{FillTravelAuthorizationInstance\}))$

Figure 8.5: DL query for the assignment of activity *Sign Travel Authorization*

8.4.4 Mapping of RAL History Constraints

RAL History extends RAL AC to define static access-control constraints (SBoD and SSoD). Thus, now the ABox must contain the specific instances of all the elements that have already been instantiated for all the process instances executed and under execution.

The extensions defined for *PersonConstraint* were described in Section §6.4.4. In short, the constraint introduced in RAL AC is extended with expression *HistoryExpression*, and one more constraint is added to refer to any participant in different process executions. The mapping has been defined using the Responsible task duty of activity

Submit Paper as the referenced performer (when required), and it is shown in Table §8.3 below. We assume that we are running an execution of the process and, thus, we know the identifier of the current BP instance (in this case $bp_{currentInstance}$).

PersonConstraint	map_c(**PersonConstraint**)
PERSON RESPONSIBLE FOR ACTIVITY SubmitPaper	
IN CURRENT INSTANCE	$\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.\{bp_{currentInstance}\} \sqcap SubmitPaperInstance)$
IN ANY INSTANCE	$\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagementInstance) \sqcap SubmitPaperInstance))$
IN ANOTHER INSTANCE	$\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagementInstance) \sqcap \neg\{bp_{currentInstance}\}) \sqcap SubmitPaperInstance)$
FROM startDate TO endDate	$\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagementInstance) \sqcap SubmitPaperInstance \sqcap \exists(wasCompleted \geq startDate) \sqcap \exists(wasCompleted \leq endDate)))$
PERSON WHO HAS BEEN RESPONSIBLE FOR ANY ACTIVITY IN	
CURRENT INSTANCE INSTANCE	$\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.\{bp_{currentInstance}\})$
ANY PROCESS INSTANCE	$\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.(\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagementInstance))$
ANOTHER PROCESS INSTANCE	$\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagementInstance) \sqcap \neg\{bp_{currentInstance}\})$
A PROCESS INSTANCE BETWEEN startDate AND endDate	$\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagementInstance) \sqcap \exists(wasCompleted \geq startDate) \sqcap \exists(wasCompleted \leq endDate)))$

Table 8.3: Mapping of some RAL History constraints into DL *concepts*

Figure §8.6 shows the DL query for some activities of the BP of our use case, according to the RAL assignments defined in Section §6.4.4.

8.5 SUMMARY

In this chapter we have described RAL formal semantics, which has been defined in DLs in order to provide a precise meaning for each RAL expression, and to ease the

Check Response. Any participant in any instance of the BP can check the answer received from the *Research Vice-chancellorship*, as long as he/she has some organisational unit in common with the person that submitted the paper in the current process instance.

RAL: `(IS PERSON WHO HAS PARTICIPATED IN ANY PROCESS INSTANCE) AND (SHARES SOME UNIT WITH PERSON RESPONSIBLE FOR ACTIVITY SubmitPaper IN CURRENT INSTANCE)`

DL: $\exists hasParticipant^{-}.(\exists hasActivityInstance^{-}.(\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagementInstance)) \sqcap \exists occupies.(\exists isMemberOf.(\exists isMemberOf^{-}.(\exists occupies^{-}.\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.\{bp_{currentInstance}\} \sqcap SubmitPaperInstance))))$

Register at Conference. The person responsible or accountable for task *Submit Paper* in the ongoing instance is due to register at the conference.

RAL: `(IS PERSON RESPONSIBLE FOR ACTIVITY SubmitPaper IN CURRENT INSTANCE) OR (IS PERSON ACCOUNTABLE FOR ACTIVITY SubmitPaper IN CURRENT INSTANCE)`

DL: $\exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.\{bp_{currentInstance}\} \sqcap SubmitPaperInstance) \sqcup \exists hasAccountable^{-}.(\exists hasActivityInstance^{-}.\{bp_{currentInstance}\} \sqcap SubmitPaperInstance)$

Make Reservations. Somebody with role *Clerk* or any person that has previously been responsible for this activity can become responsible for making the reservations again, even if he is not the one attending the conference.

RAL: `(HAS ROLE Clerk) OR (IS PERSON RESPONSIBLE FOR ACTIVITY MakeReservations IN ANOTHER INSTANCE)`

DL: $\exists occupies.(\exists participatedIn.\{Clerk\}) \sqcup \exists hasResponsible^{-}.(\exists hasActivityInstance^{-}.((\exists hasBPexecution^{-}.\{hist\} \sqcap TripManagementInstance) \sqcap \neg\{bp_{currentInstance}\}) \sqcap MakeReservationsInstance)$

Figure 8.6: DL queries for the assignments of some activities of the BP

automated analysis of the resource assignments defined with RAL.

9

Automated Person-Activity Analysis Operations

"But after observation and analysis, when you find that anything agrees with reason and is conducive to the good and benefit of one and all, then accept it and live up to it"

Buddha (563 b.C–483 b.C),
Hindhu Prince

The second main goal of this thesis is to provide support for the execution of the analysis operations that were defined in Chapter §5. In this chapter we deal with the automation of the first group of analysis operations, which focus on the relationships between persons and activities. The general approach to automate them is to leverage off-the-shelf DL reasoners to extract information from the DL-based KB described in Chapter §8 and some additional information that must be added to RAL's KB in order to deal with the automated analysis operations. Specifically, in Section §9.1, we overview the chapter and provide precise definitions for design-time and run-time resource-related analysis. Next, in Section §9.2 we describe all the information that must be added to RAL's KB. The next section present a DL-based implementation of each analysis operation that belongs to the group of relationships between persons and activities. Finally, in Section §9.4 we sum up the content of the chapter.

9.1 INTRODUCTION

As stated in Chapter §4, BP analysis helps ensure the proper operation of BPs. Specifically, analysing the resource perspective helps to know whether the human resources of an organisation are being properly managed, and to detect problems derived from an incorrect specification of resources in BP models. In this dissertation, we specify the analysis of models in terms of analysis operations and, in Chapter §5, a catalogue of resource-related analysis operations that include operations related with the potential performers of an activity and operations related to data access control was presented. Besides, these analysis operations can be used either at design time or at run time, which corresponds to the Design and Analysis, and Enactment phases of the BP lifecycle, respectively.

In design-time analysis, the context of the analysis operations is all the possible executions of the BP. Therefore, a person is potentially responsible for an activity if there is some process instance in which he/she can be the responsible for such activity. This means that no specific run-time information is available in this type of analysis, which affects resource assignments that involve RAL Data, RAL AC and RAL History. Regarding RAL Data, the value of the data fields is unknown until run time. As far as RAL AC and RAL History are concerned, the inconvenience comes from not knowing the actual performers of the BP activities, which influences the resolution of the access-control constraints. In both cases, the implementation of the analysis operations must provide mechanisms to deal with this uncertainty.

Conversely, in run-time analysis, the context of the analysis operations is a specific running instance of a BP. Therefore, a person is potentially responsible for an activity if he or she can be the person responsible for such activity in that specific running instance. This means that information about the current value of data fields and the actual performers of executed activities is available. However, most analysis operations must also take into consideration activities that have not been executed in the process instance yet. For them, there is again uncertain information regarding the value of data fields and the performers of the activities, which must be thus addressed.

The goal of this chapter is to provide a reference implementation to automate the analysis operations that focus on relationships between persons and activities both at design-time and at run-time. To this end, the approach we follow involves using the RAL semantics expressed as a DL-based KB as ground and leveraging off-the-shelf DL reasoners [96, 98, 99], which are software tools that implement several operations

on ontologies and KBs in an efficient manner, to perform the analysis operations. In addition, it is necessary to add new elements to the TBox and the ABox of the KB together with RAL semantics in order to support the execution of the operations at design time and at run time, and to take into account the BP control flow and/or the data perspectives, when necessary.

9.2 EXTENDING THE DL-BASED KB

The starting point of the DL-based KB is the mapping introduced in Sections §8.2 and §8.3. In it, the organisational model and the process model are mapped into individuals of the ABox of the KB. However, automating the analysis operations raises two issues that require new axioms and assertions to be added to the KB.

On the one hand, automating resource-related analysis operations involves coping with several BP perspectives, namely resources, data and control flow. For instance, control flow is inherently involved in operations critical participants and permanent participants, in which is is necessary to know which BP activities must be executed in all possible process instances. Furthermore, the control flow of the process usually needs to be considered in run-time analysis in order to calculate the set of activities that are involved in the analysis.

On the other hand, it is necessary to include in the KB the information derived from the mapping of RAL expressions into DLs. However, the mapping has to be slightly changed to deal with the information that is unknown at design-time, namely the value of the data fields and the actual performers of the BP activities, which influences the resolution of RAL Data and RAL AC and RAL History expressions, respectively. Furthermore, this unknown information has to be dealt with at run time as well because most analysis operations take into consideration activities that have not been executed in the process instance yet.

9.2.1 Adding Information Related to Control Flow

There are two aspects related to the control flow of a process that must be identified in order to execute the analysis operations we are interested in, namely: (i) figure out which activities can be executed after a specific activity; and (ii) figure out whether the execution of an activity is mandatory from a certain activity of the process. To do so, the following relations are defined:

Weak order (a, b) This relation allows detecting whether activity b can be executed after activity a in a BP instance, that is, if there is at least one path from a to b. It was already defined by Weidlich et al. [136] when they introduced the term of *Behavioural Profile*, which is a set of relations between the activities of a BP to detect the order in which they can be executed and, thus, identify structures such as sequences, bifurcations, and the like, in the process. The procedure defined by the authors use WF-systems [129], and calculate the *strict order*, *exclusiveness* and *interleaving* relations from the underlying Petri Net by using the *weak order* relation. As demonstrated in [136], the weak order relation can be obtained very efficiently for *sound free-choice* systems [52].

Mandatory (a, b) An activity is mandatory if and only if it is executed in all the possible instances of a BPs. We will use this concept with respect to a reference activity, that it, $mandatory(a,b)$ indicate whether the execution of activity b is mandatory from activity a. This relation is equivalent to the *leadsto* relation introduced in BPMN-Q [15], a language to query the BP control flow and data perspectives. In that approach, Awad et al. define a BPMN sub-graph of the fragment of the process that is affected by the leads to relation, and they use *Past Linear Temporal Logic (PLTL)* to perform the checking.

In order to use these two relations with RAL, we need to be able to define them in terms of DL elements and add them to RAL's DL-based KB. Consequently, we have extended the KB with two properties *weakOrder* and *mandatory* whose domain and range are *Activity* so that let $a_1, a_2 \in Activity$ be two activities, $weakOrder(a_1,a_2)$ means that there is a process instance in which a_2 takes place after a_1, and $mandatory(a_1,a_2)$ means that a_2 follows a_1 for all process instances. Figure §9.1 shows an example of the KB with these new properties together with the relationships between *ActivityInstance* and *Activity* by means of property $isOfType$.

9.2.2 Adding Information for Design-Time Analysis

To automate the analysis operations at design-time it is necessary to explicitly include in the KB information about the potential performers of an activity. To this end, we add new properties called *isPotentialResponsible*, *isPotentialAccountable*, *isPotentialSupport*, *isPotentialConsultant* and $isPotentialInformed$ for each task duty. Their domain is *Person* and their range is *Activity*. These properties represent the potential performers of a given task duty for a given activity in all possible executions of the BP.

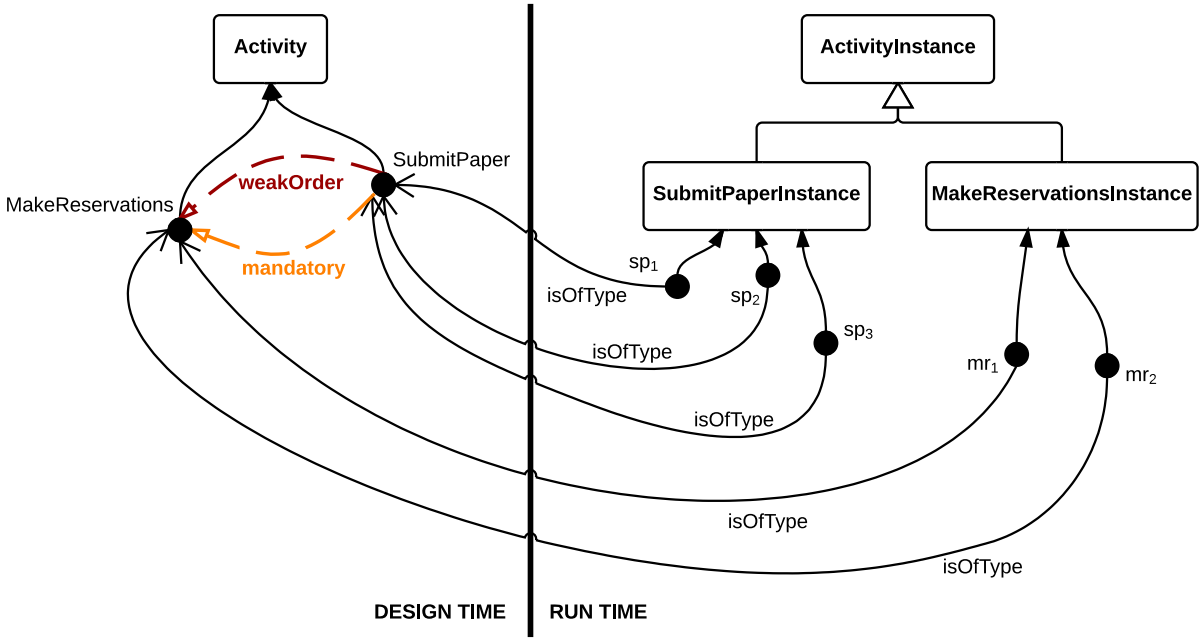


Figure 9.1: Overview of the extension of RAL's KB to deal with control flow

For instance, $isPotentialResponsible(p,a)$ means that person p can be responsible for activity a.

To give semantics to the properties in terms of the elements defined in the KB one could think of using the mapping $map_e : RALExpression \mapsto DL$ and its associated mapping for RAL constraints $map_c : RALConstraint \mapsto DL$ both specified in Section §8.4. However, at design time, not all the information required to solve some RAL Data, RAL AC and RAL History expressions is available. In particular, the *PersonConstraints* and the *GroupResourceConstraints* assume the existence of run-time information such as the content of data fields and the identity of the actual performers of previous activity instances. As this information is only known at run time, at design time we cannot use map_e for the constraints affected. Therefore, two new mappings $map_e^{DT} : RALExpression \mapsto DL$ and $map_c^{DT} : RALConstraint \mapsto DL$ that do not take into consideration run-time information must be used instead.

Mapping map_c^{DT} is the same as map_c except for those constraints that involve obtaining information from the run-time state of a process. Table §9.1 presents the mapping to DLs for such types of constraints. It consists of using: (i) all the persons in the organisation in case the specific person is supposed to come in a data field since we assume any person in the organisation may come in the data field; and (ii) all the

persons who can do a certain activity in the BP in case the concrete person is defined as an access-control constraint with respect to a previous activity instance.

PersonConstraint	$map_c^{DT}(PersonConstraint)$
PERSON IN DATA FIELD do.field	*Person*
IS PERSON RESPONSIBLE FOR ACTIVITY SubmitPaper	
IN CURRENT INSTANCE IN ANY INSTANCE IN ANOTHER INSTANCE	$\exists isPotentialResponsible.\{SubmitPaper\}$
IS PERSON WHO HAS BEEN RESPONSIBLE FOR ANY ACTIVITY IN	
CURRENT PROCESS INSTANCE ANY PROCESS INSTANCE ANOTHER PROCESS INSTANCE	$\exists isPotentialResponsible.(\exists hasActivity^{-}.\{TripManagement\})$
GroupResourceConstraints	$map_c^{DT}(GroupResourceConstraints)$
IN DATA FIELD do.field (referred to Position)	*Position*
IN DATA FIELD do.field (referred to Role)	*Role*
IN DATA FIELD do.field (referred to Unit)	*OrganizationalUnit*

Table 9.1: Design-time mapping of constraints into DLs

Regarding mapping map_e^{DT}, it is the same as map_e except for two aspects. The first one is straightforward; all expressions that use map_c should use map_c^{DT} instead. The second one involves the negative expressions (i.e. `NOT`). All negative expressions that contain one of the constraints in the tables are mapped to *Person*. To illustrate the reason of this mapping let us consider the RAL expression `NOT IS PERSON RESPONSIBLE FOR ACTIVITY a`. In this case, this expression involves all people who cannot be responsible for activity *a*, but since only one person can be responsible for activity *a* in a given instance, it also involves all people that could be responsible for activity *a*, but have not been responsible for it in a given instance. Since the person responsible for activity *a* can change in the different instances, potentially all people could satisfy that RAL expression.

Finally, given these design-time mappings, the semantics of property *isPotentialResponsible* can be defined for a BP by adding the following axiom to the KB for each

activity a of the BP:

$$\exists isPotentialResponsible.\{a\} \equiv map_e^{DT}(expr_a^r)$$

A similar axiom can be straightforwardly added for all of the others task duties, i.e., accountable, support, consult and inform.

9.2.3 Adding Information for Run-Time Analysis

To enable the run-time analysis, we follow a similar approach as the one used for design-time analysis. In this case, we define a new property for each task duty called *potResponsible*, *potAccountable*, *potSupport*, *potConsultant* and $potInformed$ whose domain is *Person* and whose range is *ActivityInstance* and represent the potential performers of a given task duty for an activity instance of a BP. From now on, we will focus on *potResponsible*, but the same approach can be straightforwardly applied to the other properties.

Note that property *potResponsible* is the run-time equivalent of property *isPotentialResponsible* that was added to enable design-time analysis, the main difference being that *isPotentialResponsible* relates people with activities, whereas *potResponsible* relates people with activity instances. The main consequence of this difference is that the definition of *isPotentialResponsible* is always the same since activities do not have state, whereas property *potResponsible* has to be defined differently depending on whether the related activity instance has been allocated (either finished or still running), is being allocated or may be allocated in the future. Furthermore, the definition of *potResponsible* changes since activity instances change their state as the execution of the process instance advances.

Let *currentInstance* be the process instance that is the context of the analysis, $AfterAllocation \sqsubseteq ActivityState$ those states of an activity instance that occur after the resource allocation, and $BeforeAllocation \sqsubseteq ActivityState$ those states of an activity instance that occur before the resource allocation. Besides, according to the mapping detailed in Section §8.3, the current state of the KB regarding the execution information of a process instance is for each activity that is being performed or has been performed, there is an individual in the ABox of the KB that represents it. Furthermore, for each of those individuals there is information about its process instance (property $hasActivityInstance^-$), its corresponding activity model (property $isOfType$), its current state (property *isInState*) and the people that has performed any task duty re-

lated with it (properties *hasResponsible*, *hasAccountable*, *hasSupport*, *hasConsultant* and $hasInformed$). Given this context, the definition of *potResponsible* is as follows:

The person that is potential responsible for an activity instance *ai* that has been allocated is exactly the person Responsible that has been allocated:

$$\exists potResponsible.\{ai\} \equiv \exists hasResponsible^{-}.\{ai\}$$

The activity instances that have been allocated are those individuals *ai* such that:

$$ai \in \exists hasActivityInstance^{-}\{currentInstance\} \sqcap \exists isInState.AfterAllocation$$

The person that is potential responsible for an activity instance *ai* that is being allocated can be defined using the mapping map_e since all the information that is necessary to be allocated is available. Therefore, $\exists potResponsible.\{ai\} \equiv map_e(typeof(ai))$, where $typeof(ai)$ is a function that returns the type of an activity instance and can be implemented as $individuals(\exists isOfType^{-}.\{ai\})$. The activity instances that are being allocated are those individuals *ai* that are not allocated yet, but they have started:

$$\begin{aligned} ai \in \exists hasActivityInstance^{-}.\{currentInstance\} \\ \sqcap \exists isInState.(BeforeAllocation \sqcap StartState) \end{aligned}$$

The definition of the resource potentially responsible for an activity instance that may be allocated in the future has the same problems than the definition of the person that is potential responsible for an activity at design-time because it is necessary to deal with unknown values of data and the person responsible for activities that have not been executed yet. Therefore, two new mappings $map_e^{URT} : RALExpression \mapsto DL$ and $map_c^{URT} : RALConstraint \mapsto DL$ have to be defined. Mapping map_c^{URT} is similar to map_c^{DT}, but in this case it refers to activity instances instead of activities as depicted in Table §9.2. Regarding mapping map_e^{URT}, it is equivalent to map_e^{DT} except for the negation expressions that refer to an access-control constraint of an activity *a* that cannot be allocated in the future. In this case its mapping is not changed into *Person* since the person who is responsible for the activity is not unknown. Given these two mappings, the axiom $\exists potResponsible.\{ai\} \equiv map_e^{URT}(typeof(ai))$ should be added to the KB for each individual *ai* that may be allocated in the future.

However, there is one more thing. According to our mapping, there are no individuals in the KB for the activity instances that may be allocated in the future because they have not started yet. Therefore, it is necessary to add an individual for each activity

PersonConstraint	$map_c^{URT}(PersonConstraint)$
PERSON IN DATA FIELD do.field	$Person$
IS PERSON RESPONSIBLE FOR ACTIVITY SubmitPaper	
IN CURRENT INSTANCE	$\exists potResponsible.(\exists isOfType\{SubmitPaper\} \sqcap \exists hasActivityInstance^{-}.\{currentInstance\})$
IN ANY INSTANCE	$\exists potResponsible.(\exists isOfType\{SubmitPaper\})$
IN ANOTHER INSTANCE	$\exists potResponsible.(\exists isOfType\{SubmitPaper\} \sqcap \neg\exists hasActivityInstance^{-}.\{currentInstance\})$
IS PERSON WHO HAS BEEN RESPONSIBLE FOR ANY ACTIVITY IN	
CURRENT PROCESS INSTANCE	$\exists potResponsible.(\exists hasActivityInstance^{-}.\{currentInstance\})$
ANY PROCESS INSTANCE	$\exists potResponsible.(\exists hasActivityInstance^{-}.TripManagementInstance)$
ANOTHER PROCESS INSTANCE	$\exists potResponsible.(\exists hasActivityInstance^{-}.(TripManagementInstance \sqcap \neg\{currentInstance\}))$
GroupResourceConstraints	$map_c^{URT}(GroupResourceConstraints)$
IN DATA FIELD do.field (referred to Position)	$Position$
IN DATA FIELD do.field (referred to Role)	$Role$
IN DATA FIELD do.field (referred to Unit)	$OrganizationalUnit$

Table 9.2: Run-time mapping of constraints into DLs

that may be allocated in this process instance in the future and define their $isOfType$ and $isInState$ properties accordingly. Specifically, an individual should be added for each activity that fulfills the following condition:

$$\exists weakOrder.(\exists isOfType^{-}.(\exists hasActivityInstance^{-}.(\{currentInstance\}) \sqcap \exists isInState.StartState)) \sqcap (\exists weakOrder.\{Self\} \sqcup \exists isOfType^{-}.(\exists hasActivityInstance^{-}.(\{currentInstance\}) \sqcap \neg(\exists isInState.(StartState \sqcup EndState))))$$

The first part of the intersection identifies those activities that may follow ($weakOrder$) all the activities that are being executed at this moment ($isInState.StartState$). The second part of the intersection is to remove all those activities to which the weak order

relation is caused by being in parallel branches (they are not in weak order with themselves [136]) and have already started (maybe even finished).

9.3 Automating Analysis Operations Using DL

In this section, we provide a reference implementation for each Person-Activity operation (cf. Section §5.3) both at design time and at run time. Our approach is to express the analysis operations in terms of standard DL reasoning operations, which are implemented in an efficient manner by any off-the-shelf DL reasoner. The reasoning operations are:

- $satisfiable(C)$ determines whether a description of concept C is not contradictory.
- $individuals(C)$ finds all individuals that are instances of concept C.

In the definition of the operations, we make the following assumptions:

1. The operations are done in the context of a BP model and an organisational model and the KB used by the DL reasoners include both their mapping as detailed in Sections §8.2 and §8.3 and the extension described in the previous section.
2. The RAL expressions used in the BP are valid, which means: (i) the resource assignments must be well-defined according to the specification of RAL; and (ii) in case of existing access-control constraints between two activities a and b, the constraint is placed in the activity that always takes place after the other.
3. The data values used in RAL Data expressions contain valid references to the organisational model, i.e., it is not possible a data value that references a person that does not exist in the organisation.

Note that our goal is not to provide the most efficient implementation of every operation, but to provide an implementation that can be used as a reference for the development of more efficient implementations for some of these operations, which can be based on other formalisms or ad-hoc algorithms.

9.3.1 Potential Performers

This operation receives an activity a of a BP and a task duty (e.g. *Responsible*) and returns the people that can potentially perform such task duty in a. In terms of DL the operation can be formulated as follows:

- At design time, the potential performers of activity a with task duty Responsible are those persons p that have a property $isPotentialResponsible(p,a)$. In DL, this can be expressed as the result of:

$$individuals(\exists isPotentialResponsible.\{a\})$$

- At run time it is similar, but using property $potResponsible$ and setting the context to the current process instance ($currentInstance$):

$$individuals(\exists potResponsible.(\exists isOfType.\{a\} \sqcap \exists hasActivityInstance^{-}.\{currentInstance\}))$$

  Note that we assume the person potential responsible for an activity that has been allocated and it is not going to take place again in the process instance is the person who was allocated as responsible for the activity.

There is a variant of this operation at run time in which we may be interested to know which are the potential performers of the activity instance that is being allocated instead of the potential performers of all possible activity instances of type a that may take place in the current process instance. This variant make sense in processes with loops in which the same activity instance can be repeated several times. In that case, the previous definition would return the people that can perform activity a considering the current activity instance and all the activity instances that may come in the future if the loop is taken. However, sometimes we may be interested in knowing which are the potential performers considering only the current activity instance. This may be the case of a BPMS that wants to know the people to which the next activity instance must be offered. In this case the mapping map_e can be used to obtain the DL expression directly: $individuals(map_e(expr_a^r))$.

9.3.2 Potential Activities

This operation receives a person p and a task duty (e.g. *Responsible*) and returns the activities that p can potentially perform with such task duty. In terms of DL the operation can be defined as follows:

- At design time, the potential activities of person p with task duty Responsible are those activities a that have a property $isPotentialResponsible(p,a)$, which can be expressed as:

$$individuals(\exists isPotentialResponsible^{-}.\{p\})$$

- At run time, the potential activities of person p with task duty Responsible are those activities a that have an activity instance that satisfies three restrictions: a person potentially responsible for a is p, belongs to the $currentInstance$, and its state is $BeforeAllocating$ since we do not consider the activities that already took place. In DL, this can be expressed as:

$$\begin{aligned} individuals(\exists isOfType^{-}.(\ & \exists potResponsible^{-}.\{p\} \\ & \sqcap \exists hasActivityInstance^{-}.\{currentInstance\} \\ & \sqcap \exists isInState.BeforeAllocating)) \end{aligned}$$

9.3.3 Consistency Checking

This operation receives an activity a and a task duty (e.g. *Responsible*) and returns whether the activity is consistent with regard to such task duty. An activity is consistent with regard a task duty if there is at least one person that is allowed to perform the task duty for the activity, according to the assignment expression associated to the task. This operation has three possible outcomes:

- Yes. The activity is consistent because there is at least one potential performer for the given task duty for all activity instances of activity a.
- No. The activity is inconsistent because there is no potential performer for the given task duty for all activity instances of activity a.
- Partial. The activity is partially inconsistent because there may be activity instances of activity a for which there is no potential performers for the given task duty.

The approach to implement this operation depends on the components of RAL used for the resource assignments.

RAL Core. In this case, there are only two possible outcomes: yes and no. The reason is that the evaluation of RAL Core expressions depends exclusively on the organisational model instead of depending on run-time information. Therefore, in terms of the KB this operation can be defined as follows:

- At design time, the activity a is consistent if there is at least one person p such that $isPotentialResponsible(p,a)$. In DL this can be expressed as:

$$satisfiable(a \in \geq 1 isPotentialResponsible^{-}.Person)$$

- At run time, the approach is similar to the one used in operation potential activities, i.e., activity a is consistent if there is an activity instance of type a that satisfies two restrictions: there is at least one person p that can be responsible for it and it belongs to the $currentInstance$. In DL, this can be expressed as:

$$satisfiable(a \in \exists isOfType^{-}.(\geq 1 isPotentialResponsible^{-}.Person \sqcap \exists hasActivityInstance^{-}.\{currentInstance\}))$$

RAL Data. In this case, the three outcomes are possible. Outcomes *yes* and *no* are possible because RAL Data includes RAL Core and outcome *partial* is possible because of expressions such as `REPORTS TO PERSON IN DATA FIELD do.f`, in which the consistency depends on whether the person referenced in the data field is reported by someone or not. To find out whether the activity is inconsistent it suffices to do the same checks as with RAL Core. If the result is *no* then it is inconsistent. However, to distinguish between consistent and partially consistent, it is necessary to check the expressions for every possible value in the data field, e.g. for every person in the organisation. If a has at least one potential performer for each possible value in the data field, then a is consistent. Otherwise, it is partially consistent. In the following we detail how to check the consistency for `PERSON IN DATA FIELD` and then, we explain the changes that have to be done to consider `IN DATA FIELD` associated with group resources.

Checking the consistency can be done in two steps. First, we need to define a new mapping that allows checking the condition of consistency for every person. Let map_c^p be a mapping equivalent to map_c^{DT} except for the fact that all the expressions related to person constraints in Table §9.1 map to $\{p\}$, where $p \in Person$, and let map_e^p be a mapping equivalent to map_e except for the fact that all expressions that use map_c should use map_c^p instead.

Algorithm 1 $isConsistent^{DT}$: Checks the design-time consistency of an activity a with regard to a set of person P.

1: **IN:** $a \in Activity$, $P \sqsubseteq Person$
2: **OUT:** *boolean*
3: $consistent \leftarrow true$
4: remove axiom $\exists isPotentialResponsible.\{a\} \equiv map_e^{DT}(expr_a^r)$ from KB
5: **for all** $p \in P$ **do**
6: add axiom $\exists isPotentialResponsible.\{a\} \equiv map_e^p(expr_a^r)$ to KB
7: $consistent \leftarrow consistent \wedge satisfiable(a \in \geq 1\ isPotentialResponsible^{-}.Person)$
8: remove axiom $\exists isPotentialResponsible.\{a\} \equiv map_e^p(expr_a^r)$ from KB
9: **end for**
10: **return** *consistent*

Second, we need to use this mapping as a way to replace the axiom that defines the potential performers of activity a with a new axiom specific for each person in the organisation. This is exactly what Algorithm 1 does. It receives an activity a and a set of persons P and returns true if a has at least one potential performer for each possible person $p \in P$. Otherwise, it returns false. The behaviour of the algorithm is quite intuitive. We first remove the current axiom from the KB. Then, it is necessary to check for each person of p that there is at least one potential performer for a. To do so, we process every person p in the following way:

- We add an axiom to the DL-based KB to establish p as the person that came in the data field (line 6).
- We check that there is at least one potential performer for activity a with this new KB (line 7).
- We remove the axiom previously added (line 8).

Therefore, at design time, an activity a is consistent if $isConsistent^{DT}(a, Person)$ is true. If a is not inconsistent and a is not consistent, then a is partially inconsistent. At run time, the reasoning is the same, the only difference being that the algorithm (cf. Algorithm 2) changes the axioms it removes and adds according to Section §9.2.3.

This way of checking the consistency can be extended to deal with `IN DATA FIELD` associated with group resources, just by modifying map_c^p and Algorithms 1 and 2 to deal with positions, roles or units. For instance, in the case of roles, there would be a map_c^r that changes all constraints related to roles to $\{r \in Role\}$ and there would be two new algorithms similar to Algorithms 1 and 2, but that receives a set of roles and iterate through them.

Algorithm 2 $isConsistent^{RT}$: Checks the run-time consistency of an activity a with regard to a set of person P.

1: **IN:** $a \in Activity$, $P \sqsubseteq Person$
2: **OUT:** *boolean*
3: $consistent \leftarrow true$
4: $ai \leftarrow individual(\exists isTypeOf.\{a\}$
$\sqcap \exists hasActivityInstance^{-}.\{currentInstance\}$
$\sqcap \exists isInState.BeforeAllocation)$
5: remove axioms $\exists potResponsible.\{ai\} \equiv map_e^{RT}(expr^r_{typeof(ai)})$ from KB
6: **for all** $p \in P$ **do**
7: add axiom $\exists potResponsible.\{ai\} \equiv map_e^{p}(expr^r_{typeof(ai)})$ to KB
8: $consistent \leftarrow consistent$
$\wedge satisfiable(a \in \exists isOfType^{-}.(\geq 1\, isPotentialResponsible^{-}.Person$
$\sqcap \exists hasActivityInstance^{-}.\{currentInstance\}))$
9: remove axiom $\exists potResponsible.\{ai\} \equiv map_e^{p}(expr^r_{typeof(ai)})$ from KB
10: **end for**
11: **return** *consistent*

Note that we assume that in each RAL expression there is only one constraint of type `IN DATA FIELD`. In case there are several of them, the mechanism to check the consistency is the same but map_c^p and Algorithms 1 and 2 should be extended to allow modifying each constraint separately.

RAL AC and RAL History. In this case, the three outcomes are also possible because of the same reasons as RAL Data. Furthermore, the approach to check the consistency is exactly the same as with RAL Data except for the fact that instead of trying with every possible person that may be referred from a data value, it is necessary to try with every possible person that may be performer of the activity related in the access-control constraint.

9.3.4 Non-participants

This operation receives a set of activities of a BP ($a_1,\ldots,a_n \in Activity$) and a task duty (e.g. *Responsible*) and returns the set of persons that can never participate in them performing such a task duty, if any. In terms of the KB:

- At design time, the non-participants of a set of activities ($a_1,\ldots,a_n$) with task duty Responsible are those persons p that do not have a property $isPotentialRespon$-

$sible(p,a)$ with any $a \in \{a_1,\ldots,a_n\}$. In DL, this can be expressed as the result of:

$$individuals(\neg\exists isPotentialResponsible.\{a_1,\ldots,a_n\})$$

- At run time it is similar, but using property $potResponsible$ and setting the context to the current process instance ($currentInstance$):

$$individuals(\neg\exists potResponsible.(\exists isOfType.\{a_1,\ldots,a_n\} \sqcap \exists hasActivityInstance^{-}.\{currentInstance\}))$$

Note that if all possible activity instances of an activity has been allocated, the non-participants of such activity are those people that has not been allocated as responsible for any of their activity instances.

9.3.5 Permanent Participants

This operation receives a set of activities of a BP ($a_1,\ldots,a_n \in Activity$) and a task duty (e.g. *Responsible*) and returns the set of persons that can participate in all of those activities performing such a task duty, if any. In terms of the KB:

- At design time, the permanent participants of a set of activities ($a_1,\ldots,a_n$) with task duty Responsible are those persons p that have a property $isPotentialResponsible(p,a)$ with all $a \in \{a_1,\ldots,a_n\}$. In DL, this can be expressed as the result of:

$$individuals(\exists isPotentialResponsible.\{a_1\} \sqcap \ldots \sqcap \exists isPotentialResponsible\{a_n\})$$

- At run time the changes are similar to previous operations:

$$individuals(\exists potResponsible.(\exists isOfType.\{a_1\} \sqcap \exists hasActivityInstance^{-}.\{currentInstance\}) \sqcap \ldots \sqcap \exists potResponsible.(\exists isOfType.\{a_n\} \sqcap \exists hasActivityInstance^{-}.\{currentInstance\}))$$

As in previous operations, if all possible activity instances of an activity has been allocated, the permanent participants of such activity are the person that has been allocated as responsible for all of their activity instances, if any.

9.3.6 Critical Participants

This operations receives a set of activities of a BP ($a_1,\ldots,a_n \in Activity$) and a task duty (e.g. *Responsible*) and returns the critical participants for the given task duty in those activities. An activity is a critical activity for a given task duty if it has only one potential performer for such a task duty. A critical participant is the potential performer of a critical activity.

Algorithm 3 $CriticalParticipants^{DT}$: Algorithm to calculate the critical participants of a process at design time

1: **IN:** $\{a_1,\ldots,a_n\} \in Activity$
2: **OUT:** $criticalParticipants \sqsubseteq Person$
3: $allActivities \leftarrow \{a_1,\ldots,a_n\}$
4: $nonCriticalActivities \leftarrow individuals(\geq 2\, isPotentialResponsible^{-}.Person \sqcap allActivities)$
5: $criticalParticipants \leftarrow \emptyset$
6: **for all** $activity \in allActivities$ **do**
7: **if** $activity \notin nonCriticalActivities$ **then**
8: $criticalParticipants \leftarrow criticalParticipants \sqcup individuals(\exists isPotentialResponsible.\{activity\})$
9: **end if**
10: **end for**
11: **return** $criticalParticipants$

- In this case, at design time a small algorithm (cf. Algorithm 3) is necessary to implement the operation. Since DL reasoners work with the open world assumption, it is not possible to prove in all cases that the number of participants is exactly one. However, it is possible to check the minimum number of participants of a given activity. Therefore, the algorithm considers as critical activities as those that do not have at least two potential performers of task duty Responsible (line 4). Then, it returns as critical participants as those individuals that are potential performers of task duty Responsible of such activity (line 8).

- At run time the approach is exactly the same, the only difference being the DL expressions used in lines 3, 4 and 8. Line 3 should be changed to consider only the activity instances that have not been allocated yet in the BP since they are the only activities that can be critical:

$$\begin{aligned} allActivities \leftarrow individuals(\, & \exists isOfType.\{a_1,\ldots,a_n\} \\ & \sqcap \exists hasActivityInstance^{-}.\{currentInstance\} \\ & \sqcap \exists isInState.BeforeAllocation) \end{aligned}$$

Lines 4 and 8 should be changed to use the run-time property *potResponsible* instead of the design-time property *isPotentialResponsible*:

$$nonCriticalActivities \leftarrow individuals(\geq 2\ potResponsible^{-}.Person \sqcap allActivities)$$

$$criticalParticipants \leftarrow criticalParticipants \sqcup individuals(\exists potResponsible.\{activity\})$$

9.3.7 Indispensable Participants

This operations receives a task duty (e.g. *Responsible*) and returns the indispensable participants for the given task duty. A person is an indispensable participant if it is the critical participant of a mandatory activity, i.e., an activity that always takes place in the BP. Therefore, the indispensable participants are those people without whom the BP always gets blocked. The approach to implement this operation both at design time and at run time is the same as in the previous operation the only difference being the initial set of activities considered (line 3), namely:

- At design time line 3 should be changed to consider only the activities that are mandatory, which are those activities a_i that are mandatory from the initial activity of the BP ($mandatory(initialActivity, a_i)$):

$$allActivities \leftarrow individuals(\exists mandatory^{-}.(\exists initialActivity.\{bp\}) \sqcap \{a_1, \ldots, a_n\})$$

- At run time line 3 should be changed to consider only the activities that are mandatory from the current state of the execution of the process instance (i.e. those activities that are in state *StartState*):

$$\begin{aligned} allActivities \leftarrow individuals(&\exists isOfType.\{a_1, \ldots, a_n\} \\ &\exists isOfType.(\exists mandatory^{-}.(\exists isOfType^{-}. \\ &\quad (\exists hasActivityInstance^{-}.\{currentInstance\} \\ &\quad \sqcap \exists isInState.StartState))) \\ &\sqcap \exists hasActivityInstance^{-}.\{currentInstance\} \\ &\sqcap \exists isInState.BeforeAllocation) \end{aligned}$$

9.4 SUMMARY

In this chapter we have introduced a reference implementation for the Person-Activity analysis operations that were identified and defined in Section §5.3.1. Specifically, we have used RAL semantics as ground, extending its DL-based KB with concepts and properties to support all the possible casuistry, that is, the execution of the operations with RAL Core, RAL Data, RAL AC and RAL History, both at design time and at run time.

10

Automated Person-Data Analysis Operations

"The more one analyses people, the more all reasons for analysis disappear. Sooner or later one comes to that dreadful universal thing called human nature"

Oscar Wilde (1854–1900),
Poet, novelist, dramatist and critic

The goal of this chapter is to provide a reference implementation of the Person-Data analysis operations, focused on the relationship between people and data. In Section §10.1 we overview the main considerations regarding these analysis operations and describe our overall approach, which is similar to the one described in the previous chapter. In Section §10.2, we describe the additional information that must be added to the KB to automate the operations. This information is based on the BP2OLC procedure to extract object lifecycles from business processes. In Section §10.3, a DL-based implementation of each analysis operation is presented. Finally, in Section §10.4 we sum up the content of the chapter.

10.1 Introduction

The Person-Data analysis operations focus on extracting information from the relationships between persons and data. In this thesis, these relationships are understood in terms of data access, i.e., which people can read or write, or both, a given data object that is used in a business process. Furthermore, since a given data object may be in different states (e.g. a proposal may be in states *created*, *evaluated*, *approved*, or *cancelled*), the analysis can be also done at the level of data states, i.e., which people can either read, write or both a given data object that is in a given data state.

The goal of this chapter is to describe a reference implementation to automate these analysis operations. The approach we follow is the same as in the previous chapter; the operations are implemented by expressing the analysis operations in terms of DL reasoning operations and then, we leverage off-the-shelf DL reasoners to implement them. Specifically, since the relation between persons and data is done through activities, the analysis is built upon both the explicit information provided by the BP model regarding data inputs and outputs of activities, and the information about the potential performers of activities, which has been exhaustively discussed in the previous chapter.

However, together with the additional information that was added in the previous chapter, it is also necessary to include more information regarding the access of activities to data objects in a given state since this information is may not be explicitly provided by the BP model. To this end, we have developed BP2OCL [31], which is a model-driven procedure to automatically transform from the usual activity-centered model of a BP in which the focus is on defining the control flow of the activities, to the set of lifecycles of the data objects involved in the process (i.e., a data-centered model of the BP). The output of this procedure is then processed to include the information to the KB.

10.2 Extending the KB with Data Information

The starting point of the DL-based KB is the mapping introduced in Sections §8.2 and §8.3, and the extension of the KB detailed in Section §9.2. However, automating the analysis operations related to data also requires knowing which activities can be executed between two states of a data object in order to calculate the persons that can

access that information during that period. The BP2OLC procedure, as depicted in Figure §10.1, is a three-step procedure grounded on model transformations that allows one to obtain such information. Then, after finishing the BP2OLC procedure RAL's KB can be extended with the information included in the *Object Life Cycle (OLC)* (s).

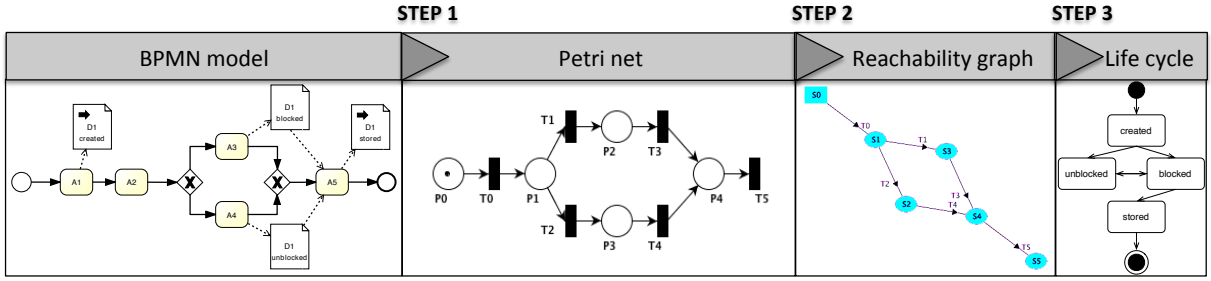


Figure 10.1: Overview of the BP2OLC procedure

The BP2OLC procedure involves four different models. The input is a BP model represented in BPMN 2.0 [94], and the output is the set of OLCs of all the data objects that are involved in the process. Specifically, the output of the BP2OLC procedure is a set of *Finite State Machines (FSMs)* representing the lifecycles of the data objects modelled in a BP. Figure §10.2 illustrates the lifecycle of data object *Resolution* of the BP mode we introduced as example in Section §2.2 (cf. Figure §2.1). As depicted in the figure, an OLC has one *start state* (represented with a filled circle), one *final state* (represented with a semi-filled circle), and one or more *intermediate states* (represented with a rectangle) that correspond with states of the data object in the BP model. Transitions (represented with directed arrows) connect two states and contain the parts of the process that are executed in the transition between states of the data object.

Definition 10.1.
An *object lifecycle of a data object of a BP* is a 2-tuple $OLC = (S_{OLC}, T_{OLC})$, where:

- $S_{OLC} = \{s_1, s_2, ..., s_n\}$ is the set of states in which the data object can be. $\forall s_i \in S_{OLC}, \bullet s_i$ and $s_i \bullet$ represent immediately previous and next states of state s_i, respectively. Let $start \in S$ and $end \in S$ be the start and the final states of the OLC, respectively. Then, $S_{OLC} \setminus (start \bigcup end) = P_D = D_{BP}$
- $T_{OLC} \subseteq S_{OLC} \times S_{OLC} \times \mathcal{P}(N)$ is the set of transitions that appear in the object lifecycle. Each transition contains a set of nodes of the reachability graph from which

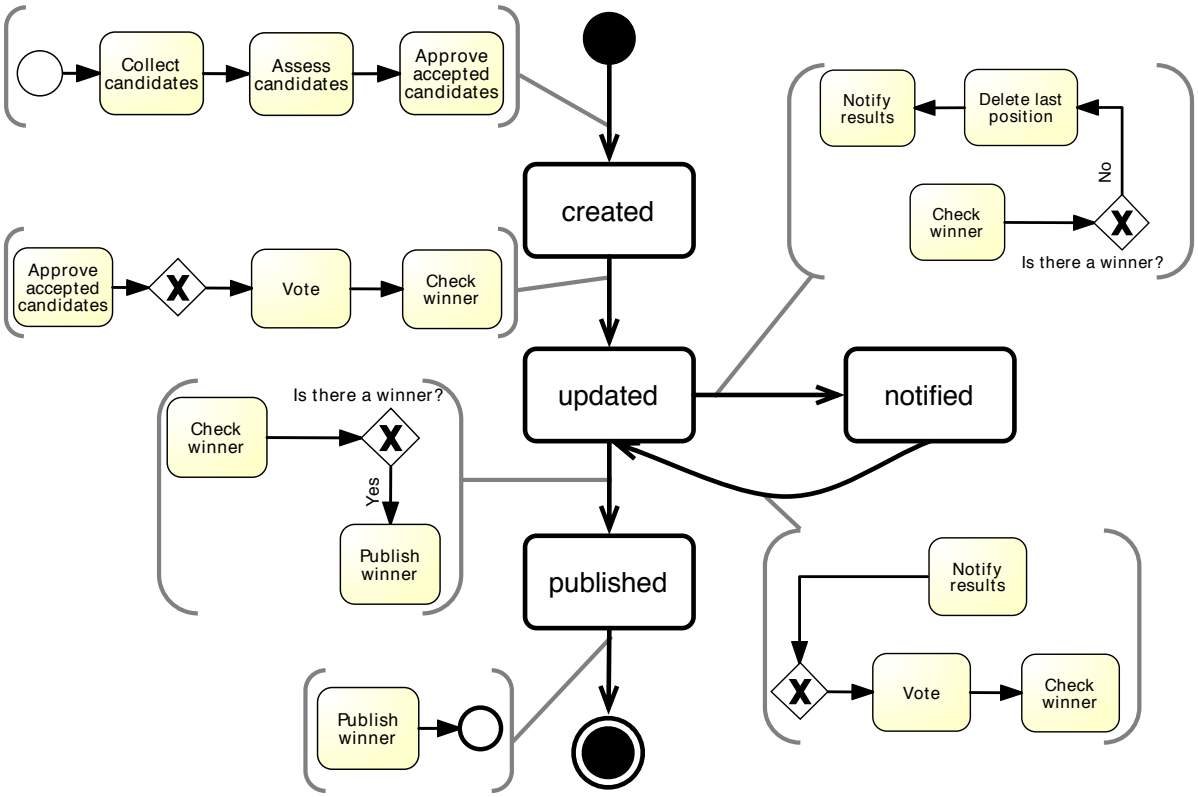


Figure 10.2: Object lifecycle of data object *Resolution*

it has been generated. Function $replace : T_{OLC} \times N \times \mathcal{P}(N) \rightarrow T_{OLC}$ replaces the set of nodes before node N in the path of a transition for a specific set of nodes.

Therefore, an OLC represents the allowed transitions between the states of a data object according to the BP diagram, where the transitions also include information about the activities of the process that are executed during the transition between two states of the data object.

Please, notice that the BP2OLC procedure must be carried out for each *data object type* present in the source BP model, and that we make the following assumptions:

1. Regarding the control flow, the BP model is *sound*, which means it has no control flow deadlocks and terminates properly [129].

2. There is only one copy of each data object in each BP instance, e.g. for the process in Figure §6.4 there is only one data object *Travel Authorization* in one instance

of the process. Besides, data objects are created within the BP instance that uses them (i.e. data objects created outside the process are not considered).

3. Each data object has *always* a state. In case an appearance of a data object in the process model is not associated with any state, this appearance will be ignored. Besides, data state identifiers are unique, that is, a data state is uniquely related to a data object.

4. The BP model can contain data objects connected to any kind of activity (sub-processes are treated like task activities). Only XOR gateways can be used.

Assumption 1 is made because checking control flow soundness is out of the scope of the procedure. Assumptions 2 and 3 are reasonable and have also been made elsewhere [16]. The last assumption implies that we do not limit the type of activities considered to takes (according to BPMN 2.0 vocabulary [94]).

10.2.1 Step 1. From BPMN Model to Petri Net

We believe that providing a semantic mapping [73] between a BPMN model and a target domain such as Petri Nets, whose semantics has been formally defined, is a good approach because it allows one to use the techniques specific to the target semantic domain for analysing the source models. We chose Petri Nets for two reasons: (i) plenty of processing algorithms on Petri Nets have already been developed and can be useful for our purpose [89, 129]; and (ii) the transformation of the control flow of a BP model into an equivalent Petri Net has already been described by van der Aalst [129].

Definition 10.2.
A *Petri Net* is a 3-tuple $PN = (T_{PN}, P, F)$, where:

- $T_{PN} = \{t_1, t_2, ..., t_n\}$ is the set of transitions of the Petri Net, represented graphically as rectangles.

- $P = \{p_1, p_2, ..., p_n\}$ is the set of places of the Petri Net, represented graphically as circles.

- $F \subseteq (P \times T_{PN}) \bigcup (T_{PN} \times P)$ is the set of arcs of the Petri Net (flow relation), represented as arrows.

A *marking (state)* or *markup* assigns a nonnegative integer to each place of a Petri Net. If it assigns to place p a nonnegative integer k, we say that p is marked with k tokens. Pictorially, we place k black dots (tokens) in place p. A markup is denoted by M, an m-vector, where m is the total number of places. The *pth* component of M, denoted by *M(p)*, is the number of tokens in place p. The firing of an enabled transition will change the token distribution (marking) in a net [89].

We use the set of rules introduced by Awad et al. [16] to do the semantic mapping between elements of a BP model with data objects and elements of a Petri Net. Let E_{BP} be the set of flow nodes of a BP (model), i.e. activities, gateways and events, D_{BP} the set of states of a data object of that BP, and $WRITERS_{BP} \subseteq E_{BP}$ be the set of activities of the BP that write that data object. The result of the semantic mapping is a Petri Net with the following characteristics:

- The places of the Petri Net are of two different kinds: control places P_C and data places P_D. Therefore $P = P_C \bigcup P_D$ and $P_C \bigcap P_D = \emptyset$.
  - $P_C = \{pc_1, pc_2, ..., pc_n\}$ corresponds to those places that represent sequence flow elements (arrows) of the business process. Each $pc_i = (ei_i, eo_i)$, where $ei_i, eo_i \in E_{BP}$ is a pair of values composed of the two flow nodes of the business process that the sequence flow element connects.
  - $P_D = \{pd_1, pd_2, ..., pd_n\} = D_{BP}$ corresponds to those places that represent states of the data object whose object lifecycle we are generating. There is exactly one data place for each possible state of the data object.
- The transitions of the Petri Net represent flow nodes of the business process model. It follows an $n : 1$ relationship, i.e., each transition represents only one flow node of the business process and a flow node may appear several times in a Petri Net. Function $elem : T_{PN} \rightarrow E_{BP}$ represents such relation.

An example of the transformation rules is depicted in Table §10.1, which illustrates an extension of the catalogue of transformations proposed in [16] to deal with loop activities. As stated in [94], a loop activity executes the inner activity as long as a loop condition evaluates to true. An attribute can be set to specify a maximal number of iterations. An example of loop activity is an activity *Update order* that updates an order in a restaurant (by customer's command) until an event or a received message indicates

no more updates are allowed. For more details about the other transformations we refer the reader to [16].

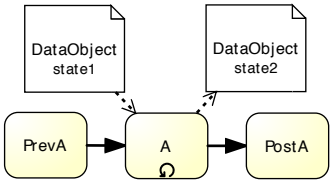
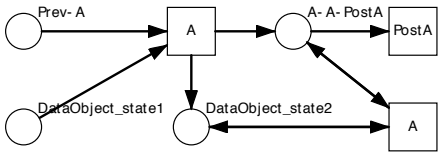
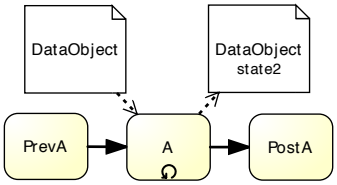
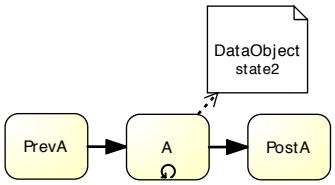
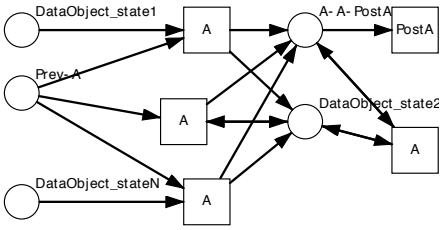
BPMN	Description	Petri Net
 a) Read-write loop activity with a condition on input state	Loop activity A has both a precondition and a postcondition on the state of the data object. The precondition will be considered only for the first execution of A. Then the object is assumed to be in state2.	
 OR  b) Read-write loop activity without a condition on input state	Loop activity A has no specific condition on the state of data object as input. We assume that after first completion of A the data object will always have state2.	

Table 10.1: Mapping for data objects association with loop activities

Finally, note that there is a small difference between this mapping and the one presented by Awad et al. [16] because we consider no data objects are supposed to exist before the execution of a BP in our BP2OLC procedure, whereas Awad et al. considers data objects have an initial state when instantiating a BP. This difference causes the transformation in [16] referring to the writing of the data object has to be slightly changed for the first writing of the object in our BP2OLC procedure, in order to comply with our assumption 2. It means the first time the data object is written, the responsible transition of the Petri Net does not have any input data places.

10.2.2 Step 2. Reachability Graph from Petri Net

Definition 10.3.

A *reachability graph related to a Petri Net* is a 3-tuple $RG_{PN} = (N, M, T_{RG})$, where:

- $N = \{n_1, n_2, ..., n_n\}$ is the set of nodes of the reachability graph. $\forall n_i \in N, \bullet n_i$ and $n_i \bullet$ represent immediately previous and next nodes of n_i, respectively.
- $M : P \times N \rightarrow \mathbb{N}$ represents the *markup* of the net.
- $T_{RG} \subseteq (N \times N)$ are the transitions of the reachability graph.

The reachability graph is obtained by analysing the Petri Net by means of well-known algorithms. Each node of the reachability graph represents a reachable marking state of the net and each arc a possible change of state, i.e. the firing of a transition. However, due to the characteristics of our semantic mapping between BPMN and Petri Net, in the reachability graph resulting from such Petri Nets it holds that $M(p,n) \in [0,1], \forall n \in N, \forall p \in P$. In addition, the information about the markup of the net contained in every node always corresponds with both a sequence flow of the BP model and a state of the data object, as illustrated in Figure §10.3. It means there is always one token in a control place of the Petri Net and one in a data place, except in the beginning (until an activity writes the data object for the first time) and in the final nodes of the reachability graph (in which, on the contrary, all the tokens in control places have been consumed).

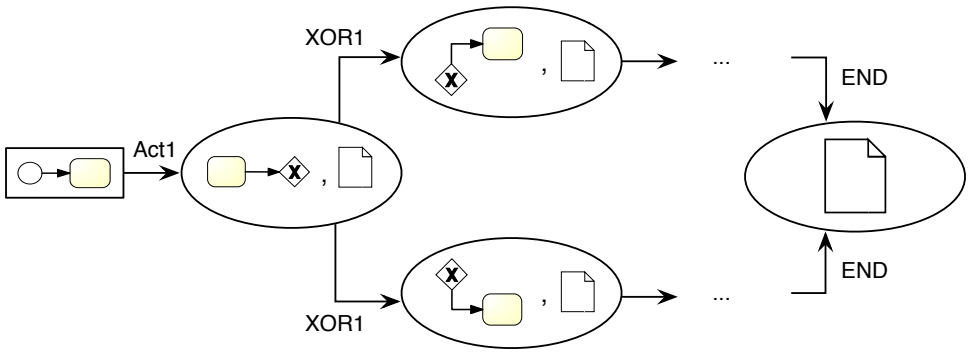


Figure 10.3: Content of the arcs and nodes of a reachability graph

Given the previous definitions, the following functions can be defined:

- Function $map : T_{RG} \rightarrow T_{PN}$ is defined to map the transitions of a reachability graph into the transitions of a Petri Net.
- Function $state : N \rightarrow P_D$ returns the state of the data object of the business process model contained in the current node of the reachability graph. $state(n) = \{p_d \in P_D : M(p_d, n) = 1\}$.

- Function $flow : \mathcal{P}(N) \rightarrow \mathcal{P}(P_C)$ returns the set of sequence flow elements of the business process model contained in a set of nodes of the reachability graph. $flow(N') = \{p_c \in P_c : \exists n \in N'(M(p_c,n) = 1))\}$.
- Function $activity : N \rightarrow E_{BP}$ returns the flow node of the business process model contained in the input arc of the current node of the reachability graph. $activity(n) = \{e_i \in E_{BP} : p_c = (e_i,e_o) \wedge M(p_c,n) = 1\}$.

The node of the reachability graph with no input arrows is called $firstNode \in N : \nexists \bullet firstNode$ and it is the start node of a reachability graph. The nodes of the reachability graph with no output arrows, whose input is called END and with no tokens in a control place are *normal* final nodes of the reachability graph. We will describe *abnormal* final nodes in next section.

10.2.3 Step 3. Object lifecycle from Reachability Graph

Algorithm 4 Algorithm to initialize an object lifecycle, call Algorithm 5 from a reachability graph and post-process nodes already processed in Algorithm 5 (RG2OLC)

IN: $RG_{DPN} = (N,M,T_{RG}); WRITERS_{BP}$
OUT: $S_{OLC}; T_{OLC}; WARN \subseteq N$
$S_{OLC} \leftarrow \{START\_STATE\}; T_{OLC} \leftarrow \emptyset$
$INPUT \leftarrow (WRITERS, firstNode, START\_STATE, \emptyset, \emptyset, \emptyset, \emptyset, S_{OLC}, T_{OLC})$
$(S_{OLC}, T_{OLC}, PNODES, PP, WARN) \leftarrow RG2OLC(INPUT)$
$found \leftarrow 1$ {P}ost-processing of nodes in PP
while $found \neq 0$ **do**
  $found \leftarrow 0$
  for all $(node, assocPath) \in PP$ **do**
    for all $(s_i, s_o, path) \in T_{OLC}$ **do**
      if $node \in path$ **then**
        $found \leftarrow found + 1; newT \leftarrow (s_i, s_o, path)$
        $T_{OLC} \leftarrow T_{OLC} \bigcup replace(newT, node, assocPath)$
      end if
    end for
  end for
end while
return $(S_{OLC}, T_{OLC}, WARN)$

We have defined Algorithms 4 and 5 to obtain an OLC from a reachability graph. Algorithm 4 receives the reachability graph resulting from the previous step and the list of activities of the BP that write the data object. Its output is the OLC together with

Algorithm 5 Algorithm to generate the lifecycle of a data object from a reachability graph (RG2OLC)

1: **IN:** $WRIT, cNode, cState, PNODES, PATH, PP, WARN, S'_{OLC}, T'_{OLC}$
2: **OUT:** $S'_{OLC}, T'_{OLC}, PNODES, PP, WARN$
3: **if** $state(cNode) \neq \emptyset \quad \wedge \quad (cState \neq state(cNode) \quad \vee \quad activity(cNode) \in WRIT)$ **then**
4: $\quad S'_{OLC} \leftarrow S'_{OLC} \bigcup state(cNode)$
5: $\quad T'_{OLC} \leftarrow T'_{OLC} \bigcup (cState, state(cNode), PATH)$
6: $\quad cState \leftarrow state(cNode); PATH \leftarrow cNode$
7: **end if**
8: $PATH \leftarrow PATH \bigcup cNode$
9: **if** $cNode \notin PNODES$ **then**
10: $\quad PNODES \leftarrow PNODES \bigcup cNode$
11: $\quad$ **if** $cNode\bullet = \emptyset$ **then**
12: $\quad\quad$ **if** $activity(cNode) = END$ **then**
13: $\quad\quad\quad S'_{OLC} \leftarrow S'_{OLC} \bigcup FINAL\_STATE$
14: $\quad\quad\quad T'_{OLC} \leftarrow T'_{OLC} \bigcup (cState, FINAL\_STATE, PATH)$
15: $\quad\quad$ **else**
16: $\quad\quad\quad WARN \leftarrow WARN \bigcup cNode$ {Deadlock detected}
17: $\quad\quad$ **end if**
18: $\quad$ **else**
19: $\quad\quad$ **for all** $next \in cNode\bullet$ **do**
20: $\quad\quad\quad IN \leftarrow (WRIT, next, cState, PNODES, PATH, PP, WARN, S'_{OLC}, T'_{OLC})$
21: $\quad\quad\quad (S''_{OLC}, T''_{OLC}, PNODES', PP', WARN') \leftarrow RG2OLC(IN)$
22: $\quad\quad\quad S'_{OLC} \leftarrow S'_{OLC} \bigcup S''_{OLC}; T'_{OLC} \leftarrow T'_{OLC} \bigcup T''_{OLC}; PP \leftarrow PP \bigcup PP'$
23: $\quad\quad\quad PNODES \leftarrow PNODES \bigcup PNODES'; WARN \leftarrow WARN \bigcup WARN'$
24: $\quad\quad$ **end for**
25: $\quad$ **end if**
26: **else**
27: $\quad$ **if** $activity(cNode) \notin WRIT$ **then**
28: $\quad\quad PP \leftarrow PP \bigcup (cNode, PATH)$ {Save for post-processing}
29: $\quad$ **end if**
30: **end if**
31: **return** $(S'_{OLC}, T'_{OLC}, PNODES, PP, WARN)$

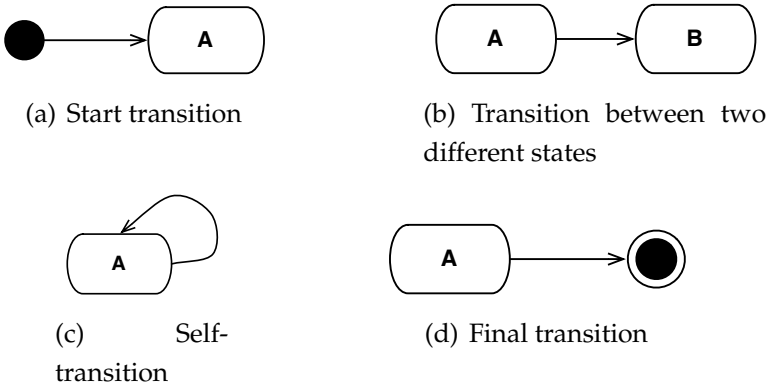
a set of data anomalies found while creating it. Its behaviour consists of calling Algorithm 5 with the appropriate parameters and post-processing the resulting reachability graph. Algorithm 5 is a recursive algorithm that builds an OLC by processing a reachability graph node by node from its start node. Its input set and steps are described below.

Input of Algorithm 5.

- $WRIT \subseteq E$ is the set of activities that write the data object.

- $cNode \in N$ is the node being processed.
- $cState \in D$ is the current state of the data object.
- $PNODES \subseteq N$ is the set of already processed nodes.
- $PATH \subseteq N$ contains a set of nodes of the reachability graph, which is the information required in the transitions of the object lifecycle.
- $PP = \{pair_1, pair_2, ..., pair_n\}$, where $pair_i = (node, assocPath), node_i \in N$, $assocPath_i \subseteq N$ is a set of pairs containing a node of the reachability graph and a set of nodes associated to that node, which conceptually corresponds to the path contained in variable PATH when processing that node.
- $WARN \subseteq N$ is a set of nodes related to deadlocks in the Petri Net.
- $S'_{OLC} \subseteq S_{OLC}$ is the set of states of the resulting object lifecycle.
- $T'_{OLC} \subseteq T_{OLC}$ is the set of transitions of the resulting object lifecycle.

Check for and add new transitions (lines 3-7). A new transition of one of the types shown in Figures 10.4(a) and 10.4(b) must be added to the OLC in case that a new state of the data object is found in the reachability graph. If, on the contrary, the node shows that the data object is still in the current state but we find that the activity of the node is one of those that write the data object in the BP model, a self-transition will be added (cf. Figure 10.4(c)). For instance: (i) in loops in a BP a data object may be written by an activity consecutively twice, giving rise to a self-transition; (ii) loop activities also cause self-transitions, as can be inferred from Table §10.1.



(a) Start transition

(b) Transition between two different states

(c) Self-transition

(d) Final transition

Figure 10.4: Types of transitions of an object lifecycle

Update variable *PATH* **(line 8).** New nodes will be added to the path in order to collect the information contained in the transitions of the lifecycle[1].

Revise the last activity executed and act consistently (lines 9-31). Algorithm 5 is returned either when a *normal* final node of the reachability graph is reached, when an *abnormal* final node[2] is found, or when the current node is in the list of processed nodes. In the last case, if, furthermore, the activity represented in the node writes the data object, the node has been properly processed in lines 3-7 and the rest of the reachability graph does not have to be re-processed. Otherwise, the already processed node is saved in a list of nodes that must be properly post-processed later. This way we avoid processing nodes of the reachability graph more than once and we ensure that Algorithm 5 always terminates. If none of the previous situations appears, we must go on processing the reachability graph and update the variables whose values must be propagated.

Post-process the necessary nodes (lines 6-17 of Algorithm 4). Some scenarios represented in a BP model can give rise to the appearance of transitions between the same two states, which differ from each other in their contents. This situation is detected in the reachability graph when reaching a node that has already been processed and its corresponding activity of the BP does not modify the data object being examined. In Algorithm 5 only one of the transitions is added to the OLC. To add the new transition with the right content, we must find the transition to which the node refers, add a duplicate transition to the OLC and set its content to the proper value.

10.2.4 Adding Information related to OLC

Once we obtain the OLCs of the data objects of a BP, RAL's KB must be extended with the information included in the OLC (s). To do so, two new object properties are added to the KB to represent the relation between an activity and the states of the data object(s) it handles, so that an activity reads and/or writes a data object when it is in one or more data states. Without these explicit properties, information about what the state of a data object is when it is read or written, is missing. The relation with the rest of concepts of the KB is depicted in Figure §10.5.

[1] Note that operator *union* ($\bigcup$) neither inserts duplicates nor null or empty values.

[2] *Abnormal* final nodes are those with no output transitions and with no input transitions called END. They indicate there is a deadlock in the Petri Net that stops the execution. We collect them in a list of warnings that will have to be addressed later.

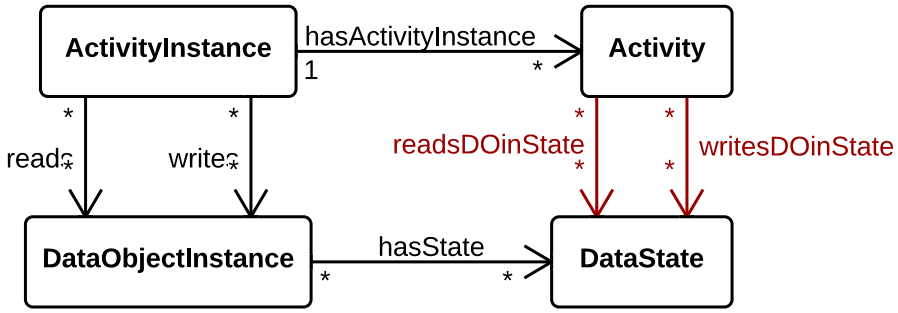


Figure 10.5: Properties for mapping an OLC into a DL-based KB (in red)

Algorithm 6 shows the algorithm to automatically instantiate the KB with regard to these new elements from an OLC generated by the BP2OLC procedure. It works as follows. Each transition in an OLC contains a set of nodes of the reachability graph from which it has been generated and each node corresponds to a BP element, some of which are activities. So, for each activity contained in each transition of the OLC, if the activity has input data object(s) and the set involves the data object in question, a *readsDOinState* is added to the KB to indicate that the activity reads a data object in a specific data state (line 5). The state is the input state of the transition in the OLC. Similarly, for each activity contained in each transition of the OLC, if the activity has a set of output data object(s) associated and it involves the data at hand, a *writesDOinState* is added to the KB to indicate that the activity writes a data object in a specific data state (line 8). The state is the output state of the transition in the OLC.

Algorithm 6 Algorithm to introduce the information in the transitions of an OLC like the one in Figure §10.2 into RAL's DL-based KB according to Figure §10.5

1: **IN:** S_{OLC};T_{OLC};d_i
2: **for all** $(s_i, s_o, path) \in T_{OLC}$ **do**
3: **for all** $activity \in path$ **do**
4: **if** $satisfies(activity \in \exists inputDataObject.\{d_i\})$ **then**
5: add axiom $readsDOinState(activity, s_i)$ to the KB
6: **end if**
7: **if** $satisfies(activity \in \exists outputDataObject.\{d_i\})$ **then**
8: add axiom $writesDOinState(activity, s_o)$ to the KB
9: **end if**
10: **end for**
11: **end for**
12: **return** KB

10.3 Automating Analysis Operations Using DL

In this section we provide a reference implementation based on DLs for each analysis operation focused on checking aspects related to data access control. Data objects are involved in all these operations, although in a different way than in the case of the operations related to activities. In particular, data access control operations are not related to the content of the data objects, but to the relationship between activities and data objects. Besides, the implementation must have two aspects into account. On the one hand, the implementation of the analysis operation may slightly change whether the data object have states defined or do not. On the other hand, like the other analysis operations, they may be executed either at design time or at run time.

Note that in the following definitions we assume that each person that performs a task duty in an activity except for task duty informed should have read access to the data objects that are input to the activity and write access to the data objects that are output to the activity. Therefore, to make the formulation of the operations simpler, we add two new properties into the KB called *isPotentialInvolved* and *potInvolved*, respectively. The former has *Person* as domain, *Activity* as range and is a super property of properties *isPotentialResponsible*, *isPotentialAccountable*, *isPotentialSupport* and *isPotentialConsultant*. The latter has *Person* as domain, *ActivityInstance* as range and is a super property of properties *potResponsible*, *potAccountable*, *potSupport* and *potConsultant*. An advantage of defining this abstraction is that if we decide to change the criteria regarding which task duties have access to data objects, only the definition of *isPotentialInvolved* and *potInvolved* must change.

10.3.1 Potential Accessors to Data States

This operation receives a set of data states $ds_1,\ldots,ds_n$ and an access type (read, write, or read/write) and returns the persons that are allowed to access the data object in such data states. Obviously, this operation only makes sense for data object that have data states defined. In terms of DL the operation can be formulated as follows:

- At design time, the persons that have read access to data states $ds_1,\ldots,ds_n$ are those persons that are potentially involved in an activity that has one of the given

data states as input. In DL this can be expressed as the result of:

$$individuals(\exists isPotentialInvolved.(\exists readsDOinState.\{ds_1\}) \sqcap \cdots \sqcap \exists isPotentialInvolved.(\exists readsDOinState.\{ds_n\}))$$

Once we have defined the expression for read access, it is straightforward to extend it to write access:

$$individuals(\exists isPotentialInvolved.(\exists writesDOinState.\{ds_1\}) \sqcap \cdots \sqcap \exists isPotentialInvolved.(\exists .writesDOinState.\{ds_n\}))$$

For read and write access to the data states, the DL expression is the intersection of the people with read access and people with write access.

- At run time the persons that have read access to data states $ds_1, \ldots, ds_n$ are those persons that are potentially involved or have been involved in an activity that has one of the given data states as input. In DL this can be expressed as:

$$\begin{aligned} individuals(\exists potInvolved.(\exists isOfType(\exists readsDOinState.\{ds_1\}) \\ \sqcap \exists hasActivityInstance^{-}.\{currentInstance\}) \sqcap \cdots \sqcap \\ \exists potInvolved.(\exists isOfType(\exists readsDOinState.\{ds_n\}) \\ \sqcap \exists hasActivityInstance^{-}.\{currentInstance\})) \end{aligned}$$

The definition for write access and read and write access are equivalent to those at design time.

10.3.2 Potential Accessors to Data Objects

This operation receives a set of data objects $do_1, \ldots, _n$ and an access type and returns the persons that are allowed to access all of the given data objects regardless of their state. This operation can be formulated in DL as follows:

- At design time the operation returns those persons that are potentially involved in an activity that has the given data object as input, which can be expressed as:

$$individuals(\exists isPotentialInvolved.(\exists inputDataObject.\{do_1\}) \sqcap \cdots \sqcap \exists isPotentialInvolved.(\exists outputDataObject.\{do_n\}))$$

The changes to the definition for write access and read and write access are equivalent to those in the previous operation

- At run time the operation returns those persons that are potentially involved or have been involved in an activity that has one of the given data objects as input. In DL this can be expressed as:

$$\begin{aligned} individuals(\exists potInvolved.(\exists isOfType(\exists inputDataObject.\{do_1\}) \\ \sqcap \exists hasActivityInstance^{-}.\{currentInstance\}) \sqcap \cdots \sqcap \\ \exists potInvolved.(\exists isOfType(\exists outputDataObject.\{do_n\}) \\ \sqcap \exists hasActivityInstance^{-}.\{currentInstance\})) \end{aligned}$$

  The definition for write access and read and write access are equivalent to those at design time.

10.3.3 Potential Data States Allowed for a Person

This operation receives a person p, a data object do and an access type, and returns the states of the data object do that p can access with the given access type. This operation can only be applied to data objects that have data states defined. Next we detail the formulation of the operation in DL for read access. The formulation for write access and read and write access can be straightforwardly defined as in the previous operations.

- At design time the operation returns those data states of do that are read by an activity to which p is potentially involved. This can be expressed as:

$$\begin{aligned} individuals(\exists hasState^{-}.\{do\} \sqcap \\ \exists readsDOinState^{-}.(\exists isPotentiallyInvolved^{-}.\{p\})) \end{aligned}$$

- At run time the operation can be defined similarly, but changing *isPotentiallyInvolved* with the run-time property *potInvolved*:

$$\begin{aligned} individuals(\exists hasState^{-}.\{do\} \sqcap \\ \exists readsDOinState^{-}.(\exists isOfType^{-}(\exists potInvolved^{-}.\{p\} \sqcap \\ \exists hasActivityInstance^{-}. \\ \{currentInstance\}))) \end{aligned}$$

10.3.4 Potential Data Objects Allowed for a Person

This operation receives a person p and an access type and returns the data objects that p can access with the given access type. If the data objects have data states, it returns the data objects that can be accessed by p in at least one data state. As in the previous operation, next we formulate it in terms of DL for read access. The formulation for write access and read and write access can be straightforwardly defined as in the previous operations.

- At design time the operation returns those data objects that are input of an activity to which p is potentially involved. This can be expressed as:

$$individuals(\exists inputDataObject^{-}.(\exists isPotentiallyInvolved^{-}.\{p\}))$$

- At run time the operation can be defined similarly, but changing *isPotentiallyInvolved* with the run-time property *potInvolved*:

$$individuals(\exists inputDataObject^{-}.(\exists isOfType^{-}(\exists potInvolved^{-}.\{p\} \sqcap \exists hasActivityInstance^{-}.\{currentInstance\})))$$

10.4 SUMMARY

In this chapter we have provided a reference implementation for the Person-Data analysis operations. To deal with data objects that may be in different states, we have introduced the BP2OLC procedure, which is a model-driven procedure to automatically transform from the usual activity-centered model of a BP to a data-centered model. This transformation eases the inclusion of new axioms in the KB that are necessary to formulate the analysis operations.

11

Evaluation of the Contributions. Tool Support

"All life is an experiment. The more experiments you make the better"

Ralph Waldo Emerson (1803-1882),
Poet

Our contributions are materialized in CRISTAL, a system that deals with the specification, binding and analysis of resources in BPs. It is introduced in Section §11.1, and its components are described in Section §11.2. Furthermore, the implementation of tool support has been useful to evaluate our approaches with regard to each and every problem identified in Chapter §5, as outlined in Section §11.3. Section §11.4 presents a summary of the application of some of our research results in a real scenario. Finally, a summary and conclusions drawn from the chapter are presented in Section §11.5.

11.1 INTRODUCTION

The assessment of our contributions has been mainly performed by implementing tool support for the approaches developed. This has given rise to CRISTAL[1] [32], a set of tools and languages for the management of resources in BPs, focused on the specification and analysis of the BP resource perspective. CRISTAL has enabled us to evaluate all of our proposals with respect to each and every problem identified from the state of the art.

Furthermore, part of the analysis operations addressed in this thesis have validated in a real scenario as part of a project whose main aim was tp develop a system to provide support for the automated checking of all kinds of compliance rules, resource-related rules among them.

11.2 CRISTAL

Collection of Resource-centrIc Supporting Tools And Languages (CRISTAL) is our approach to improve resource management in BPs in two main directions: the specification of resources in BP models, and the analysis of resources in BPs. Specifically, CRISTAL provides a language for the specification of resource assignments, RAL, and it offers two different forms of binding the resource specification to BP model: (i) an all-in-one binding approach based on the use of RAL with BPMN; and (ii) a separate approach that relies of the use of the so-called RACI matrix, extended with binding information defined with RAL. An automated procedure to switch from one binding strategy to the other is also provided. Besides, CRISTAL is equipped with analysis capabilities that enable the automate analysis of the BP resource perspective, taking into account not only the resource specification, but also the control flow and data perspectives, when required.

In the following, we provide an overall description of the functionalities of the system, and then we step to provide detailed descriptions of the specification and analysis modules that compose CRISTAL.

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

11.2.1 CRISTAL Overview

Figure §11.1 provides an overview of CRISTAL, in which the different modules (components) are represented in rounded rectangles coloured in red, modelling software developed by us appears in dark yellow, inputs and outputs of the components (usually documents) are linked with dashed arrows, and the interconnection between the modules and other software tools is depicted with solid arrows. The existing systems that we have extended and/or with which we have interacted, are depicted in gray. CRISTAL components can also be used in isolation from the rest of components. Let us outline the functionality of each component.

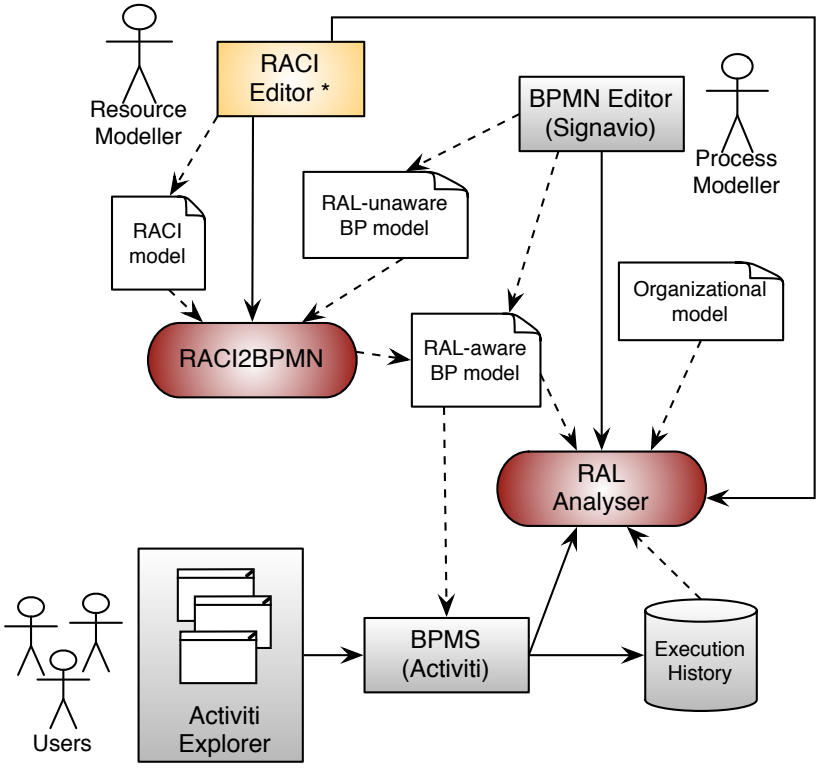


Figure 11.1: Overview of CRISTAL

The first feature provided by the system is RAL [30], a language to define resource assignments for the task duties of the process activities. As shown in Chapter §6, RAL expressions range from very simple assignments based on specific persons or organisational concepts, to compound assignments that can contain access-control constraints with respect to the assignments of other activities. The specification of resources defined with RAL can be bound to the BP models of an organisation in two different ways: either by using it with BPMN 2.0, for which CRISTAL uses Signavio

Core Components[49] as process modelling tool; or by means of the *RACI Editor*, a Web RACI matrix editor that allows specifying not only the task duties corresponding to the RASCI roles (i.e. Responsible, Accountable, Support, Consulted, Informed), but also the binding information desired, using RAL.

Pursuing binding flexibility (cf. Section §5.2.3), CRISTAL also provides functionality to automatically switch between the two binding approaches offered. In particular, the *RACI2BPMN* module implements the procedure described in Section §7.4 to shift from separate binding to all-in-one binding. This has several advantages: (i) consistency is automatically maintained between two different binding models; and (ii) BPs and resources can be specified separately, while being executed jointly. This is possible because the output of the RACI2BPMN module is a BPMN-compliant model and, thus, it can be opened in any editor that supports BPMN 2.0, and executed in any BPMS that can deal with RAL assignments.

As far as analysis is concerned, *RAL Analyser* is the component that encapsulates all the analysis functionality provided by CRISTAL. Grounded on RAL formal semantics based on DLs (cf. Chapter §8), provides the required support to automatically analyse the BP resource perspective at design time and at run time. In particular, a module of the RAL Analyser (*Design-Time RAL Solver*)[33] provides support for design-time execution of the following Person-Activity operations: Potential Performers, Potential Activities, Non-participants, Permanent Participants, and Indispensable Participants, as well as for the Comparison between the sets of potential performers of two activities. As a proof of concepts, this module has been integrated as a plugin in Signavio Core Components and has been tested in Signavio Oryx[2].

Another module of the RAL Analyser (*RT RAL Solver*) offers run-time analysis support focused on automatically calculating the set of potential performers for each activity in a process model during its execution in a BPMS. It has been implemented as a library that can be integrated and used in Activiti[3], an open-source BPMS.

11.2.2 Resource Specification in Business Processes with CRISTAL

Next, we describe how each CRISTAL component dealing with resource specification in BPs has been implemented. Regarding RAL, developing a specific tool to show how RAL can be used in BP models has not been necessary. We have used Signavio

[2] http://bpt.hpi.uni-potsdam.de/Oryx
[3] http://activiti.org/

Core Components, which uses BPMN, for that purpose. As explained in Section §7.2, BPMN allows the use of any assignment language for resource specification. Thus, using the *Resources* property associated to the process activities, RAL expressions can be assigned to the Responsible and Accountable task duties of the BPMN activities, like depicted in Figure §11.2.

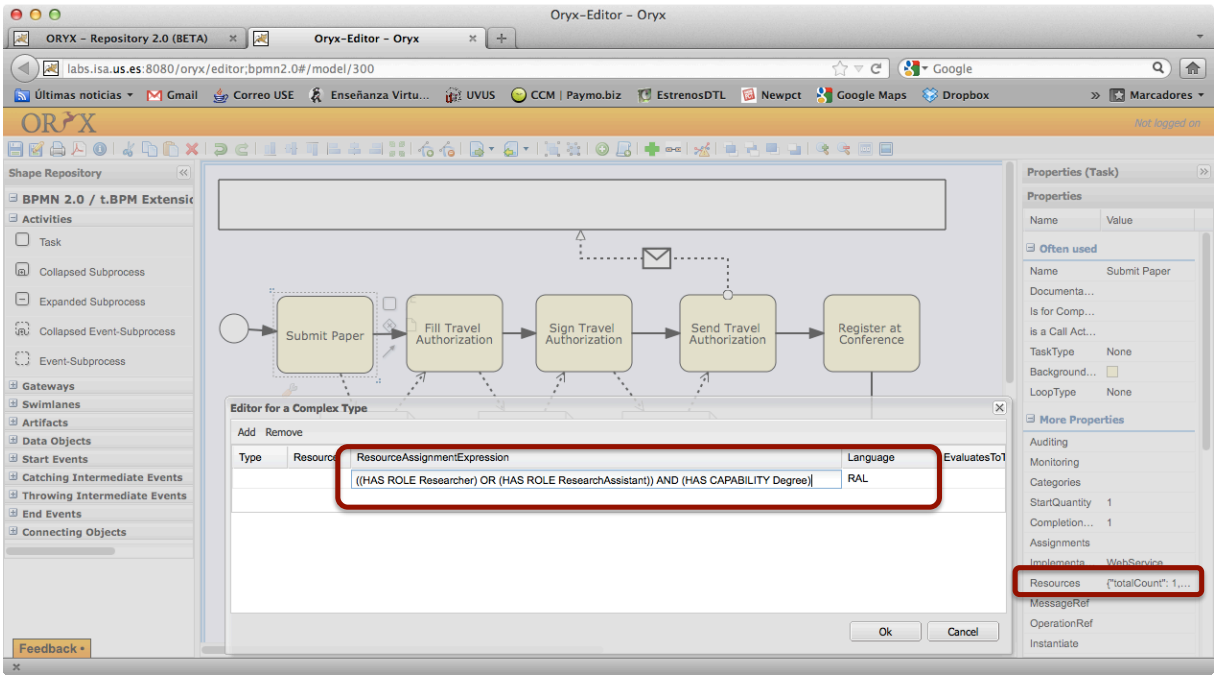


Figure 11.2: Use of RAL in Signavio Oryx

RACI Editor

The RACI Editor is a Web application developed in Java that allows the user to load a resource-unaware BPMN model and define the resource information associated to the activities by filling a RASCI matrix and, optionally, adding binding information with RAL, as shown in Figure §11.3. This application generates a JSON file with the resource-related information of the process, which can serve as input of the RACI2BPMN tool.

RACI2BPMN

The RACI2BPMN component allows changing the separate binding model composed of RASCI matrices and binding information, in an single BPMN model that keeps the same information using an all-in-one binding approach. It receives an XML file with the representation of a resource-unaware BPMN model, and a JSON file with

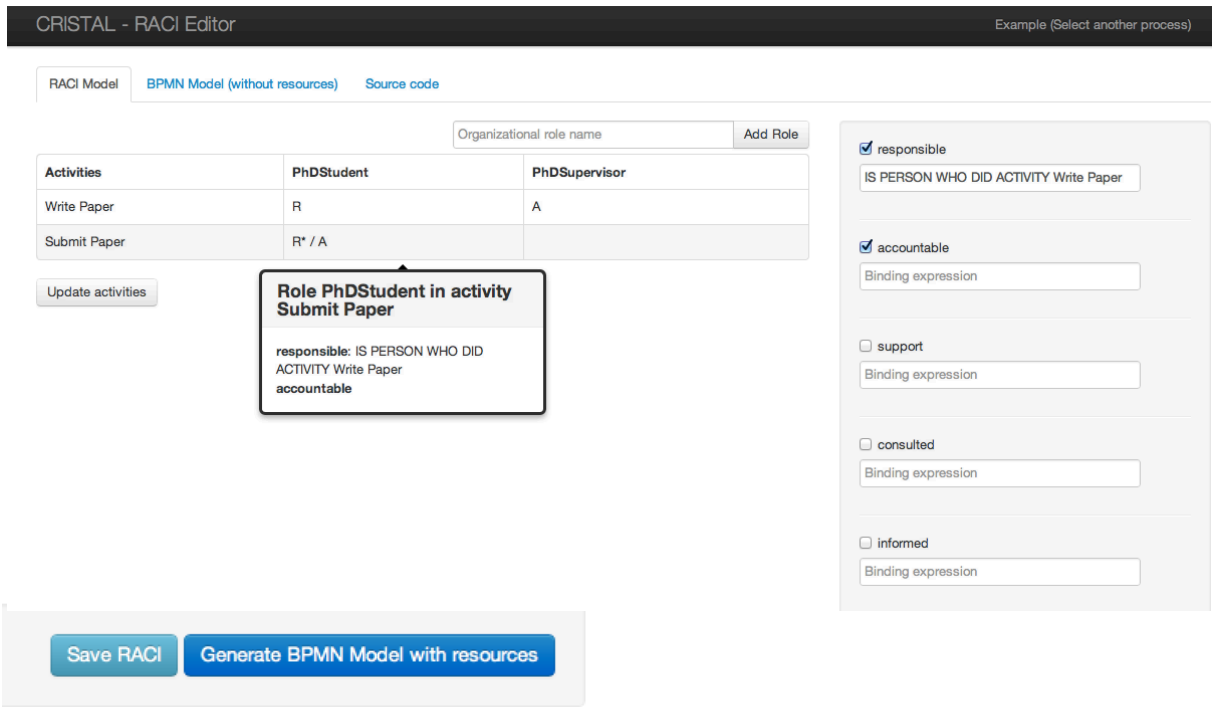


Figure 11.3: RACI Editor

the description of the RASCI matrix and optional binding information defined with RAL representing the resource perspective for the BP. The functionality of this component has been embedded in the *User Interface (UI)* of the RACI Editor (cf. Figure §11.3). Then, the RACI2BPMN component applies the transformations described in Section §7.4 [34] to automatically turn the input BPMN model into a RACI-aware BP models with all the resource information received as input. Furthermore, the BPMN generated is standard BPMN and, hence, it can be manipulated in any BP modelling tool, such as Signavio Oryx, and executed in a BPMN-compliant BPMS (e.g. Activiti) that has the required support for RAL.

11.2.3 Resource Analysis in Business Processes with CRISTAL

The implementation of the DL-based KB for RAL semantics has been done with OWL [88]. OWL is a knowledge representation scheme designed specifically for its use on the semantic Web, which exploits existing Web standards (XML and RDF), adding the familiar ontological primitives of object and frame based systems, and the formal

rigor of DLs (cf. Appendix §C for further information about OWL and DLs).

The support for the analysis of resources in BPs has been integrated in a component called *RAL Analyser*, depicted in Figure §11.4, which incorporates a module (the RAL KB Mapper) that prepares RAL's KB with the information necessary for the analysis of the BP resource perspective in isolation, together with control flow and data, and at design time and run time, necessary to perform the analysis operations defined in Section §5.3. In turn, some of the information added to the KB is generated by another component: the BP2OLC module, which executes the procedure described in Chapter §10. Then, two modules are in charge of providing the specific support for design-time and run-time analysis. Please, notice that the RAL Analyser module could also be integrated and used in other platforms. Let us see insights of the implementation of each module.

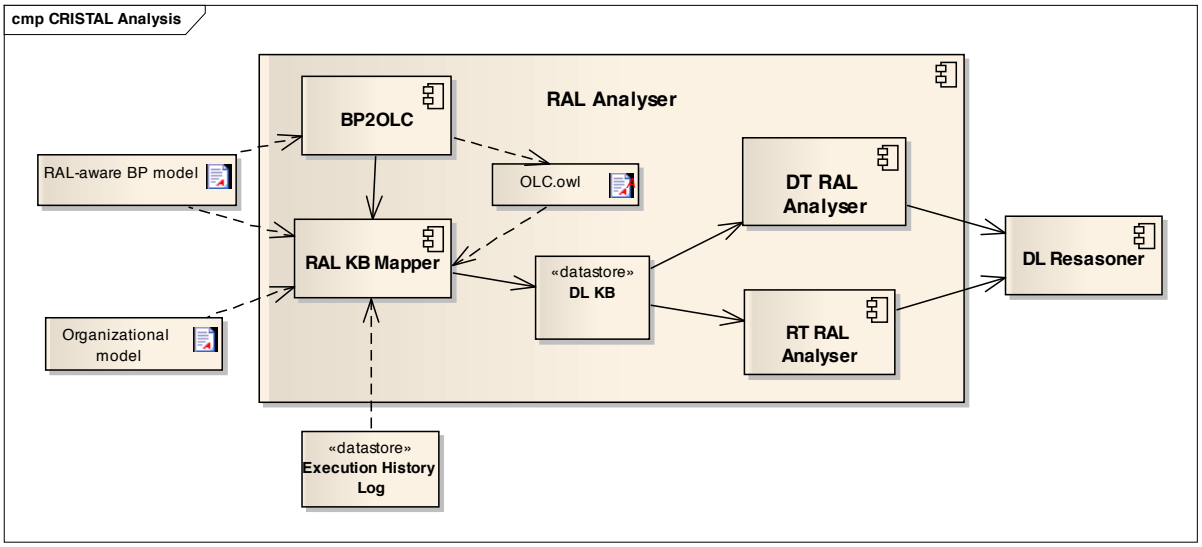


Figure 11.4: CRISTAL's analysis components

BP2OLC

The BP2OLC module takes as input a *data-aware BP model*, that is, a process model with information about the data objects read and written by the BP activities, as well as the states they can go through along the process. Then, it applies the BP2OLC procedure to generate the OLCs of the data objects of the process, which provide a data-centered view of the process the keep information about the control flow. The implementation of this tool is currently a prototype that reads the reachability graph of a Petri Net with information about the control flow and the data flow of a BP, obtained

from the ProM tool [5] in PNML format, and it executes the RG2OLC algorithm we have developed to process the reachability graph and generate the OLCs.

RAL KB Mapper

This component is in charge of extending RAL's KB with information specific to the application scenario, that is, the activities of a concrete BP model, the information of the OLCs associated to the data objects of the process, and the organisational model of the company where the process is operated. The extensions that can be applied to the KB were described throughout Chapters §9 and §10.

Thus, this tool receives a RAL and data-aware BPMN model and an organisational model, and it instantiates the DL-based KB generating two OWL files: one with information about the BP, and the other with the definition of the organisational structure. In case the BP is data-aware too, and operations related to data access control have to be performed, the RAL KB Mapper can also read the OLCs generated by the BP2OLC component for the process in question, and introduce the corresponding information into the DL-based KB, producing as output one more OWL file.

Design-Time and Run-Time RAL Analysers

As different mappings may be required for some RAL expressions (to distinguish between design-time and run-time analysis), two different modules are in charge of performing each one of them, generating the corresponding OWL files. Specifically, *DT RAL Analyser* is responsible for design-time analysis, and *RT RAL Analyser* provides run-time support. Furthermore, as depicted in Figure §11.4, they read information from the KB, that is, the OWL files generated by the mapper, in order to obtain all the information required to solve the RAL expressions.

Finally, a *DL reasoner* is used to compose and execute the specific analysis operations over the resource assignments (in our case *HermiT* [99]).

11.3 VALIDATION

The validation of the contributions with respect to the problems that we aimed at overcoming, have been done in several ways.

RAL's traceability comes from the definition of the language itself, which is grounded on an organisational meta model (cf. Section §6.2). Therefore, the concepts handled

n the RAL expressions are directly traceable with the concepts involved in the organisational model, as far as it uses the person, capability, position, role and organisational unit concepts, or a subset of these.

RAL's expressiveness was validated on the basis of the task duties and the assignment patterns covered from the set of them considered in this thesis, as it is the criteria that was used to evaluate the related approaches found in literature (cf. Section §5.4.1). As a result, RAL proved to be a very expressive resource assignment language.

The binding flexibility provided by CRISTAL has been validated by implementing the RACI Editor and the RACI2BPMN component. With them, we have shown that the proposals introduced in Chapter §7 are feasible and that, thus, it is possible to provide a separate binding method for the resource perspective of BPs while keeping the assignments expressive, as well as to automatically switch from a binding strategy to another while keeping consistency between the two models.

As far as analysis is concerned, the prototype of the BP2OLC procedure, the definition of the OWL files corresponding to RAL semantics, and the plugins and libraries developed for Signavio Oryx and Activiti, respectively, have been used for validating the research results, at the same time as for receiving feedback to detect improvement points and apply them to RAL semantics. Despite not all the analysis operations are integrated in CRISTAL, the whole RAL semantics has been validated, and the missing features will be included in the system sooner than later.

Besides validating the results as described in the previous paragraphs, part of the analysis capabilities developed in this thesis has been validated in a real scenario, by integrating it into a system developed in the scope of a transfer project with a company. The details of the project and the work performed are described in next section. Finally, the last validation mechanism we have used has been the publications of the results obtained from the research in relevant conferences and workshops in the scope of this thesis. All the publications related to the thesis along with other publications in the BPM field, are outlined in Section §12.2.

11.4 Application to the BPCMS Project

We have conducted a transfer project with a well-known multinational that applies some results of this thesis, as well as results of other research topics that we have worked on. The project was named *Business Compliance Management System (BPCMS)*,

and the goal was to develop a system that allowed the graphical definition of business rules, which could also be used by non-technical users, and the automatic verification of the compliance between the rules and the BPs used in the organisation, both at design time and at run time.

Specifically, the company needed to check 175 business rules that we classified in several groups according to the types of verifications required[4], e.g. generic rules, existence rules, control flow-related rules, data flow-related rules, and resource-related rules. Therefore, some rules dealt with the assignment of resources to the process activities. The rules were textually defined in XLS files.

The repository of BPs contained a number of processes modelled in Enterprise Architect, where the information related to resources was specified in tables associated to the BP activities in the tool. Only the Responsible task duty was considered.

When the project started, the organisation did not have any automatic mechanism for compliance checking and there were inconsistencies and incomplete information both in the definition of the rules and in the BP models. As part of the solution, the first thing we did was to redefine the resource assignments using RAL instead of the previous tables.

On the ground of research results on BP compliance [28, 37], the solution developed used the Web mashup technology as instrument for rule modelling and checking. A Web mashup is a data-driven WF (i.e. a data flow) generated from information from one or more data sources that is then aggregated, filtered, ordered, or somehow manipulated to generate a desired output [142]. We introduced and used the concept of compliance mashup as a DSL that allows to integrate heterogeneous data sources and to provide an operative specification of compliance rules over subsets of the information that can be extracted from them [37], and we developed a Mashup Editor in UML that was integrated as a UML Profile into Enterprise Architect.

Mashups are made up of a set of components connected to one another, which return an output that, in our case, is a boolean value with the result of the compliance analysis. In the compliance mashups, components for two types of mashups were defined: those that had to be executed at design time, and the ones targeted at run-time compliance checking. Figure §11.5 illustrates the use of the mashup editor with a rule that has to be checked at run time.

The way of integrating the analysis operations described in this thesis in this solu-

[4]We cannot show examples of the rules for privacy reasons.

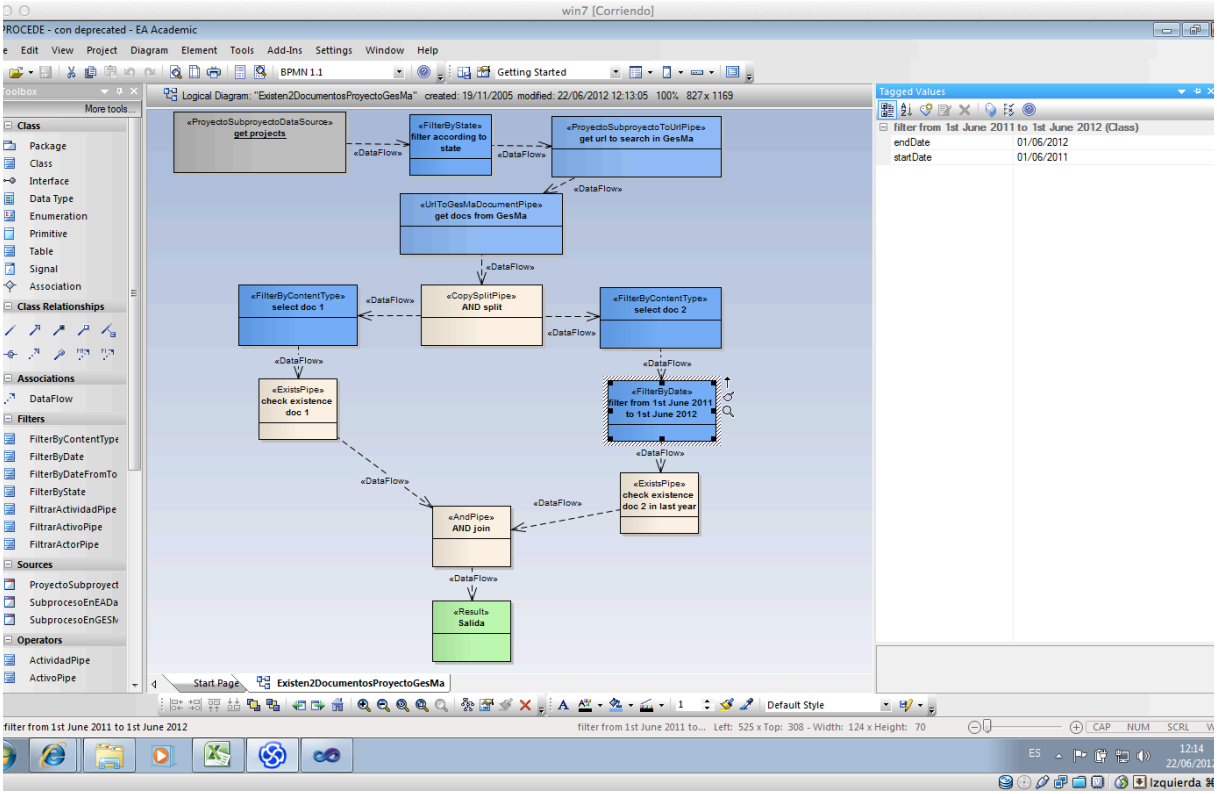


Figure 11.5: Compliance mashups for run-time compliance checking

tion, is by implementing components that encapsulate the functions corresponding to the analysis operations. Thus, some of the operators shown in Table §11.1 have been necessary in order to extract information about resources, specifically those related to the potential performers of an activity.

As result of the project with regard to what is of interest for this thesis, we concluded that it is possible to automate the analysis of resources defined with RAL on the basis of DLs, as well as to integrate the analysis approaches we propose in real scenarios. It is remarkable to say that, however, in order for the efficiency of the DL-based analysis to be appropriate, the resource assignments must not be complex, in terms of using RAL AC or RAL History with more than one access-control constraint.

11.5 SUMMARY

The work developed to achieve the goals set for this thesis, have been conceptually grouped in a system called CRISTAL, aimed at providing support for resource spec-

DS operation	Output	Function
Operations on business process models		
getProcess (PID)	BP	It returns the BP elements of the selected BP model
getActivities (PID)	[Activity]	It returns the set of activities of the selected BP model
Operations on organizational models		
getPerson (ID)	Person	They return the information associated to a specific element of an organizational model
getPosition (ID)	Position	
getRole(ID)	Role	
getUnit(ID)	Unit	
Operations using information from several data sources		
potentialOwners (A, OM)	[Person]	Potential performers of an activity according to its associated resource assignment expression. Sometimes looking through the organizational model may be required
potentialOwners ([Activity], OM)	[Activity: [Person]]	Potential performers of a set of activities. Sometimes looking through the organizational model may be required
positionView ([Activity: [Person]], OM)	[Activity: [Position]]	It classifies the persons by positions (according to the organization structure) and shows the positions associated with the activities
roleView ([Activity: [Person]], OM)	[Activity: [Role]]	This operation classifies the persons by roles (according to the organization structure) and shows the roles associated with the activities
unitView ([Activity: [Person]], OM)	[Activity: [Unit]]	It classifies the persons by organizational units (according to the organization structure) and shows the units associated with the activities

Table 11.1: DS operators for resource-aware compliance mashups

ification and analysis. In this chapter, we have provided an overview of CRISTAL, together with a description of each tool involved in them, and we have explained how we have validated all the approaches proposed in the thesis, specially some analysis results involved in the scope of an industrial project.

However, we would like to mention as well some limitations that CRISTAL currently presents, and which are target of future work.

- Only the BP resource perspective is currently considered in the implementations of the analysis operations as part of the RAL Analyser component. The code for the consideration of control flow issues is being tested outside CRISTAL, being its inclusion in the system after testing straightforward. The support for the process-

ing of data fields that activities may need in order to solve RAL Data expressions, on the contrary, was no implemented because it depends on the specific BPMS in which RAL is used, and we did not consider it was an urgent issue in a proof of concept of CRISTAL. It will be included in the future, though.

- On the other hand, the operations related to data access control are not yet covered by the system. Nevertheless, part of the BP2OLC has been prototypically implemented and is available under demand via email at It is about to be completed and integrated into CRISTAL.

Part IV

FINAL REMARKS

12

Conclusions and Future Work

"In three words I can sum up everything I've learned about life: it goes on"

Robert Frost (1874–1963),
Poet

"If you think in terms of a year, plant a seed; if in terms of ten years, plant trees; if in terms of 100 years, teach the people"

Confucius (551 b.C–479 b.C),
Philosopher

This chapter closes this thesis by providing an overview of all the work performed and the contributions presented. In Section §12.1 we outline the content of this manuscript together with the conclusions we have drawn and our highlights of relevant issues detected while carrying out this research. Most of the results presented in this thesis have been published in relevant forums in the scope of BPM research, and are summarized in Section §12.2. In Section §12.3 we describe how the results of our work can be used in other application domains. Finally, in Section §12.4 we mention the limitations we have identified in our approaches, we provide guidelines on how the problems can be overcome, and we bring forward some near and future work.

12.1 Conclusions

In this thesis our main goal was to enhance the management of the BP resource perspective, focused on human resources, at the Design and Analysis, and Enactment phases of the BP lifecycle. In particular, our interest was focused on the existing problems related to resource specification and analysis in BPs.

Regarding resource specification, we identified three issues to be taken into consideration. The first one is the type of organisational model used to assign resources to the BP activities. We noticed that there is a great variety of proposals introducing organisational structures, and that many of the approaches dealing with resource specification rely on concepts of a specific organisational metamodel to assign resources to activities, that is, they provide *traceable* resource specification solutions. However, standards such as BPMN [94] and BPEL4People [2] are not traceable to organisational models.

The next issue in resource specification relates to *expressiveness*. Specifically, we checked which of the five *task duties* we identified are supported by the current approaches, and what types of resource assignments they cover among twelve assignment patterns we identified from literature. Our perception from the results of the study is that a big part of the current approaches considers only the Responsible task duty of the activities and that, when other duties are considered, it tends not to be clearly specified to what extend they are supported. To this regard, we were surprised of the lack of literature dealing with RACI matrices in combination with BPs. As far as we knew, current BPMSs are increasingly caring about providing support for the modelling of RACI information in BP diagrams, e.g. Signavio Academic[1], ARIS[2]. Furthermore, we are aware that organisations are interested in this functionality, e.g. *Servicio Andaluz de Salud (SAS)*.

Another issue related to the expressiveness of the resource specification approach, is the amount of *assignment patterns* supported by the proposal. We concluded that most of the approaches deal with assignments based on the concepts of the organisational model they handle, or on access-control constraints (SoD, BoD), but only a few of them face the specification of other types of assignments. That is the reason why none of the approaches could be described as highly expressive according to our expressiveness criteria.

[1] http://www.signavio.com/joomla/en/academic.html
[2] http://www.softwareag.com/corporate/products/aris_platform/default.asp

The last issue we identified regarding resource specification was how to bind the resource specification to the control flow and data perspectives represented in a BP model. We defined three *binding strategies* and we concluded that the existing approaches propose solutions focused on only one of them (typically the all-in-one strategy), and thus, they do not provide *binding flexibility* to let the user select the most appropriate approach according to his/her needs.

With regard to the analysis of the BP resource perspective, the current support is limited to checking access-control constraints at design time, and/or calculating the potential performers of the task duties of the activities at run time. From literature we identified two *analysis operations* that had already been addressed, and up to nine more that either had been mentioned but not implemented, or could be derived from the previous operations. We classified the operations in two groups and noticed that not only the BP resource perspective needs to be taken into account to execute them. Indeed, control flow and data are involved in some resource-related operations. Our goal was to provide support for the automated resolution of all the eleven analysis operations at design time and at run time.

In this thesis, we have introduced contributions to face the problems identified from the study of approaches dealing with the aforementioned issues. The result is CRISTAL, which includes the following elements:

- RAL: a resource specification language traceable with a well-known organisational metamodel, very expressive (eleven out of twelve assignment patterns supported), and whose syntax is very close to natural language. Furthermore, it can be integrated into BPMN with no need to change the BPMN metamodel or semantics.

- Flexible binding of the BP resource perspective: an all-in-one approach based on the use of RAL with BPMN has been introduced. Furthermore, separate binding approach using RACI matrices and binding information has been presented. It outperforms the characteristics of other approaches pursuing a similar goal, which rely on the use of the BPMN swimlanes and on the introduction of activities specific for task duty modelling in the process models [29]. That has two problems: the BPMN swimlanes lack of semantics, so the models cannot be executed in a BPMN-standard engine taking into consideration the resource specification; and scalability, since activities representing task duties are modelled at the same level of the activities of the normal process, making readability worse.

- Capabilities for automated analysis: RAL semantics has been defined in DLs to provide RAL expressions with precise meaning, and to use the analysis capabilities that DLs offer. Consequently, we have been able to provide a reference implementation for the eleven analysis operations identified, providing support for design-time analysis and run-time analysis of the BP resource perspective.

Most of the contributions have been implemented and integrated into CRISTAL, and others have been tested in isolation and/or in real scenarios.

12.2 PUBLICATIONS

Most of the results presented in this thesis have already been published in scientific forums, and/or validated in real scenarios. Others constitute immediate future work. Furthermore, research results on other BPM areas that are not explicitly involved in this thesis, as well as collaborations with other researchers, have given rise to scientific publications, too.

Figure §12.1 outlines the publications achieved in the course of this thesis, whose *status* is indicated by the different kinds of marks explained in the legend of the figure, and which have been classified according to two aspects: type and topic. The *types* are represented by the background colour of the figure. Specifically, four types of publications are defined: journals, conferences, tool demos and workshops. The most relevant publications have a quality level associated, which corresponds to the JCR index for journals [100], and the position in the CORE and *Microsoft Academic Search (MAS)* rankings for conferences. Regarding the *topic* dimension, five lines are depicted in five different colours: (i) the green line represents the publications related to resource specification in BPs; (ii) in purple are shown the publications related to the analysis of resources in BPs; (iii) the yellow line involves the publications and presentations of CRISTAL, the tool support of our contributions; (iv) the publications related to BP compliance, which support the concept of compliance mashup that we have used to test some analysis capabilities in the scope of the BPCMS project, are depicted in red; finally, (v) the blue colour refers to collaborations and publications in other research lines, always in the area of BPM.

Publications related to Resource Specification

BPD 2011 [30] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes. In *Busi-*

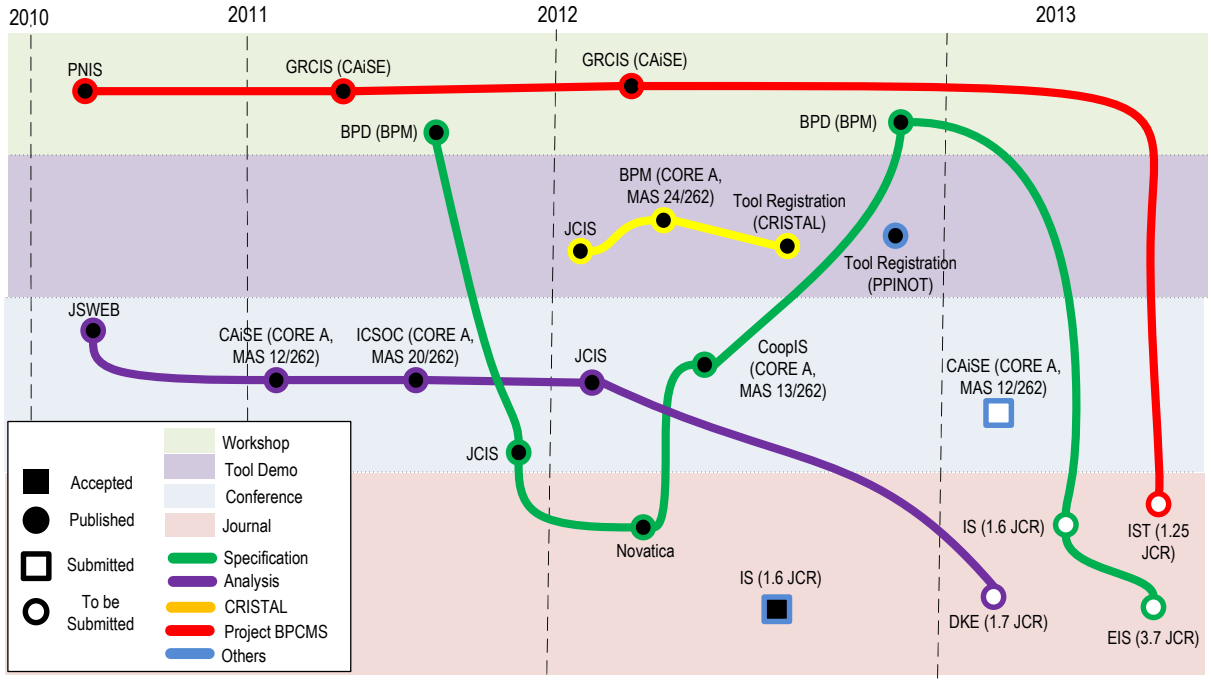


Figure 12.1: Publications overview

ness Process Management Workshops (BPD'11), pages 50-61, 2011. In this paper we introduce RAL specification, show example of its use and describe how it can be used with BPMN 2.0.

JCIS 2011 [29] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Mixing RASCI Matrices and BPMN Together for Responsibility Management. In *VII Jornadas en Ciencia e Ingeniería de Servicios (JCIS'11), volume 1, pages 167-180, 2011*. This paper introduces an approach for the modelling of RACI information in BPMN based on the BP swimlanes and on an extension of BPMN to model RACI-related activities.

Novatica 2012 [36] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Integrando las matrices RASCI en BPMN para la Gestión de la Responsabilidad. In *Novática: Revista de la Asociación deTécnicos de Informática, volume 216, pages 62-68, 2012*. The previous paper won the Best Paper Award in JCIS 2011 and was then enhanced and published in this national journal.

OTM 2012 [34] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés.Automated Resource Assignment in BPMN Models using RACI Matrices. In *OTM 2012 (CoopIS'12), volume 7565 (I), pages 56-73, 2012*. In this work we describe our separate modelling approach of the BP resource perspective using RACI matrices and binding

information, as well as the transformations to turn the resulting model into an all-in-one approach based on BPMN and RAL.

BPD 2012 [35] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Designing Business Processes with History-Aware Resource Assignments. In *BPM 2012 Workshops (BPD'12), volume 132, pages 101-112, 2012*. An extension of RAL specification and semantics to deal with history-based allocation is introduced in this paper.

IS 2013 An extension of the version of the already published RAL specification and analysis capabilities, is about to be submitted to the Information Systems journal.

EIS 2013 Our approach to provide flexible binding, after the extension of the explanation of the transformation from the separate to the all-in-one binding strategies, and the development of a new procedure to automate the switch also in the other direction, will be submitted to the EIS journal.

Publications related to Resource Analysis

JSWEB 2010 [26] C. Cabanillas, M. Resinas, and A. RuizCortés. On the identification of data-related compliance problems in business processes. In *VI Jornadas Científico-Técnicas en Servicios Web y SOA (JSWEB'10), pages 89-102, 2010*. In this paper we present a classification of data anomalies and data-related compliance problems that can appear in BP models. This was the starting point to develop the BP2OLC procedure.

CAiSE 2011 [31] C. Cabanillas, M. Resinas, A. RuizCortés, and A. Awad. Automatic Generation of a Data-Centered View of Business Processes. In CAiSE, volume 6741, pages 352-366, 2011. In this collaboration with Dr. Ahmed Awad we introduce the BP2OLC procedure (cf. Section §10.2), that is, the automated procedure to generate the OLCs of the data objects of a process model.

ICSOC 2011 [27] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Defining and Analysing Resource Assignments in Business Processes with RAL. In *ICSOC, volume 7084, pages 477-486, 2011*. In this paper, we present RAL formal semantics and examples of how to automatically execute analysis operations based on DLs, for which we take into consideration only the BP resource perspective.

JCIS 2012 [38] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Summary of "Defining and Analysing Resource Assignments in Business Processes with RAL". In *VIII Jornadas de Ciencia e Ingeniería de Servicios (JCIS'12), 2012*. It is a summary of

the ICSOC 2011 publication, presented in the session of already published relevant papers of the most significant Spanish conference on service engineering and BPM.

IST 2013 We aim at extending the paper published in CAiSE 2011 to apply the result of the BP2OLC procedure to the check the similarity between the BP data perspective of two BPs.

Publications related to CRISTAL

JCIS Demos 2012 [33] C. Cabanillas, A. del Río-Ortega, M. Resinas, and A. Ruiz Cortés. RAL Solver: a Tool to Facilitate Resource Management in Business Process Models. In *VIII Jornadas de Ciencia e Ingeniería de Servicios (JCIS'12), 2012*. This tool demo presents the functionality of CRISTAL related to the analysis of resources in BPs with RAL, that is, the RAL Solver component.

BPM Demos 2012 [32] C. Cabanillas, A. del Río-Ortega, M. Resinas, and A. Ruiz Cortés. CRISTAL: Collection of Resource-centrIc Supporting Tools And Languages. In *BPM 2012 Demos, volume 940, pages 51-56, 2012*. In this demo we show the specification and analysis features provided by CRISTAL that currently count on tool support.

CRISTAL Tool Registration 2012 C. Cabanillas, A. del-Río-Ortega, M. Resinas and A. Ruiz-Cortés. CRISTAL. entry refers to the official registration of CRISTAL in the Intellectual Property Record of Andalusia.

Publications related to the BPCMS Project

PNIS 2010 [25] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Hints on how to face business process compliance. In *III Taller de Procesos de Negocio e Ingeniería de Servicios (PNIS'10) in JISBD'10, volume 4, pages 26-32, 2010*. In this paper we present the results of a study we conducted on the literature related to BP compliance, which is aimed at serving as starting point for those who are starting their research on this topic.

GRCIS 2011 [28] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Exploring Features of a Full-Coverage Integrated Solution for Business Process Compliance. In *CAiSE 2011 Workshops (GRCIS'11), volume 83, pages 218-227, 2011*. This paper describes the main features that a compliance management system should provide in order to cover the entire BP lifecycle.

GRCIS 2012 [37] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Introducing a Mashup-Based Approach for Design-Time Compliance Checking in Business Processes. In *CAiSE 2012 Workshops (GRCIS'12), volume 112, pages 337-350, 2012*. This paper introduces the concept of compliance mashup that we defined as a means to specify and automatically check business rules.

DKE 2013 The publication of a detailed explanation of the insights of compliance mashups, their functionality, applicability, and the results of the BPCMS project, are an objective for my post-doctoral period.

Publications from Collaborations

PPINOT Tool Registration 2012 A. del-Río-Ortega, C. Cabanillas, M. Resinas and A. Ruiz-Cortés. PPINOT. The PPINOT tool suite has also been registered in the Intellectual Property Record of Andalusia.

IS 2012 [51] A. del-Río-Ortega, M. Resinas, C. Cabanillas and A. Ruiz-Cortés. On the Definition and Design-Time Analysis of Process Performance Indicators. *Information Systems*. Accepted. This article presents the PPINOT metamodel and a subset of the automated analysis operations based on DLs provided by PPINOT.

CAiSE 2013 A. del-Río-Ortega, C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Defining and Analysing Resource-Aware Process Performance Indicators. Submitted to CAiSE'13. In this collaboration we combine RAL with PPINOT in order to enable the specification and automated analysis of resource-aware *Process Performance Indicators (PPIs)*.

Table §12.1 shows the relation between the contributions presented in this thesis and the publications achieved during its execution. Regarding the contributions of the thesis (excluding the BPCMS project), there are three papers published in CORE A conferences, one tool demo published in the reference conference for our research (i.e. the BPM conference), two workshop papers in CORE A conferences, and several publications in national venues.

12.3 Application Scenarios

Some of the research results (total or partial) of this thesis can be applied to other domains to solve similar or different problems. Specifically:

Contribution Group	Contribution	Publication
Specification of resources in BP models	RAL	- BPM Workshops 2011 - BPM Workshops 2012
	Flexible Resource Specification with RAL	- JCIS 2011 - OTM 2012 (CoopIS) - Novatica 2012
Analysis of resources in BPs	RAL Semantics	- ICSOC 2011 - JCIS 2012 - CAiSE 2011
	Person-Activity Operations	
	Person-Data Operations	
Evaluation	CRISTAL	- JCIS Demos 2012 - BPM Demos 2012
	The BPCMS Project	- CAiSE Workshops 2011 - CAiSE Workshops 2012 - Transfer Project

Table 12.1: Outline of the contributions and the publications achieved

- RAL could be integrated into existing notations and systems (e.g. BPMN [94], YAWL [9]) in order to supplement the capabilities they currently provide for resource management by adding new types of resource assignments and capabilities for automated resource-related analysis.

- Contributions to the area of the so-called Social BPM [23] may be performed. Specifically, from a study we have conducted in collaboration with the research group headed by Professor Fabio Casati (University of Trento, Italy), we have concluded that resource specification is required in social networks, crowd sourcing platforms, and the like. Thus, RAL could be a good candidate to fill the existing gaps in such domains.

- In addition, RAL can help express some *Key Performance Indicators (KPIs)* for BP performance measurement (also called PPIs) [50], given rise to *resource-aware PPIs* that could serve for BP analysis and evaluation goals. Indeed, we are currently carrying out work in this direction, and interesting checks have already been identified, e.g. to define and measure a PPI that states that a person cannot invest more than 20 hours a month in performing activities of a specific process.

- The BP2OLC procedure developed as a pre-step for the extension of RAL's KB with information of the BP data perspective (cf. Section §10.2), can be applied to solve many problems in data-aware BPM [31]. For instance, data analysis can

be simplified by using the data-centered view of the BP generated by the procedure. Indeed, the procedure has already been used to detect data anomalies in BP models, as explained in [31]. It could also improve the support for data management of tools such as WebRatio [7], which currently does not consider the states of the data objects. Besides, the OLCs obtained could be used to check the degree of similarity between two BPs with regard to the way they handle information, which could serve for extending current results on the detection of BP similarity [54].

- The solution we used for the BPCMS project that we introduced in Section §11.4, which was based on the use of compliance mashups for rule specification and checking, provides a framework that allows the integration of existing BP compliance checking techniques [15, 65, 66, 80, 110]. Thus, rules involving different BP perspectives and relying on different analysis formalisms could be specified and checked with a single compliance mashup.

12.4 Limitations and Future Work

"Perfection does not exist" and our contributions currently present some limitations, to be named:

RAL's expressiveness. RAL might not cover *every* need for *every* organisation. Indeed, one of the assignment patterns used as evaluation framework in this thesis are not supported (Restricted Team Size).

Extension: RAL can be extended to add new expressions and/or constraints, as stated in Chapter §6. In fact, three extensions have already been developed from RAL Core (RAL Data, RAL AC, RAL History). Thus, RAL could be extended to deal with teamwork and collaborative environments, taking into consideration the Restricted Team Size pattern. Apart from that, we intend to provide support for Retain Familiar, the variety of the BoD pattern dealing with assignment preferences. Specifically, we plan to apply the results on service discovery and selection described by García et al. [64] with RAL.

Performance of DLs. The main problem of DLs is performance. We have realized that the resolution of assignments with several access-control constraints or commonality expressions, becomes slow.

Extension: A subset of DLs could be used for the implementation of certain analysis operations in order to increase their efficiency. In addition, other formalisms could be studied to implement certain operations.

Scalability. The BP2OLC procedure developed as an intermediate step for the extension of the KB with information related to the management of data objects in a process, relies on the reachability graph of the Petri Net associated to the BP to generate the OLCs of the data objects. However, with large processes, a state explosion can occur in the reachability graph.

Extension: In these cases, we could resort to the use of the so-called *coverability graph* instead of the reachability graph, and adapt the algorithm developed to process also coverability graphs.

Incomplete tool support. As stated in Section §11.5, support for considering the BP control flow perspective in resource analysis, and partial support to deal with the data perspective as well, are missing in the current implementation of CRISTAL.

Extension: as we have tested all the contributions by implementing them separately, we "just" need to adapt the ad-hoc code generated for each of them to provide the missing support.

The extensions outlined to overcome the aforementioned limitations is near-future work. Further future work is the application of the results from this thesis to the scenarios described in Section §12.3. Besides, we plan to consider the extension of the catalogue of analysis operations to include abductive operations in order to provide automatic answers to questions such as why it is not possible to assign a certain activity to a certain person. As an initial conjecture of this future work, we claim that in general, explanatory analysis should be regarded as a set of related operations that constitute a parallel hierarchy with respect to the catalogue of operations introduced in this thesis. This conjecture is taken from the Automated Analysis of Software Product Lines where it has been successfully applied by Trinidad and Ruiz-Cortés [125].

PART V

APPENDICES

A

Approaches Dealing with Resource Specification in Business Processes

This appendix presents an overview of the main proposals for resource specification found in literature, studied according to the foundations described in Section §3.2, that is, the underlying organisational model, the assignment patterns covered, the task duties considered for the process activities, and the assignment model proposed, if specified. We have focused on the field of WF and BP management, where a total of 21 approaches have been found.

A.1 Role-Based Access Control (RBAC)

As stated in [82], "RBAC is a framework for controlling user access to resources based on roles. It can significantly reduce the cost of access control policy administration and is increasingly widely used in large organisations".

The idea of using roles in access control emerged in 1984 [81], "in recognition of a need among government and industry purchasers of information technology products for a consistent and uniform definition of role-based access control features" [11], but it was two decades later that the ANSI standard for RBAC was approved, after several approaches and public reviews [60, 61, 77, 112]. RBAC was later revised to improve its capabilities to specify constraints in resource assignments (e.g. to describe SoD on activities), and to consider different kinds of role hierarchies. Further extensions have been proposed n recent years. For instance, Strembeck and Neumann [121] have introduced an approach to extend RBAC to deal with context constraints and conditional permission.

Today, the standard has three components [82]:

- Core RBAC: it defines core functionalities on permissions, users, sessions, and roles, according to the specification in [61]. A *role* can be thought of as a set of *transactions* that a user or set of users can perform within the context of an organisation. A transaction can be thought of as a transformation procedure [42] (a

program or portion of a program) plus a set of associated *data* items. Thus, this is not like conventional usage of the term in commercial systems. Such transactions include, for instance, the ability for a doctor to enter a diagnosis, prescribe medication, and add a entry to (not simply modify) a record of treatments performed on a patient. The role of a pharmacist includes the transactions to dispense but not prescribe prescription drugs. Transactions are allocated to roles by a system administrator. Membership in a role is also granted and revoked by a system administrator. [...] *Roles are group oriented*.

In addition, each role has an associated set of *individual members*. As a result, RBAC provides a means of naming and describing many-to-many relationships between individuals and rights. Figure §A.1[1] depicts the relationships between individual users, roles/groups, transformation procedures, and system objects. The formal notions of RBAC are the following:

- For each subject, the active role is the one that the subject is currently using.
- Each subject may be authorized to perform one or more roles.
- Each role may be authorized to perform one or more transactions. This means that the approach provides support for Role-Based Distribution.
- Subjects may execute transactions.

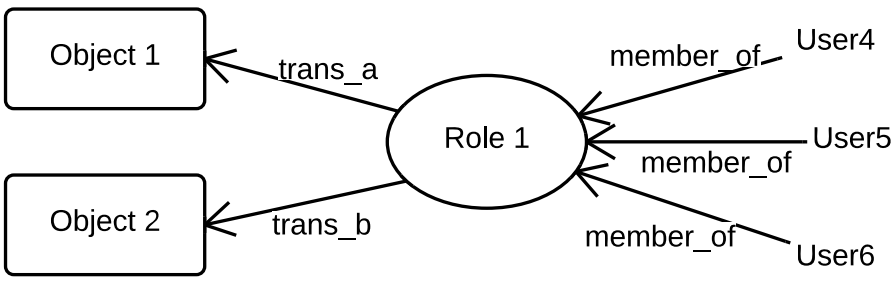


Figure A.1: RBAC role relationships

- Hierarchical RBAC. Hierarchical RBAC allows a role to inherit permissions from other roles without being granted those permissions directly. The ANSI standard for hierarchical RBAC has two sub-components: general hierarchy, which allows multiple inheritance, and limited hierarchy, which allows only single inheritance. For both of them, the standard requires that the inheritance relation be acyclic. However, Liu et al. argue that allowing unrestricted inheritance, where

[1]Picture taken from [61]

the inheritance relation is unrestricted and thus may contain cycles, could also be useful in some scenarios, since it provides extra flexibility. In that case, a cycle simply means that all the roles in the cycle are in an equivalence class and indeed have the same permissions.

- Constrained RBAC: Constrained RBAC supports SoD, whose purpose is to reduce fraud by limiting the power of individual users (SSoD) or individual sessions (DSoD), so fraud can be perpetrated only through collusion among multiple users or multiple sessions. This implies that the two modalities of the SoD pattern are implicitly included in the RBAC model.

Notice that RBAC is not exactly referred to tasks, but to objects. Specifically, it speaks about assigning permissions to users for access control to objects. Therefore, task duties are out of the scope of this approach. Anyway, it does no specify the type of duty authorized. Furthermore, not being related to process activities implies It neither deals with the binding strategy to be used for resource assignment in BPs. In spite of these two issues, we wanted to include the RBAC model in our study because it has been widely studied in different environments, and extended and applied to BPM for the specification of resource assignments in BP models.

A.2 XACML

The XACML profile [4] offers an implementation of the RBAC model with role hierarchies and inheritance, according to the ANSI-RBAC [61]. However, different names are used to refer to the RBAC concepts, e.g. *Subject* is used for RBAC's term User, *SubjectAttribute* refers to a Role, and $Role < PolicySet > / Permission < PolicySet >$ stands for Permission.

Furthermore, unlike Core RBAC, which requires extra support for multiple users per role, multiple roles per user, multiple permissions per role, and multiple roles per permission, all these requirements are supported by XACML. Specifically:

- XACML allows multiple Subjects to be associated with a given role attribute.
- XACML allows multiple role attributes or role attribute values to be associated with a given Subject.

- The *Permission* < *PolicySet* > associated with a given role may allow access to multiple resources using multiple actions.

- In addition to the basic Core RBAC requirements, XACML policies using this profile can also express arbitrary conditions on the application of particular permissions associated with a role. Such conditions might include limiting the permissions to a given time period during the day, or limiting the permissions to role holders who also possess some other attribute, whether it is a role attribute or not.

As far as the assignment patterns are concerned, the profile implicitly covers Role-Based Distribution. Moreover, the resource to be assigned to an activity can be referred by using other attributes different from its identity. XPath 2.0 expressions can be used for such a purpose. Therefore, we believe Capability-Based Distribution is also supported.

About authorization, the policies specified in XACML assume that all the roles for a given subject have already been enabled at the time an authorization decision is requested, and also that the presence in the XACML Request Context of a role attribute for a given user (Subject) is a valid assignment at the time the access decision is requested. In addition, as stated in its specification [4] "it does not deal with an environment in which roles must be enabled dynamically based on the resource or actions a subject is attempting to perform. For this reason, the policies specified in this profile also do not deal with static or dynamic SoD. A future profile may address the requirements of this type of environment".

XACML always refers to the performance of tasks. Therefore, it seems to be focused on the Responsible task duty. The profile does not specify the binding strategy to be used.

A.3 BERTINO ET AL.

Bertino et al. proposed a *Constraint Specification Language (CSL)* to assign users and roles to WF activities. They consider an organisation in which users are classified into roles, for which a domination relationship is established. So, given two roles R_i, R_j, "if R_i dominates R_j, then R_j should be given higher priority over R_i when assigning a role to the task" [20].

With that structure, their language allows for the specification of both simple and complex assignments, covering the following assignment patterns: (i) assignments to single users (Direct Distribution); (ii) assignments on the basis of roles (Role-Based Distribution); (iii) static and dynamic segregation of duties (SoD); (iv) assignments indicating who is and/or is not allowed to undertake a task; and (v) preferences by implementing an "unless" operator (Retain Familiar). Furthermore, compound assignments (e.g. with operator AND, and/or representing an "if...then..." block) can also be defined.

The specification of the language does not mention the support of task duties further than the resource that is Responsible for a task.

Regarding the binding strategy, the so-called *Constraint Base (CB)* containing all the information related to resource assignments and specified with the CSL introduced by the authors, is defined separately from the process model. Then, a role planner is in charge of calculating possible task assignments given the data in the CB, and filling in the process model with resource-related information before run time.

Potential future extensions of the work intend to deal with more complex organisational models inline with the RBAC specification.

A.4 Business Activities

The Business Activities constitute an integrated approach to enable the specification of process flows as well as process-related RBAC models and constraints [120]. They rely on the RBAC model to define the structure of the organisation, which thus consists of a hierarchy of roles that are assigned to the members of the company. Inheritance is assumed in the role hierarchy, so senior roles inherit the permissions from their junior roles.

The conceptual model of Business Activities is depicted in Figure §A.2[2]. They deal with the assignment of roles to process tasks, as well as the specification of static and dynamic SoD and BoD constraints. Therefore, patterns Role-Based Distribution, SoD, BoD, Case Handling and History-Based Distribution, are supported by the approach.

An implementation by extending the UML2 activity models has been performed. It includes a runtime engine to solve the resource assignments and constraints. It is then

[2]Picture taken from [120]

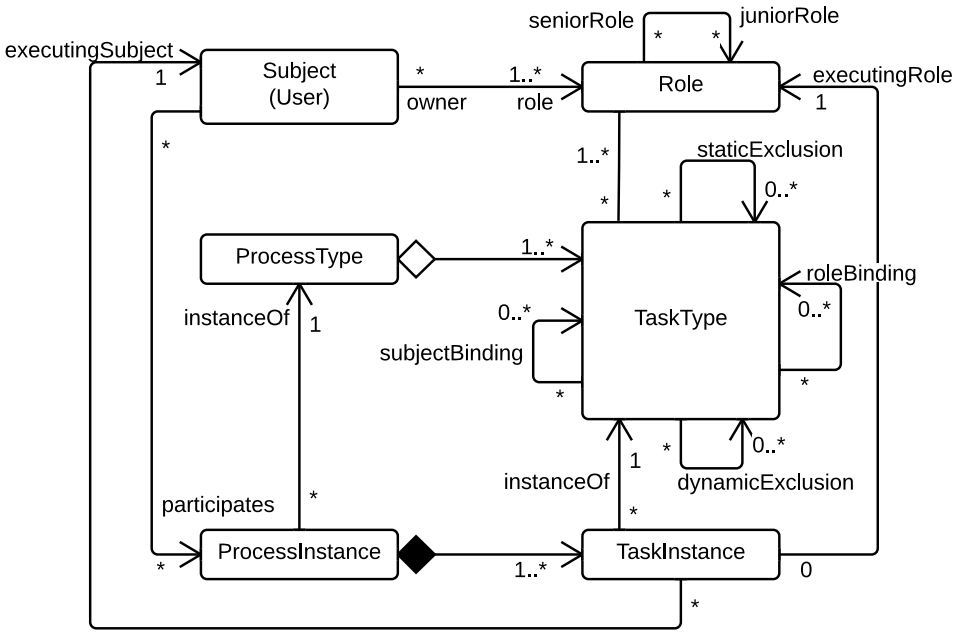


Figure A.2: Business Activity RBAC model

when the authorization for task execution are enforced. OCL is used to specify invariants for the UML2 extension in order to configure specific semantics for the Business Activities. Among them, the prohibition to assign two activities related to each other with a SSoD constraint to the same role (neither directly nor transitively), is included. This avoids problems due to role inheritance in the RBAC model.

Regarding the duties of the people with repeat to task execution, only the Responsible task duty is mentioned in this work.

The implementation provided uses a split binding strategy in which the assignment of tasks to roles is performed in an external model together with the whole configuration of the company, and only the access-control constraints are indicated in the activities of the BP model.

A.5 WIDE

The *Workflow on Intelligent Distributed database Environment (WIDE)* WF management system was developed within the scope of the WIDE Project[3], one of whose main goals was to define an advanced conceptual model for describing both the flow of WF

[3]Esprit Project 20280 - WIDE: http://cordis.europa.eu/esprit/src/20280.htm

activities and the organisational environment in which these activities are performed [40].

WIDE opts for separately modelling the specification of the organisational model and the process model, as depicted in Figure §A.3[4]. In the picture, the structure of both worlds and their relationships are shown. The organisation is defined in terms of different types of agents, which constitute a disjoint set. According to [40], the definition of each agent type is the following:

- *Actor.* An actor is an individual processing entity that may be of human or automated (mechanical or electronic) nature. The availability of an actor can also be specified (in terms of calendar, holidays, free time, and illnesses in case of human actors; in terms of down time, for instance for maintenance, in case of automated actors).
- *Group.* A group is a specification of a set of actors based on common organisational characteristics, e.g., an organisational unit.
- *Organisation function.* A function may be performed by a group (or a number of groups) or by individual actors. A skill attribute is associated to the function entity.

Hierarchy of agents can be established.

In WIDE, "it is assumed both that the structure of the organisation can be changed at any moment, and that new workflow schemas can be defined and the existing ones modified, and that the authorization to agents to perform a given role may vary over time, and can be dynamically changed during workflow execution. Such an issue is orthogonal both to process modeling and to organisational modeling", as shown in Figure §A.3.

Regarding resource assignment, "WIDE adopts the concept of *role* for specifying the ability for an agent to perform a given task. A role is a description of the processing entities that can and are allowed to perform a specific task. [...] They are defined *separately* from agents and from the organisational structure. In fact, the definition of roles is performed during process design, i.e., when specifying the characteristics of a given workflow schema". Therefore, roles constitute the connectors between the elements of the two models. Role-Based Distribution is thus supported. Each task can

[4]Picture taken from [40]

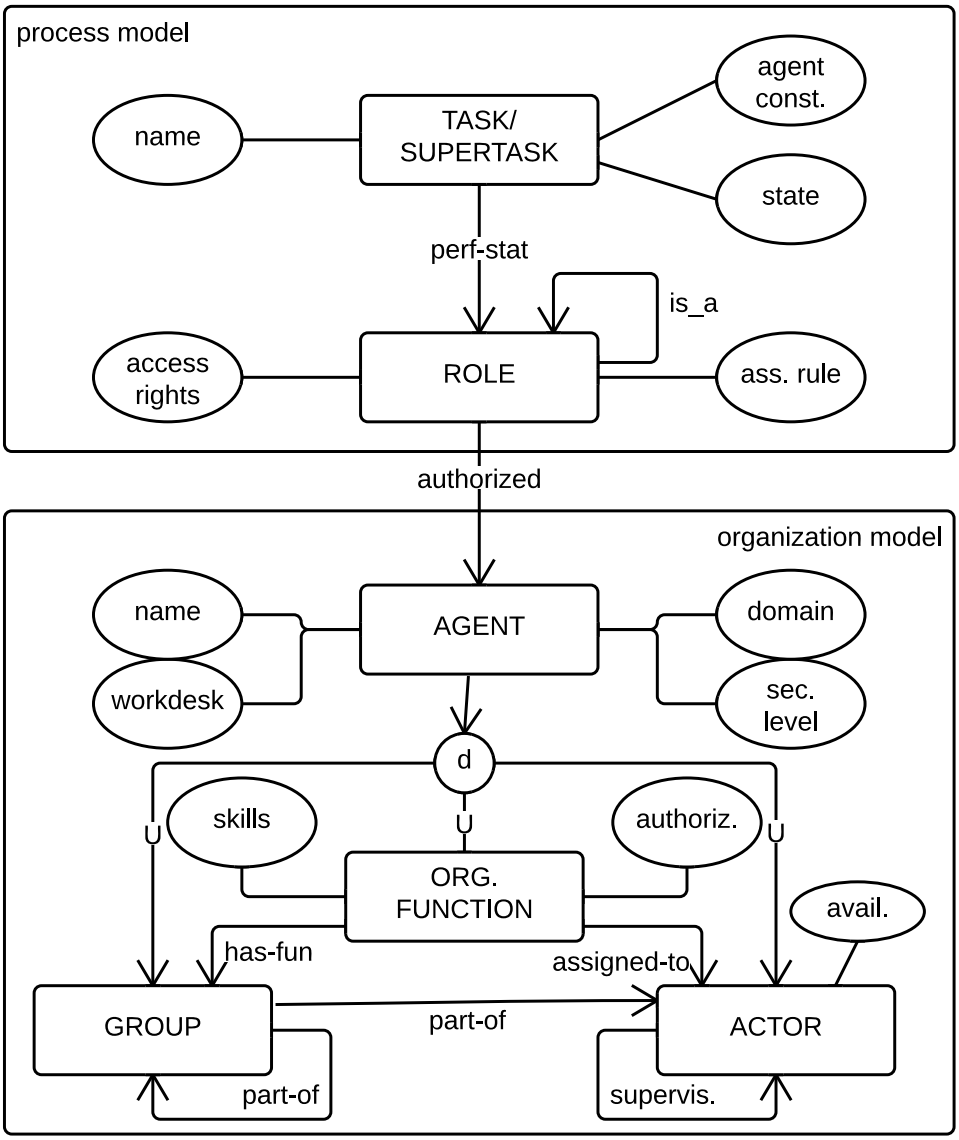


Figure A.3: WIDE organisational metamodel

be associated to one or more roles. The mapping between roles and specific agent entities is performed when the authorization information is defined. Furthermore, "the following types of constraints can be specified for resource assignment: (i) constraints based on variables; (ii) constraints on where a task is executed; (iii) constraints related to authorization in task-specific cases (it involves SoD and BoD, so the SoD and Case Handling patterns are covered); and (iv) constraints related to dynamics of task execution based on functions on the history of the case (i.e. History-Based Distribution). Besides, "a possible assignment operation is also the delegation of a task by an agent to another agent" [40]. The language used for the representation of WF models is *Workflow Definition Language (WFDL)*.

Although task duties Responsible and Accountable can be set with regard to a WF case (i.e. WF instance), regarding activities WIDE only deals with the resource Responsible for executing the task, which is called Task Executor. However, as stated in [40], "in some cases, the participation of different agents is needed, which is registered in the history log of the task". This happens for example in the so-called multi-tasks managed in WIDE: "a multitask is seen from two different points of view. When modelling, a multitask is seen as a unique task, with a set of input and/or output information, to be performed by some agent(s). When executing, a multitask is split into different instances that are executed independently of the rest of tasks belonging to the multitask" [40]. Therefore, task duty Support can be considered to be supported, and so is the Additional Participants assignment pattern.

The binding strategy for resource assignments is all-in-one: the assignment of roles to tasks and the configuration of the constraints with WFDL are included in the WF model.

A.6 EXTENDED WIDE

Casati et al. presented an advanced role-based authorization model for WF processes, extended with organisational levels and authorization constraints in [41]. The main elements of the organisational model are agents, roles and organisational levels. The last two describe capabilities of agents to execute tasks according to organisational assets. Roles and organisational levels are organized in hierarchies, to facilitate the assignment of tasks to agents.

Figure §A.4[5] depicts the advanced authorization model proposed by the authors [41]. It extends the traditional notion of models in which resource assignments are established once and are then used in all the process instances at run time, to deal with constraints such as SoD and restricted task execution (i.e. authorization for a specific period of time). To do so, three types of *authorization constraints* are introduced:

- Instance authorization constraints valid for specific process instances.
- Temporal authorization constraints valid for a period of time.
- History authorization constraints that depend on the state of the system at a certain point in time during WF execution, e.g. SoD.

[5] Picture taken from [41]

The authorization constraints are expressed as *Event-Condition-Action (ECA)* rules.

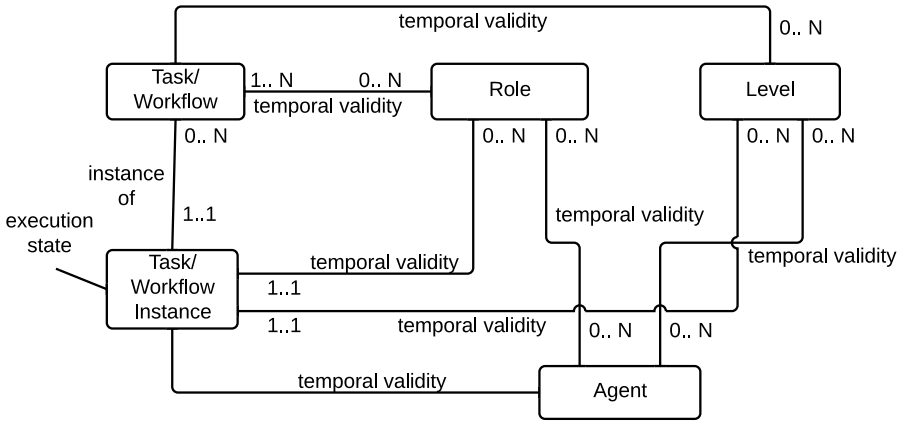


Figure A.4: Workflow authorization model by Casati et al.

As far as the assignment patterns are concerned, as depicted in Figure §A.4, roles and organisational levels are used to assign agents to WF activities. The Role-Based Distribution and Group-based Allocation patterns are thus supported. When specifying a certain instance, however, agents can be directly assigned (the Direct Distribution pattern).

The SoD and BoD access-control constraints can also be expressed by means of authorization constraints. Thus, patterns SoD, BoD and Case Handling are covered as well. History-Based Distribution is addressed by the history authorization constraints.

This work assumes we deal with the assignment of task performers in a WF (i.e. task duty Responsible). No other duties related to WF activities are mentioned in [41], but as WIDE deals with the Support task duty, support for this duty is implicit in the extension. The binding strategy is not explicitly explained. Intuitively, it could consist of a separate modelling approach. Nonetheless, we believe that split modelling can be applied too, by assigning roles and organisational levels in the WF model, and expressing the authorization constraints with ECA rules in separate files.

A.7 RUSSELL ET AL.

Russell et al. introduced an organisational metamodel that has been widely used together with the WRPs, also defined by these authors [105]. We are using the same definitions provided by the authors.

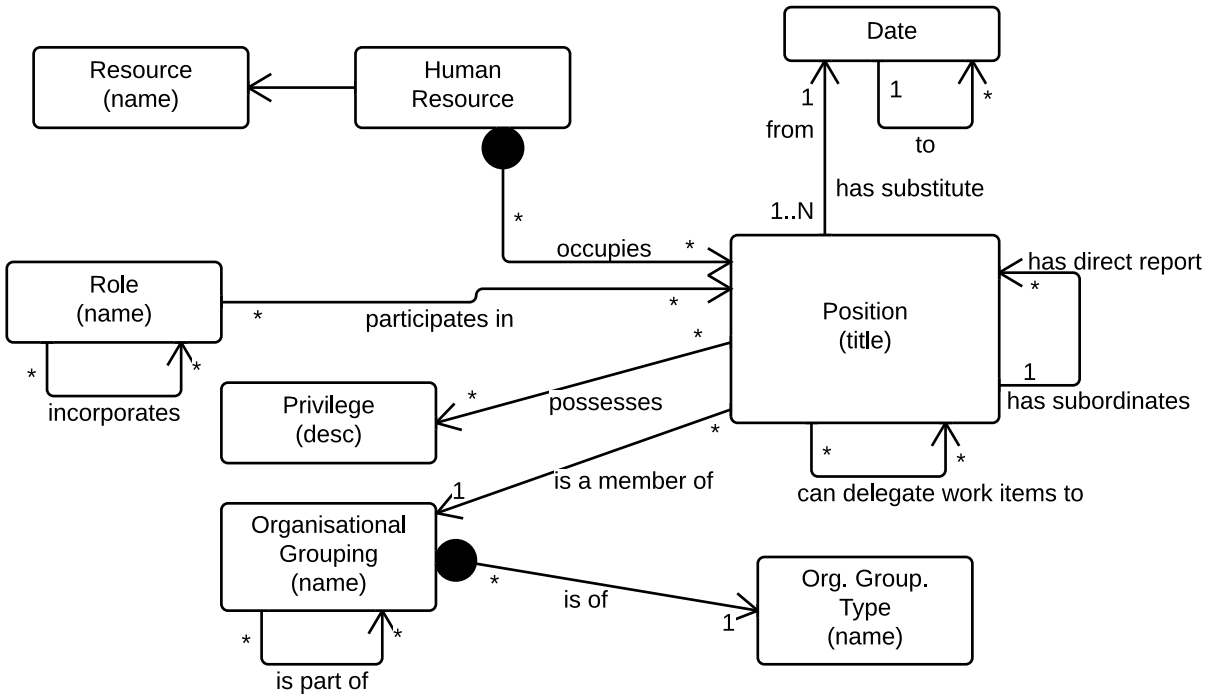


Figure A.5: Excerpt of the organisational metamodel described by Russell et al.

Figure §A.5[6] shows an excerpt of the metamodel, using an *Object Role Model (ORM)* [71] diagram. A *human resource* is typically a member of an organisation. An *organisation* is a formal grouping of resources that undertake work items pertaining to a common set of business objectives. They usually have a specific *position* within that organisation and in general, most organisational characteristics that resources possess relate to the position(s) that they occupy rather than directly to the resource themselves. As a consequence of their position(s), resources may have a number of associated *privileges*. They may also be a member of one or more *organisational units* which are permanent groups of human resources within the organisation that undertake work items relating to a common set of business objectives. Similarly they may also be a member of one or more *organisational teams*. These are similar to organisational units but not necessarily permanent in nature. Even less formal in nature is the notion of organisational *groups* which are often used to define groupings of resources with some common characteristic or cause e.g. social club members, fire-wardens etc.

Each resource is generally associated with a specific *branch* which defines a grouping of resources within the organisation at a specific physical location. Resources may

[6]Picture taken from [105]

also have a *level* which indicates their position within the organisational hierarchy. They may also belong to a *division* which defines a large scale grouping of resources within an organisation either along regional geographic or business purpose lines.

In terms of the organisational hierarchy, each resource may have a number of specific relationships with other resources. Their *direct report* is the resource to whom they are responsible for their work. Generally this is a more senior resource at a higher organisational level. Similarly, a resource may also have a number of *subordinates* for whom they are responsible and to which each of them report. Finally, a resource may also have a *delegate* which is an alternate human resource to which they assign work items previously allocated to them. This reassignment of work items may occur on a temporary or permanent basis.

A resource may have one or more associated roles. *Roles* serve as another grouping mechanism for human resources with similar job roles or responsibility levels e.g. managers, union delegates etc. Individual resources may also possess *capabilities* or attributes that further clarify their suitability for various kinds of work items. These may include qualifications and skills as well as other job-related or personal attributes such as specific responsibilities held or previous work experience.

This organisational metamodel served as basis to define many of the WRPs [17, 106], as well as to develop the *Yet Another Workflow Language (YAWL)* language and tool summarized in next section.

Therefore, even though the work in [105, 106] does not introduce a specific approach for the specification of resource assignments in BPs, it is sufficiently important to be mentioned in this section. Besides, the metamodel in Figure §A.5 has been used in some of the contributions of this doctoral thesis.

A.8 YAWL

YAWL is an open-source WF modelling language developed on the basis of Petri nets and the WRPs[7]. In particular, the language extends Petri nets with dedicated constructs to deal with the patterns [9, 132]. The YAWL system has been built with the foundations of the language. YAWL copes with not only resource management in BPs, but offers comprehensive support for the control flow and the data perspectives as well. However, "the Resource Service is the largest and the most complex of the

[7]Workflow Patterns: www.workflowpatterns.com

YAWL Custom Services", as stated in [8].

Figure §A.6[8] illustrates the underlying organisational metamodel, as described in [8]. In YAWL, a human resource is referred to as a Participant. Each participant may perform zero or more Roles, hold zero or more Positions (each of which may belong to an Organisational Group), and possess zero or more Capabilities. To maintain flexibility in the model, a participant's relationship to the other entities is not enforced; thus, a simple set of participants is sufficient to allow resourcing of work items in YAWL.

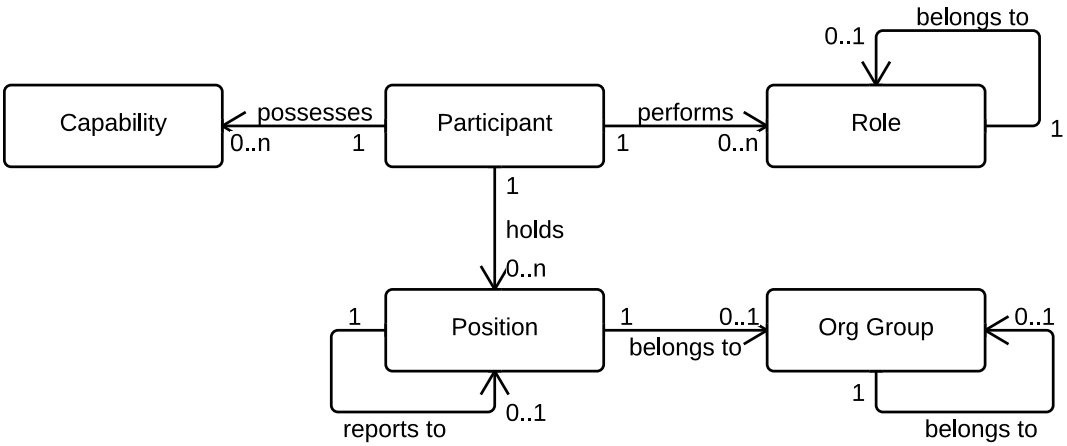


Figure A.6: YAWL organisational model from the perspective of a unique participant

Regarding resource assignment, the other types of entities of the organisational model are used to perform restrictions over the set of people previously selected. For a manual task, a designer may provide details of a *distribution set* of resources to which the task should be offered at runtime. A distribution set may consist of the union of zero or more individual participants, zero or more roles, and zero or more dynamic variables (which at runtime will be supplied with details of participants and/or roles to which the task should be offered, thereby supporting the late binding of resources to tasks, i.e. Deferred Allocation). Therefore, participants and roles can be directly assigned to process tasks (i.e. Direct Distribution and Role-Based Distribution, respectively). The resultant distribution set may be further filtered by specifying that only those participants with certain capabilities (Capability-Based Allocation), occupying certain positions (Position-based Allocation) and/or being members of certain organisational groups (Group-based Allocation), be included.

A designer may also specify certain constraints to apply, for example, that a certain work item must not be performed by the same participant who completed an earlier

[8]Picture taken from [8]

specified work item in a process (i.e. the SoD pattern), or that if a participant who is a member of the distribution set of a work item is the same participant who completed a particular previous work item in the process, then they must also be allocated the new work item (i.e. the Retain Familiar patterns).

As stated in [8], "the YAWL Resource Service provides full support for 37 of the 43 resource patterns identified by Russell et al. [106] (the remaining six being particular to the case-handling paradigm)".

As for the task duties, only the recipient of potential performers for process tasks is mentioned, which corresponds to *our* task duty Responsible. The binding strategy seems to be all-in-one, although task configuration is made by means of emerging windows in the YAWL system.

A.9 BPMN 1.0 Extension by Grosskopf

With the aim of providing a resource layer for BPMN, in 2007 Großkopf proposed an extension for the unique version of the standard released so far, BPMN 1.0 [69]. In BPMN 1.0, and still at present, the swimlane concept was rather open and unspecific. For user tasks the attribute Performers could optionally be set to identify the person(s), group(s) or organisational units which will perform the task at run time. However, this attribute did not yet properly capture the information needed. It seemed BPMS was going to adopt the *Business Process Definition Metamodel (BPDM)* standard for process modelling in subsequent years. BPDM aims to provide a notation independent representation of process models, and thus defines a metamodel to capture information relevant for BPs, which also includes resource information. BPDM has Performer Roles specified to be responsible for an activity (i.e. task duty Responsible is implicitly supported). Performer Roles can delegate responsibility to other Performer Roles. A performer role is mapped to an actor at run time who actually performs the task.

Using the WRPs [106] as resource management evaluation framework, Großkopf assessed the support provided by BPMN and BPDM, and extended the BPDM Activity Model with attributes and associations to capture the yet missing patterns. The resulting metamodel is shown in Figure §A.7[9]. The attributes *allocationConstraint*, *mandatory* and *uniquePerformance* were added to the Activity to capture constraints at design time for the allocation and execution of tasks at run time.

[9] Picture taken from [69]

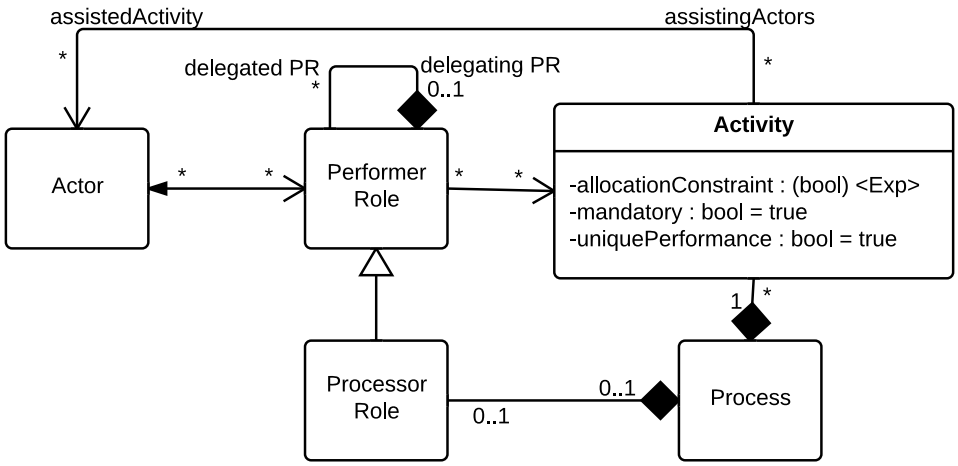


Figure A.7: Enriched BPDM Activity Model by Grosskopf

The extension provides support for the following assignment patterns: Direct Distribution, Role-Based Distribution, Deferred Allocation, SoD, BoD, Case Handling, and Retain Familiar.

Capability-Based Allocation, History-Based Allocation, Position-based Allocation and Group-based Allocation, are out of scope for the work as they require the ability of the WF system to provide environment information, and they "do not assume that data outside the scope of the current process instance is directly accessible".

Note that both the organisational metamodel and the expression language to specify the resource assignments are left up to the modeller.

The person in charge of an activity can request for help form additional participant. Therefore, the Support task duty is covered in the approach, which uses an all-in-one binding strategy, since all the resource-related information must be contained in the process model.

A.10 BPMN 1.0 Extension by Wolter and Schaad

Starting from a basic role-based organisational metamodel (i.e. composed of users and roles) and the BPMN 1.0 specification, Wolter and Schaad extended the process modelling standard to enable a description of role hierarchies, authorization constraints (separation of duty and binding of duty), and cardinality constraints, without affecting the control flow semantics of the notation [139]. In particular, they provided it with support for resource allocation patterns such as SoD, Role-Based Allocation,

BoD, Case Handling, and History-Based Allocation, by introducing and/or modifying some entities in the BPMN metamodel, shown in gray in Figure §A.8[10]:

- A boolean attribute *Manual Task* is added to the process activities to specify that a task is manual, i.e. it must be performed by an user.
- *Nested lanes* are used to represent the role-based task authorization inheritance and role hierarchy.
- Multiple instance tasks and looped tasks are considered as a *Group* with exactly one task, but an arbitrary number of task instances.
- *Text Annotations* are used to specify the authorization constraints, e.g. "task $t1$ and $t2$ must be performed by the same user"; or "two users must be involved in the execution of $t1, t2, t3, t4$ and each user may only perform two out of the four tasks".

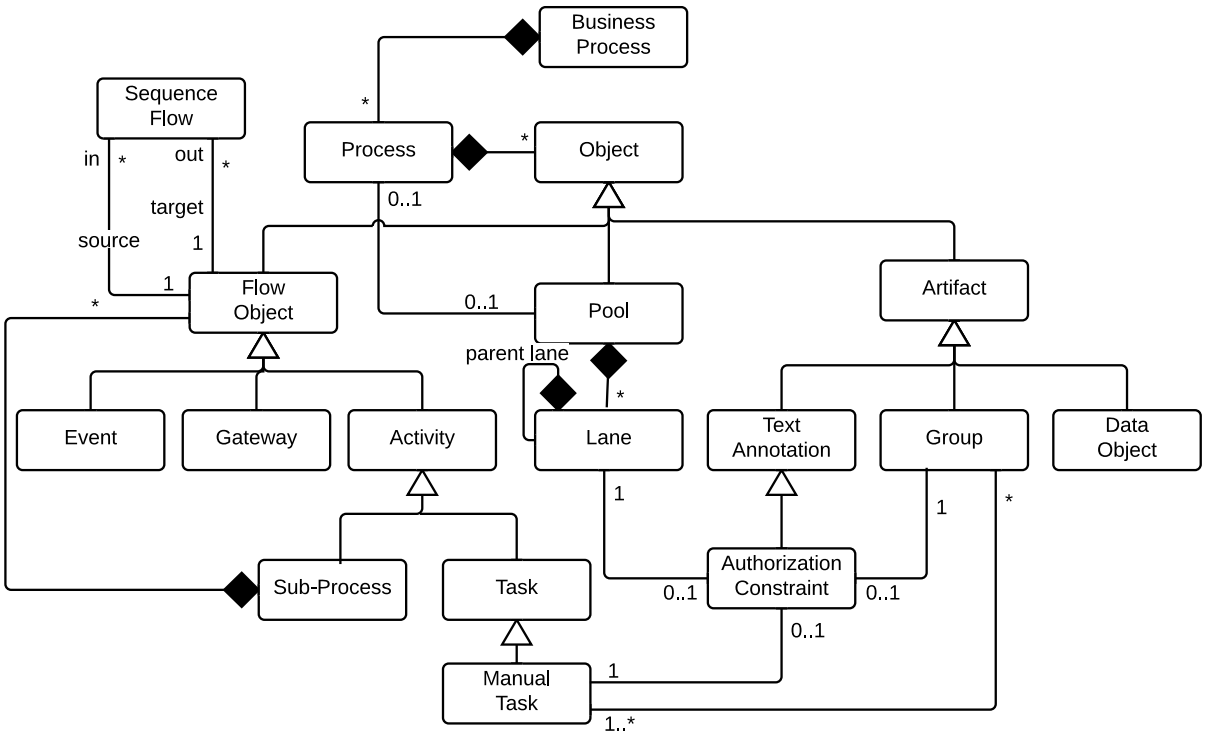


Figure A.8: Extended BPMN 1.0 Metamodel by Wolter and Schaad

[10]Picture taken from [139]

As the authors state, "the approach is limited to constraints defined on a pair of tasks and has to be expanded to an arbitrary set of tasks per constraint" [139].

The only task duty mentioned in this all-in-one approach (cf. Section §3.2.2) is the corresponding to the resources Responsible for the activities.

A.11 BPMN 1.1 Extension by Meyer

Meyer dealt with the BP resource perspective in his Master Thesis [86]. The goal addressed was three-fold. First, he introduced the ARPs, some of which were considered in Section §3.2.3 to define the assignment patterns used in this Doctoral Thesis. Second, he performed a review and re-categorization of the WRPs [106] and the ARPs according to criteria such as functional dependencies and use. As a result of the assessment, he developed an extension for BPMN 1.1 to include of the elements required to cope with complete resource management.

Figure §A.9[11] shows the extended metamodel proposed in [86]. Looking at the pure organisational structure (disregarding entities for allocation purposes), the main entity in the model is the Resource. The resource can be partitioned into two disjunct sets: Human Resources and Non-Human Resources. "Resources can also be part of hierarchically structured Teams for solving collaborational tasks and do may get assigned to also hierarchically structured Profiles, which embrace experiences, skills, and capabilities. A tertiary association exists between the entities Resource, Role, and Permission. Resources and Roles are assigned to each other and Permissions are connected to both entities to allow special actions like power of attorney, i.e. the permission to act on behalf of the organisation. Roles are structured hierarchically for supporting superior-subordinate-relationships. Furthermore, they are disjunctly partitioned into Functional Roles and Organisational Roles: Functional Roles cover all business related aspects, whereas Organisational Roles represent an organisation's structure.

The third contribution of the master thesis is a prototypical implementation of the resource-related features introduced, into Oryx. The prototype supports the Direct Distribution, Role-Based Distribution, Group-based Allocation, SoD, BoD, Case Handling patterns. SoD and BoD are only dynamic. The inclusion of features to deal with SSoD and SBoD is part of future work.

Regarding the task duties, only the performers of the tasks are involved in the im-

[11] Picture taken from [86]

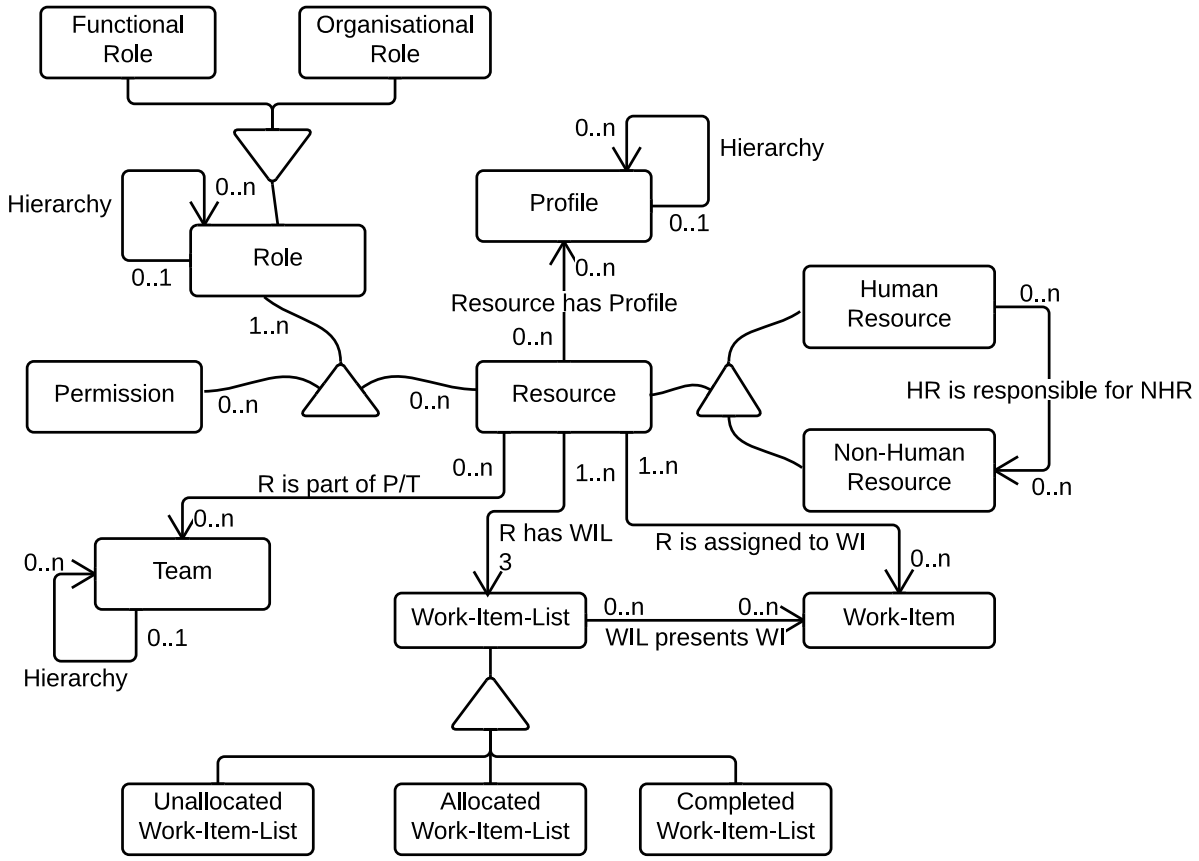


Figure A.9: Organisational Metamodel by Meyer

plementation provided. Thus, the only task duty certainly supported corresponds to the resource that is Responsible for the activity.

According to the prototype presented, the approach uses an all-in-one binding strategy.

A.12 BPMN 1.1 Extension by Awad et al.

Preliminary results of the final approach presented by Meyer in [86] (see previous section), were published by Awad et al. in [17]. The organisational metamodel is an excerpt of Meyer's metamodel, as depicted in Figure §A.10[12]. Unlike in Meyer's master thesis, in that work only certain WRPs are considered for resource management evaluation, which uses OCL constraints to specify resource assignments for BP activities.

[12]Picture taken from [17]

All the creation patterns and some detour patters are supported. Specifically, patterns Direct Allocation, Role-based Allocation, SoD, BoD Case Handling, Retain Familiar, Capability-based Allocation, History-based Allocation, and Group-based Allocation described in Section §3.2.3 are addressed.

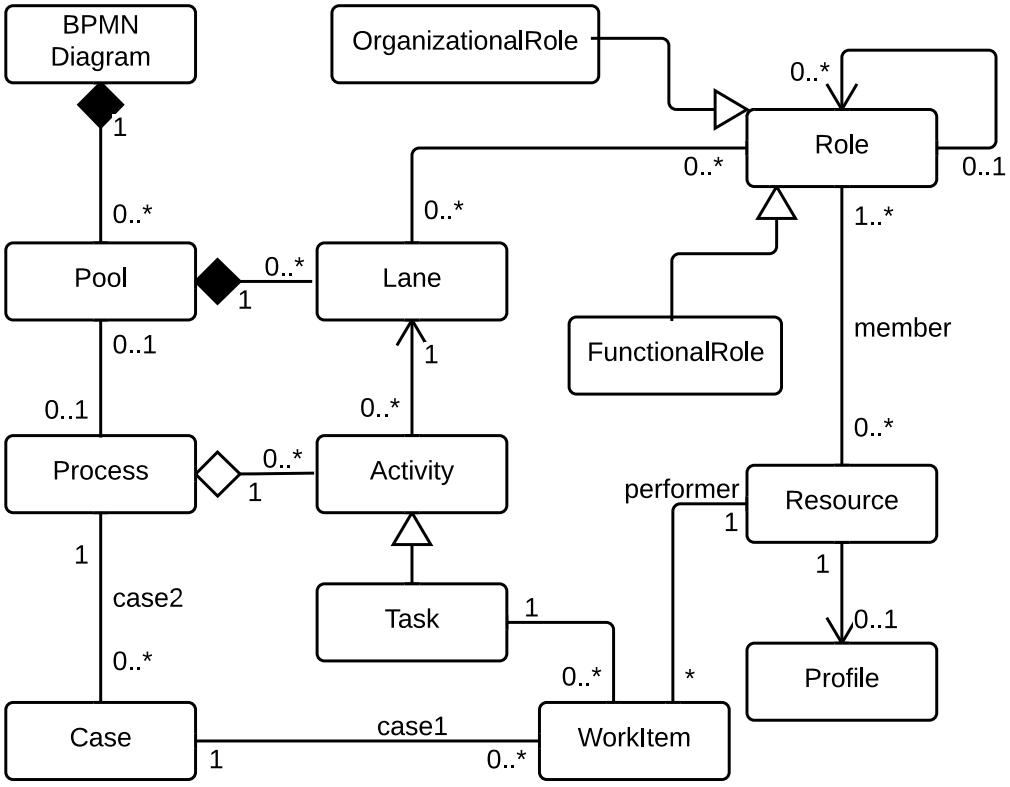


Figure A.10: Organisational Metamodel by Awad et al.

A plug-in for Oryx with this functionality has been developed. In the tool, the OCL constraints are inserted by means of a wizard-driven UI. They are then added to the Performers property of the task. This has a two-fold implication. On the one hand, only the Responsible task duty is considered. On the other hand, BP models and resource assignments are modelled separately, but they are then used together when running the process.

A.13 BPMN 2.0

Despite interesting improvements have been added to the current version of BPMN (2.0), the resource perspective still remains a bit less attended than other BPM aspects [94]. The need of bridging the gap between BP models and organisational models is a

fact, and it has been described in many papers of this research field [20, 86]. However, BPMN is not provided with an underlying organisational metamodel that worked as source to assign resources to the process activities.

By default, XPath expressions are associated to the activities to specify the potential owners of the activities (i.e. the Responsible and Accountable task duties are considered and assumed to be associated to the same resource). Therefore, the support with regard to the assignment patterns described in Section §3.2.3 is mainly the one provided by the XPath language. Specifically, Direct Distribution, Role-Based Distribution, Capability-Based Allocation, History-Based Distribution, Position-based Allocation and Group-based Allocation, are covered. However, access-control constraints such as SoD and BoD cannot be specified with XPath.

It is inherent in BPMN that all the features provided be included in the process model, so it follows an all-in-one binding strategy.

A.14 BPEL4People/WS-HumanTask

BPEL4People is an extension of the BPEL notation to address human interactions in BPEL as a first-class citizen. It defines a new type of basic activity which uses human tasks as an implementation, and allows specifying tasks local to a process or use tasks defined outside of the process definition [2]. This extension is based on the WS-HumanTask specification, which enables the integration of human beings in service-oriented applications [3].

WS-HumanTask (and thus BPEL4People) does not use any pre-defined organisational structure, neither is it provided with any analysis mechanism to deal with resource assignments once they have been specified and associated to the process activities, as stated in its own specification: "Out of scope of this specification is how human tasks and notifications are deployed or monitored. Usually people assignment is accomplished by performing queries on a people directory which has a certain organisational model. The mechanism determining how an implementation evaluates people assignments, as well as the structure of the data in the people directory is out of scope" [3]. Therefore, WS-HumanTask just considers that there are a set of people belonging to the organisation who can be used for resource assignment purposes.

As far as the assignment patterns are concerned, WS-HumanTask offers different ways to assign people to process activities, thus covering several patterns, in particular:

- *Logical people group*. A logical people group represents one or more people, one or more unresolved groups of people (i.e., group names), or a combination of both, classified according to their features, functions or permissions with regard to certain activity or process, or any other criteria. Group names can be thus used to indicate the potential owners of an activity. A logical people group is bound to a people query against a people directory at deployment time. At runtime, this people query is evaluated to retrieve the actual people assigned to the task or notification.

- *Literals*. People and group names can be directly used to specify the potential owners of a process activities. The Direct Distribution pattern is thus covered with this assignment method.

- *Expressions* defined in any language can be associated to the process tasks to limit the set of potential owners for them. The default language in WS-HumanTask is XPath[13], so the WRPs addressed to this respect are those covered by this notation, to be named: Role-based Distribution, Capability-based Distribution, and Organisational Distribution.

- A *Routing Pattern* is a special form of potential owner assignment in which a task is assigned to people in a well-defined order, either in sequence or in parallel. Each assignment must result in a separate sub-task.

In WS-HumanTask, some decisions for resource allocation can be deferred until run time, thus covering the Deferred Distribution pattern. Furthermore, Automatic Execution is also covered by the notation, since it serves for automatic BP execution.

These patterns can be used to specify the potential owners of process tasks, which are mandatory in a BP model, and will give rise to a final actual owner of each task (i.e. task duties Responsible and Accountable assumed to be associated to the same resource). Nonetheless, the so-called *Generic Human Roles* defined in WS-HumanTask explicitly include some of the task duties we mentioned in Section §3.2.2, too. Specifically the *task stakeholders* represent the people Accountable for process activities, and the *notification recipients* involve those people that must be Informed of milestones related to task execution.

All the information related to resource assignments is included in the BP models, thus following the all-in-one binding strategy.

[13] http://www.w3.org/TR/xpath/

A.15 ARIS

ARIS contains a set of applications dealing with the management of different BP perspectives, and it introduces several process modelling methods associated to the different process views [113].

As far as the organisational view is concerned, ARIS provides an organisational metamodel with the following entities connected as depicted in Figure §A.11[14], mainly: a hierarchy of organisational units based on the unit types, positions, roles, and qualifications. Persons are considered a special type of organisational unit composed of only one member. The term role appears as a way to classify individuals in user classes (a.k.a. profiles) according to their qualifications and skills. Groups of people can be settled for referring to people sharing certain characteristics, too [119]. This is done by means of the Person Type class.

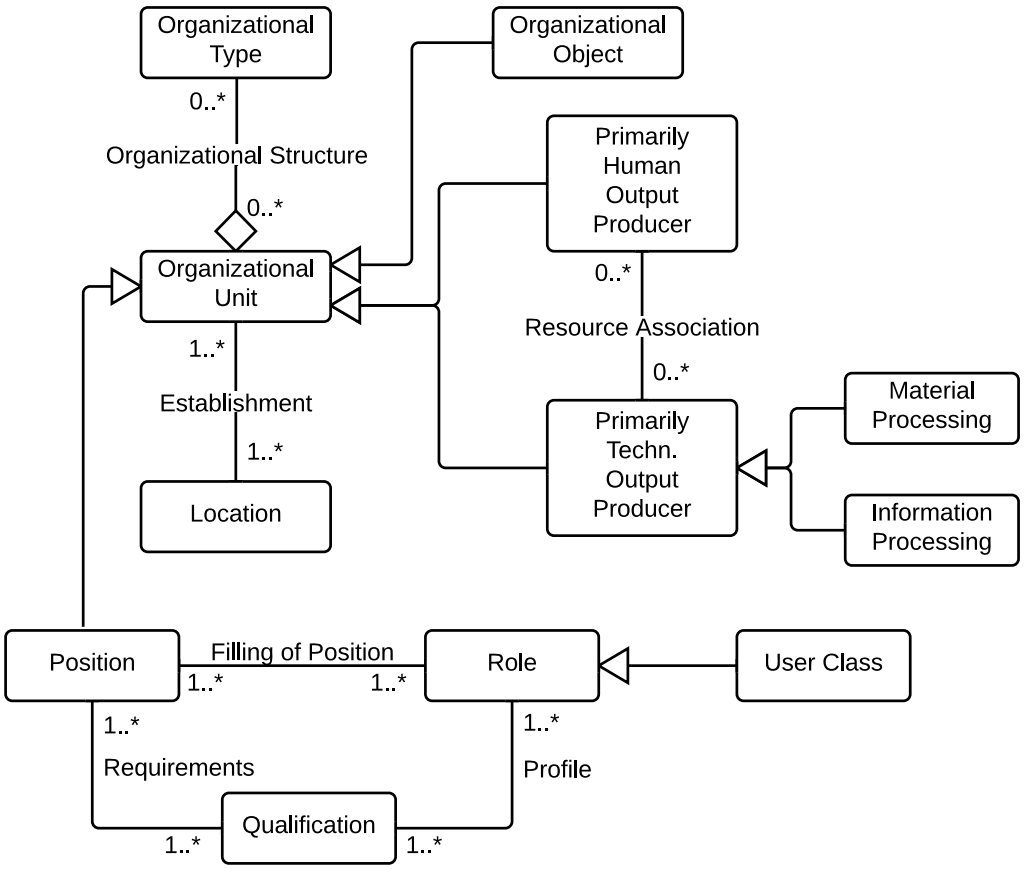


Figure A.11: ARIS organisational view

[14]Picture taken from [119]

The current version of ARIS assigns resources to process activities on the basis of individuals, person types (i.e. roles) and organisational units, so the Direct Distribution, the Role-based Allocation, and the Group-based Distribution patterns are supported by the system. The specification of SoD and BoD constraints is also allowed by means of attributes Segregation of Duties and Commit Employee, respectively, which can be set for a task. Therefore, the SoD, BoD and Case Handling patterns are also covered.

In ARIS, human resources are associated to Human Tasks and Notification Tasks. It supports indicating the potential owners/performers (i.e. task duties Responsible and Accountable) and an email notification recipient (i.e. task duty Informed), for both kinds of activities. For the former, the notification recipient represents the person being notified that the activity is still pending. For the latter, it is the target of the task.

ARIS follows an all-in-one binding strategy.

A.16 Enterprise Ontology

The *Enterprise Ontology (EO)* was developed as part of the Enterprise Project [63], a collaborative effort to provide a method and a computer tool set for enterprise modelling. It includes a great variety of carefully defined terms which are widely used for describing enterprises in general. Regarding the organisational structure, the main concept handled in the ontology is the organisational unit, which ranges from a single person to the whole corporation. Organisational units can be divided into levels. "The terms enterprise and organisation are not defined in the ontology, but a user of the ontology may wish to define one or other of them as a high-level organisational unit, perhaps corresponding with highest units in the scope of interest" [126].

EO specifies what it calls *actor roles*, which comprise the concepts of performer, owner and delegate, so the Responsible and Accountable task duties are supported by the ontology. Organisational units may be responsible for activities, and so become performers of the tasks (i.e. Group-based Allocation). Persons and roles can be used to assign resources to the latter actor role. As stated in [126], "the concept of a role is not directly represented in the formal EO. Instead, a role corresponds to the semantics of an argument in a relation". Therefore, the Direct Distribution and the Role-Based Distribution patterns are somehow supported as well.

Resources are associated to tasks in EO. The binding strategy to be used is not mentioned in the ontology specification, so it depends on the actual implementation

developed.

A.17 DU ET AL.

Du et al. introduced the fundamentals for a resource management system addressing distributed resource management in [56]. In such a system, "there are *Local Resource Managers (LRMs)* that pre-exist and have their own resource models and communication protocols, and *Global Resource Managers (GRMs)* that represent integrated views of part or all of the underlying LRMs. GRMs have the same resource model and communication protocol, and can be further subdivided into Enterprise GRMs (or *Enterprise GRMs (ERMs)*) and Site GRMs (or *Site GRMs (SRMs)*)". Therefore, they present a three-level hierarchy of resource managers. The organisational model of the resource managers "contains a hierarchical collection of concepts representing *resource types*". Resource types can have two kinds of attributes: (i) *capabilities*, used to build the hierarchy inherited lower in it; and (ii) *non-capabilities*, only applicable to them. As stated in [56], "a resource model defines static behaviors of resource types (e.g., things they can do) and the relationships among them. It also contains the knowledge of *where* to get instances of that type. Dynamic behaviors and relationships (e.g., a resource is only allowed to do a task under certain changing condition) are specified using policies".

The original model has been extended by introducing the concept of *role*. "Roles are a boolean expression specifying the resource types needed for the activity, e.g. $Role_{Act1} : R1 = (Programmer and Analyst) and (Computer or Secretary)$. They constitute virtual nodes. [...] Rules are only stored in the virtual nodes and thus are only associated with roles (not resource types)". Figure §A.12[15] shows an example of such an extended hierarchical model.

Three *Structured Query Language (SQL)*-like languages have been developed to specify resource assignments (called queries by the authors) on the basis of this distributed organisational structure [75].

- *Resource Query Language (RQL)* is composed of SQL *select* statements augmented with optional activity specifications (for clauses). The *select* clause is mandatory and may contain either a resource type (e.g., Programmer) for simple WF activities that only require a single resource, or a role specification. Therefore, Direct Distribution and Role-Based Distribution are two assignment patterns supported

[15] Picture taken from [56]

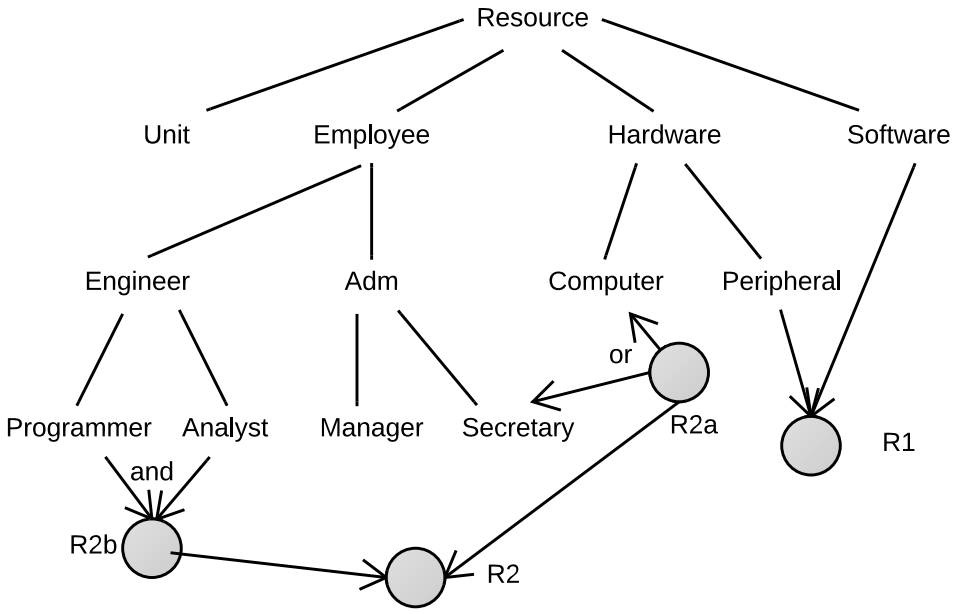


Figure A.12: Resource hierarchy extended with roles

by the system. Filtering with the attributes associated to a resource is possible, so Capability-Based Distribution is also covered.

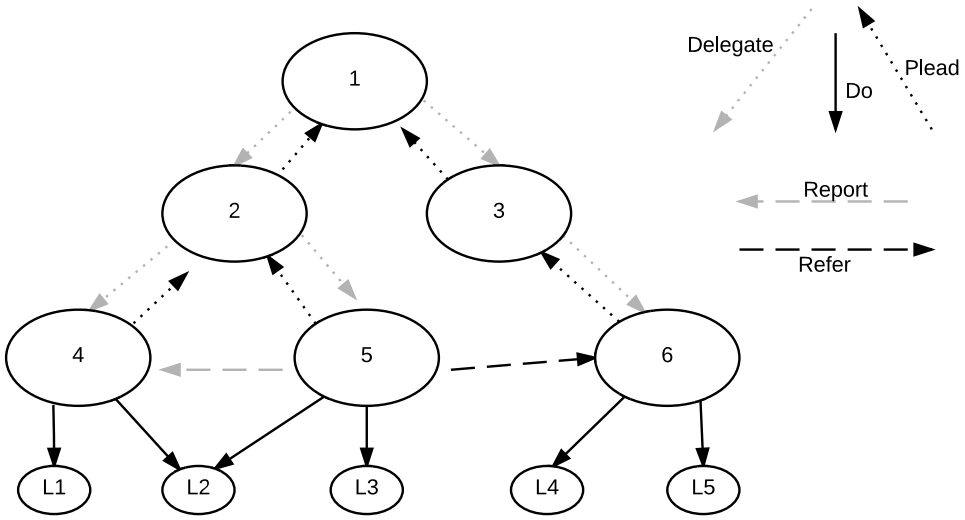
- *Resource Policy Language (RPL)* allows managers/supervisors to define qualification, requirement and substitution policies, by means of clauses *qualify*, *require* and *substitute*, respectively. The first states that a type of resource is qualified to do a type of activities. Requirement policies define additional conditions a resource must satisfy in order to perform a given WF activity. The last type specifies possible substituting resources for a given WF activity in case the originally specified primary resource is not available.

- *Resource Definition Language (RDL)* allows for graphical modeling and manipulation of resource groups.

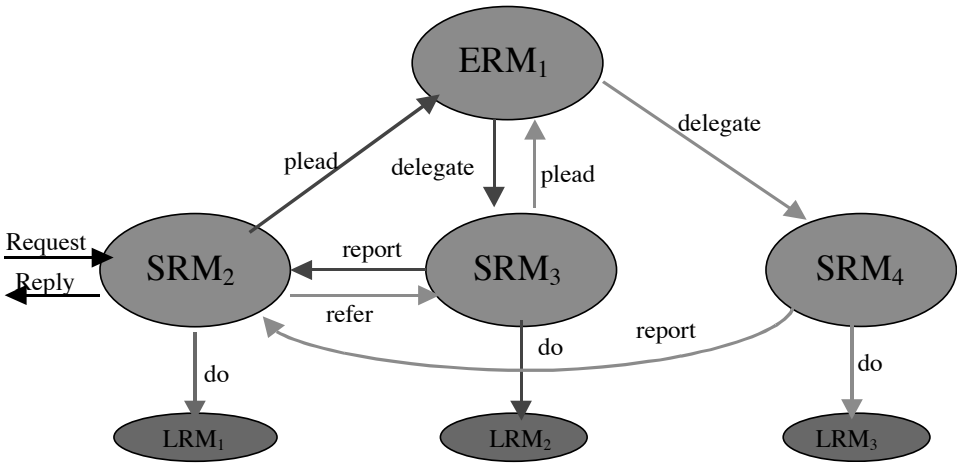
Furthermore, the RM-RM (cf. Figure §A.13[16]) protocol is introduced in order to manage the communication between two GRMs. Figure A.13(a) shows a graphical description of the functions implemented, and Figure A.13(b) shows an example with the real resource manager levels of the system.

No task duties apart from the resource Responsible for the WF activities, are mentioned in the approach, which adopts the separate binding strategy.

[16]Figure taken from [56]



(a) Functions



(b) Use

Figure A.13: The RM-RM protocol

A.18 Team-Enabled Workflow Reference Model

In [131], Van der Aalst and Kumar introduce the concept of work team in a basic organisational metamodel, which initially consisted of resources and roles (or resource class), where a role referred to a group of resources with similar characteristics on the basis of functional requirements (e.g., qualifications, capabilities, or competences), organisational requirements (e.g., teams, organisational units, branches, or departments), or positions within organisational entities (e.g., department head, dean of a faculty). Therefore, the term role was used in a broader sense than is common WF literature. In the original metamodel, tasks were assigned to resources by means of their roles.

Since roles involve organisational units, positions, capabilities and the like, Role-Based Allocation, Group-based Allocation, Position-based allocation, and Capability-based allocation are supported. Furthermore, "one role could be a subclass or superclass of another role, so that if role A is a subclass of role B, then resources with role A can execute tasks mapped onto role B" [131]. It conforms a role-based hierarchy.

Extrapolating that organisational metamodel to teamwork in a company, the following concepts (related to each other s shown in Figure §A.14[17]) come up:

- A *team* can be defined as a group of resources (i.e., workers, participants) working together on a single work item. In existing WF systems work items are distributed over resources. Some teams are created on-the-fly. Others are of a more permanent nature and handle many activities.

- A *team type* does not refer to specific resources but can be seen as the role concept extended to teams. As stated in [131], "a team type refers to a structure which corresponds to a group of resources having certain properties with respect to the composition of the team in terms of sizes and roles of its members".

- A *team position* is a specified role within a team.

- *Contributions* are produced by resources within the context of an activity and link team positions to concrete resources. Without such a notion, the relationship between resources within a team and team positions remains undefined.

A *role* can be considered as a special team type consisting of only one team position. Thus, the previous association of tasks to roles is replaced by an association between task and team type. For those tasks requiring one resource, having a specific role, a singleton team type is introduced, i.e., a team type with one position of cardinality 1.

Constraints can be added at different levels by using OCL rules. Among them, the SoD, BoD and Case Handling patterns described in Section §3.2.3, are supported. However, "it is important to note that roles are associated with the actual execution of work and not with issues like responsibility and accountability" [131].

The binding strategy is not specified. We believe any of them could be valid.

[17] Picture taken from [131]

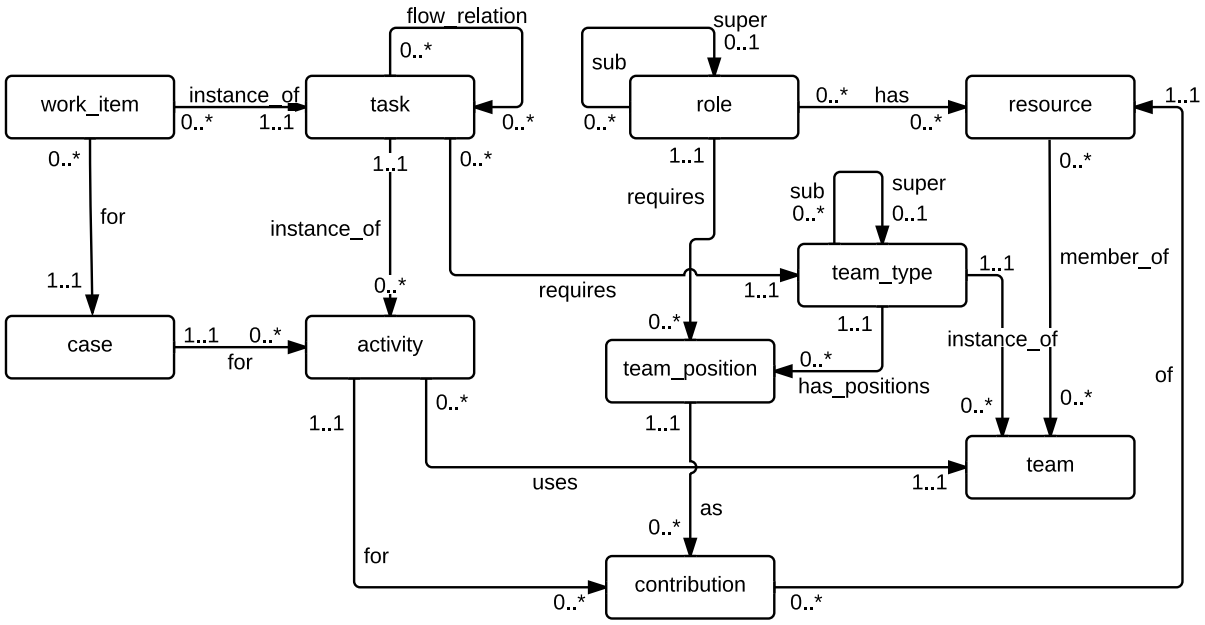


Figure A.14: Team-Enabled Wokflow Reference Model

A.19 TAN ET AL.

The work by Tan et al. is mainly focused on checking consistency issues in WFs [124]. But, they argue that the authorization schema must include the specification of two types of constraints in order to deal with security policies that are not inherent in common resource specification models, such as the so-called *entailment constraints* defined between two WF activities (e.g. SoD and BoD).

They rely on the RBAC model that contains a role hierarchy and in which, if a role is authorized to play a task, the roles dominating that role inherit the execution authorization (cf. Section §A.1 for explicit information about RBAC). However, they argue that inheritance should prohibited when dealing with entailment constraints because it may be problematic. The authors then extend the RBAC schema, which associates roles with tasks in the WF specification (assignment pattern Role-Based Distribution), to define a constrained WF authorization schema including two types of constraints:

- Entailment constraints, to specify restrictions between the resource assignments of two tasks or task instances in a WF model.
- Cardinality constraints, which impose restrictions on the number of users or roles required to execute a task or a set of tasks. Local cardinality constraints are used

by the authors to specify SoD and BoD constraints (i.e. patterns SoD, BoD and Case Handling are covered). Global cardinality constraints are used to indicate that some subset of tasks in the process must be executed by a number of roles greater than some threshold value. Therefore, the Restricted Team Size pattern can be considered to be supported.

A task can be assigned to at most one local cardinality constraint and at most one global cardinality constraint. There is no restriction on the number of entailment constraints that can be assigned to a task. Inter-case constraints are not considered, that is, they focus on a single WF instance. The authors assume that different roles implies different users.

In this scenario, only the potential performers of an activity are specified (i.e. the Responsible task duties), leaving aside other task duties (cf. Section §3.2.2 for a definition of possible task duties).

Regarding the binding strategy, the authors do not specify how to proceed to define the resource assignments and the constraints. Actually, any of the binding strategies described in Section §3.2.4 could fit in this approach.

A.20 HUMAN RESOURCE METAMODEL

Koschmider et al. presented an approach for proper role assignment based on the Hidden Markov Model [78]. They used the HRMM, a part of the RML metamodel introduced by Oberweis and Schuster [93] that combines approaches known by other resource metamodels in BPM [**?**] with competence descriptions as utilized in human resource management.

The HRMM is depicted in Figure §A.15[18]. As stated in [78], "central concepts are: HumanResource, Role, OrganisationalUnit and the competence related modeling objects Competence, Skill and Knowledge. [...] Human resources (HumanResource) are used to depict manpower. To represent their integration in organisational structures human resources can be associated to organisational roles (OrganisationalRole) and positions (OrganisationalPosition). Organisational hierarchies are detailed and reflected by the elements OrganisationPosition, OrganisationalUnit and their associated relationships *hasAdvisor* and *isAdvisorTo* of OrganisationalRole. Roles may be de-

[18]Picture taken from [78]

tailed by rights (Privilege), obligations (Duty) and predefined communication channels (CommunicationPath), e.g., to model escalation mechanisms. Furthermore organisational roles can be used to determine appropriate resources for task execution. [...] The metamodel allows definitions of organisational aspects and hierarchies; furthermore, it allows an explicit extension of these structures by descriptions of competences (in particular competences, skills and knowledge). The competences can be modeled independently and may be reused for further specifications of roles and human resources. With the intention of empowering assignment strategies, it is essential to know that competences, skills and knowledge can be detailed by a level of proficiency; furthermore, competences may require other competences, skills and knowledge, while skills can require other skills and knowledge. Additionally, competences can be prioritized by a correlation coefficient".

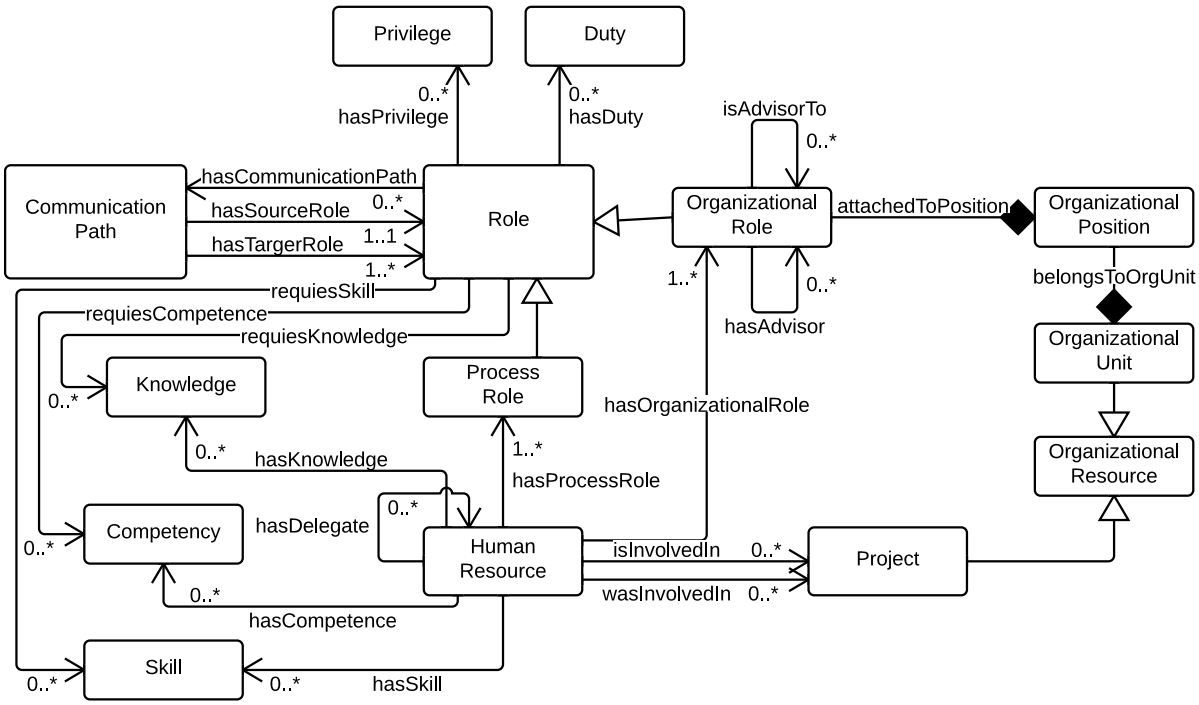


Figure A.15: Human Resource Meta-Model

As the goal of the approach is to automate resource assignment from history data, it is not concerned with specifying how to express the resource assignments. Anyway, as shown in [78], the assignment after the statistical and probabilistic analysis is based on roles (i.e. Role-Based Distribution).

Regarding the task duties, we did not find evidence of task duties different from

Responsible. The due binding strategy to specify the role-based assignments is not mentioned, either.

A.21 RACI

The RACI matrices provide a way to plan, organize and coordinate work, and consist of assigning different degrees of responsibility to the members of an organisation for each activity undertaken in the company, such as who is in charge of performing the activity and who must be informed once the action is complete [44]. They are, thus, focused on indicating the task duties with regard to every task undertaken in an organisation, and every person belonging to it.

In their standard modality, they are utilized to associate activities with (human) resources, typically by using organisational roles [118]. Therefore, they typically offer Role-Based Distribution (cf. assignment patterns in Section §3.2.3). Nonetheless, they are independent from the organisational structure of the company, and any other resource-related entity (i.e. person, group) could be used to assign the activities. Thus, patterns such as Direct Distribution or Organisational Allocation could also be considered to be supported.

Table A.1: Example RASCI matrix

	Project's PhD Student	**PhD Thesis Supervisor**	**Project Coordinator**	**Project's Administrative Assistant**	**Research Group's Clerk**
Submit Paper	R/A				
Fill Travel Authorization	R		A/C		
Sign Travel Authorization	I		R/A		
Send Travel Authorization	I				R/A
Register at Conference	R/A	I	C/I	I	
Make Reservations	R/A	C	C	C/I	S

Figure §A.1 illustrates an example of a RACI matrix. The rows represent *activities* undertaken in a company, the columns of the matrix are *(human) resources*, and each cell contains zero or more *RACI initials* indicating the type of responsibility of *such a* resource on *such an* activity.

As stated in Section §3.2.2, these initials are called *roles* in RACI, although we have

usually used the term *RACI role(s)* in this thesis in order to differentiate them from typical organisational roles:

- *Responsible (R):* person who must perform the work, responsible for the activity until the work is finished and approved by an accountable. There is typically only one person responsible for an activity.
- *Accountable - also Approver or Final Approving Authority - (A):* person who must approve the work performed by the person responsible for an activity, and who becomes responsible for it after approval. There must be one and only one accountable for each activity.
- *Consulted - sometimes Counsel - (C):* this role involves the people whose opinion is sought while performing the work, and with whom there is two-way communication.
- *Informed (I):* person who is kept up-to-date about the progress of an activity and/or the results of the work, and with whom there is just one-way communication. There may be more than one informed person for an activity.

There are several variants of the original version of RACI matrices. Some are based on extending the number of RACI roles to be considered for every activity, e.g, RASCI or RACI-VS. Others give different meanings to the RACI initials. Among them, RASCI matrices are of special interest to us, since they add a new function somehow related to the delegation of work:

- *Support (S):* people who may assist in completing an activity, i.e., the person in charge can delegate work to them. Unlike *Consulted*, who may provide input information to the activity (i.e., information helpful to perform some work), *Support* will actively contribute in the completion of the activity.

The fact of counting on human support for performing the BP activities implies implementing the Additional Participants assignment pattern.

Obviously, there is a separate binding strategy in this approach.

Approaches Dealing with Resources Analysis in Business Processes

This appendix presents an overview of the main proposals for resource analysis found in literature, studied according to the foundations described in Section §4.2, that is, the analysis operations implemented, the BP perspectives involved in the analysis, the BP lifecycle phase(s) at which the analysis operations are performed, and the analysis technique used, if specified. We have focused on the field of WF and BP management, where a number of seven approaches have been found.

B.1 Bertino et al.

As described in Section §A.3, Bertino et al. introduced a *Constraint Specification Language (CSL)* in [20]. Still, another goal of that work was to perform *consistency* analysis and planning, for which the authors developed a three-step procedure dealing with static and dynamic resource analysis. They call the resulting system *constraint analysis and enforcement module*.

Regarding consistency checking, Bertino et al. state that "intuitively, a CB is consistent if and only if the constraints it encodes are satisfiable. [...] It is necessary to verify that the facts belonging to the CB model do not express contradictory information. [...] Finally, it is necessary to verify that for each task there exists at least a user playing a certain role that, when assigned to the task, ensures constraint satisfiability". They perform this checking in the different phases of the methodology they propose, which is summarized below according to the description provided in [20]. Please, note that the authors make two significant assumptions: (i) a WF consists of several tasks to be executed *sequentially*; and (ii) all the tasks execute successfully. The former implies that control flow is not considered in consistency checking.

1. Static Analysis Phase. This phase verifies the consistency of the static part of the CB. If the check fails, the constraints specified for the WF are inherently inconsistent, and thus no assignment to tasks is generated. The system security officer

(or WF designer) has to modify the role assignments to tasks and/or the constraints. If this phase succeeds, it determines a set of *obliged roles* for each task, if any, for which it is *mandatory* to execute the task, and a set of incorrect *denied role assignments*, that is, a set of roles that, when assigned to a task, do not satisfy the constraints.

2. Pruning Phase. This phase modifies the WF role specification to take into account the results of the static analysis phase. Moreover, CB is modified to eliminate redundant rules. Pruning will make the execution of the subsequent steps more efficient.

3. Planning Phase. This phase receives the modified workflow and the modified CB generated by the pruning phase as input and generates the set of possible assignments of roles and users to tasks so that all the constraints associated with the WF are satisfied. Two steps are carried out to perform the resource assignment: (i) the Role Planning Phase; and the (ii) User Planning Phase; in the order they have been mentioned. If no assignment can be generated, an error is returned to the system security officer.

   The planning phase, intended to perform an initial plan, is executed before the WF execution starts. Therefore, this approach focus on the Design and Analysis phase of the BP lifecycle (cf. Section §2.4 for a description of such a lifecycle).

Please, notice that in the described work, the authors do not solve resource assignments in order to calculate the set of potential performers for tasks, as most of the BPMSs and similar approaches do. Instead, they evaluate the *consistency* of the WF model according to the resource assignments established, and suggest possible assignments that eliminate redundancy and ensure the correct operation of the WF at run time.

B.2 Business Activities

In [120], where the Business Activity RBAC Models were introduced (cf. Section §A.4 for a brief summary about resource specification with Business Activities), Strembeck and Mendling stated that "there are two different types of correctness that need to be considered: static correctness and dynamic correctness. *Static correctness* refers to the logical consistency of the elements and relationships in the Business Activity

RBAC Model. It is static in the sense that it refers to the design-time of the model, i.e. it refers to process types and task types. Dynamic correctness relates to the compliance of process instances with the mutual exclusion and binding constraints at runtime. Thus, it is dynamic in the sense that it refers to the runtime execution of a particular process". They relied on Petri Nets for *consistency* checking, building the reachability graph of the BP. Then, they defined a set of constraints with which the reachability graph must comply, and identified the states corresponding to process models dynamically correct. Furthermore, a set of potential conflicts derived from the addition of access-control constraints in the models were mentioned in that work.

As an extension of the Business Activities, Schefer et al. developed a set of algorithms to detect potential resource-related conflicts at the level of design-time constraint definition, design-time assignment relations, and runtime task allocation [115], some of which had already been introduced in [120].

Regarding constraint definition, the authors in [115] argue that "when defining SSoD, DSoD, and BoD based on users and/or roles constraints at design-time, a number of conflicts may occur that would lead to inconsistencies in the corresponding process-related RBAC model". Specifically, they have developed a set of algorithms aimed at checking whether the definition of new access-control constraints in a model leads to conflicts in the process. They also propose resolution procedures and strategies for the potential problems derived from the addition of new constraints, e.g. changing a type of constraint for other, or removing constrains.

The same is done for the other two issues addressed. With regard to the design-time assignment relations, it is stated that "assignment conflicts arise at design-time when defining new assignment relations between roles, subjects, and tasks". They thus define some algorithms to check the design-time consistency of a process-related RBAC model when defining a task-to-role, role-to-role, or role-to-subject assignment relation. Resolution strategies for the identified problems are also presented. Finally, as far as run-time task allocation is concerned, some runtime conflicts may arise when actually enforcing constraints during the execution of the process. In particular, mutual-exclusion and binding constraints directly impact the allocation of tasks to subjects. Specifically, the authors have identified five potential conflicts when allocating a concrete task instance to a certain subject, and propose three resolution strategies to deal with them.

As a conclusion from this work, we have identified two main analysis operations being addressed: (i) the resolution of the resource assignments, i.e. calculation of the

potential performers; and (iii) the checking of the consistency at different levels. Notice that they work both at design time and at run time, thus dealing with resource analysis at the Design and Analysis, and Enactment phases of the BP lifecycle (cf. Section §2.4 for a summary of the BP lifecycle). The technique or formalism used to perform the checks is not specified in [115]. As stated in [120], they use Petri Nets to check the issues related to the dynamic behaviour of the process, i.e. the control flow perspective.

B.3 WIDE

In the *Workflow on Intelligent Distributed database Environment (WIDE)* WF management system, the actual assignment of tasks (instances) to agents (instances) are performed at run time. As stated in the WIDE specification [40], "this assignment may either be performed by a scheduler, where the scheduler engine of the WfMS assigns the task to the best individual actor of that set according to the policy of work assignment (automatic assignment), or chosen directly for execution by a user (manual assignment, which may be computer assisted to identify priorities and critical cases). [...] The Local Scheduler module is responsible for dispatching requests for allocation of tasks to agents. This modules uses different criteria when choosing between a set of agents who can perform a task. Workload, availability of agents, and priorities are some of the available discrimination mechanism. This module can also be instructed to assign directly a task to a given agent. This module is part of the Scheduling System; this is a distributed service which operates in a network of WIDE nodes. If the Local Scheduler module is not able to satisfy the request in its domain, then re-sends the request to the Global Scheduler modules which finds a suitable Local Scheduler module, in the net, to satisfy the request. In this way, the assignment of tasks to agents can be done in a distributed way".

So, regarding our classification criteria, WIDE allows for the calculation of the potential performers of WF activities at the Enactment BP lifecycle phase, thus taking into account only the BP resource perspective (since information coming from data objects is not said to be involved in the resource assignments). The specific technique or formalism used to solve the resource assignments is not mentioned.

B.4 YAWL

Yet Another Workflow Language (YAWL) 2.0 is equipped with a runtime engine that deals with resource offering and allocation. Therefore, the resource assignments are automatically solved during BP execution, and the tasks are usually offered to the corresponding participants. Then, according to the specification of YAWL's Resource Service, "the work item is allocated to a single participant, so that the participant is committed (willingly or not) to performing that work item at a later time" [8]. It means only the *potential performers* of the tasks are automatically calculated. No other resource-related analysis operations are mentioned in YAWL's specification [9].

B.5 ARIS

Similarly to YAWL, ARIS addresses the resolution of resource assignments at run time, since it is the main feature a BPMS should provide regarding resource management. As stated in [119], "the organisational units that are directly connected to a function or assigned via a function allocation diagram are converted into participants". This means they solve the assignments for the different task duties considered for the activities, resulting in the calculation of the *potential performers* of the corresponding task duty. As for the Informed task duty, the person notified is meant to be given in the data flow of the model. Therefore, we assume that data is also taken into consideration in the resolution of some resource assignments. To the best of our knowledge, no other operations are supported, and design time analysis is outside the scope of ARIS as well.

B.6 Du et al.

As described in Section §A.17, Du et al. introduce a resource management system for enterprise WF environments [56]. One of its key components is the resource engine, "a component of a workflow system that allows run time resource allocation". It is thus in charge of finding resources meeting the conditions established in the resource assignments and constraints. The resource engine has a resource model associated with it (cf. Section §A.17). "If a particular resource is found to satisfy the request, the control engine component returns the result as appropriate. If not, then the request is

sent to the policy engine, where substitution policies are applied, and then it is sent back to the resource engine. If a resource is found, then the result is returned. If the request is still not satisfied and the resource manager has authority over the resource type, then a NULL is returned".

Therefore, the resource engine implements the operation to calculate the potential performers from task assignments at the Enactment phase of the BP lifecycle. Nothing is said about the technique utilized to do so.

B.7 TAN ET AL.

Tan et al. [124] focus on solving the consistency problem of the resource constraints set in a WF, with the aim of helping the WF designers to define a sound constrained WF authorization schema. They state that "Bertino et al. [20] identified three different types of constraints: static, dynamic and hybrid constraints. We are only interested in hybrid constraints because we want to statically check the consistency of such constraints defined in a WF".

They define a CSL to assign resources and roles to tasks, as well as to specify different types of constraints (cf. Section §A.19 for more details about how the approach deals with resource specification). Then, they define consistency rules for constraint-task pairs that guarantee there is no inconsistency, ambiguities and redundancy contained in the set of constraints. "When the constraint-task pairs conform to these rules, then for each user and each role authorized in a task in the WF, there is at least one successful WF instance that satisfies all the constraints" [124]. To do so, they make some assumptions on the WF model, such as the following:

- A task t can be entailed by no more than one task t_1.
- If two or more entailment constraints containing the same task t_1 are associated with the same task t, then they are all either user-based or role-based constraints.
- The predicates are different. A consequence of this assumption is that there are at most two user-based entailment constraints and at most six role-based entailment constraints that can be associated with a task.

Considering these and other assumptions, the approach presents a set of algorithms representing consistency rules that guarantee there is no inconsistency or ambiguities

within these constraints even when they inter-play with each other. This is meant to be done at design time in order to make sure the WF is correct before executing it.

The authors mention nothing about the possible existence of exclusive gateways and/or complex WF structures. Therefore, we assume only the BP resource perspective is taken into account in the approach.

C

Description Logics in a Nutshell

C.1 Description Logics

Description Logics (DLs) are logics that serve primarily for formal description of *concepts*, *roles* (relations between the concepts) and *individuals* (instances of the concepts). Semantically, they are found on predicate logic, but their language is formed so that it would be enough for practical modeling purposes and also so that the logic would have good computational properties such as decidability.

Knowledge representation systems based on DLs consist of two components: TBox and ABox. The TBox describes *terminology*, i.e., the ontology in the form of *concepts* and *property* definitions and their relationships, while the ABox contains *assertions* about individuals using the terms from the ontology.

"The basic form of declaration in a TBox is a concept *definition*, that is, the definition of a new concept in terms of other previously defined concepts" [90]. For example, an instance of a BP $TripManagementInstance$ can be defined as the union of all the activity instances that participate in the process, e.g. $SubmitPaperInstance$ and $MakeReservationsInstance$ by writing this declaration:

$$TripManagementInstance \equiv SubmitPaperInstance \sqcup MakeReservationsInstance$$

Such a declaration is usually interpreted as a logical equivalence, which amounts to providing both sufficient and necessary conditions for classifying individuals. In a DL-based *Knowledge Base (KB)*, thus, a terminology is constituted by a set of concept definitions of the above form. However, there are some important common assumptions usually made about DL terminologies:

- only one definition for a concept name is allowed;
- definitions are acyclic in the sense that concepts are neither defined in terms of themselves nor in terms of other concepts that indirectly refer to them.

This kind of restriction implies that every defined concept can be expanded in a unique way into a complex expression containing only atomic concepts by replacing every defined concept with the right-hand side of its definition. In particular, the basic task in constructing a terminology is *classification*, that consists in placing a new concept expression in the proper place in a taxonomic hierarchy of concepts.

The ABox contains extensional knowledge about the domain of interest, that is, assertions about individuals, usually called membership assertions" [90]. For example,

$$Position(THEOSsProjectCoordinator)$$

$$Role(ProjectCoordinator)$$

states that the *THEOSsProjectCoordinator* individual is a *Position* and the *ProjectCoordinator* individual is a *Role*. Similarly,

$$participatesIn(THEOSsProjectCoordinator, ProjectCoordinator)$$

specifies that the *THEOSsProjectCoordinator* position participates in the *ProjectCoordinator* role. Assertions of the first kind are also called concept assertions, while assertions of the second kind are also called role assertions.

"The basic reasoning task in an ABox is *instance checking*, which verifies whether a given individual is an instance of (belongs to) a specified concept. Although other reasoning services are usually considered and employed, they can be defined in terms of instance checking. Among them we find *knowledge base consistency*, which amounts to verifying whether every concept in the knowledge base admits at least one individual; *realization*, which finds the most specific concept an individual object is an instance of; and *retrieval*, which finds the individuals in the knowledge base that are instances of a given concept. These can all be accomplished by means of instance checking" [90]. Hence, a distinguished feature in DLs is the emphasis on reasoning as a central service, allowing thus to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base.

DLs apply the so-called *open world assumption*. It 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. For instance, if we do not include the previous assertion

$$participatesIn(THEOSsProjectCoordinator, ProjectCoordinator),$$

DLs do not interpret that a *THEOSsProjectCoordinator* does not participate in role *ProjectCoordinator*. Instead it may simply mean that the knowledge is incomplete and/or this fact has not been asserted yet.

C.2 Description Languages

"Elementary descriptions are *atomic concepts* and *atomic roles*. Complex descriptions can be built from them inductively with *concept constructors*" [18]. There are many varieties of Description Languages, and they are distinguished by the constructors they provide. We focus on the family of $\mathcal{AL}$-*languages*. The language $\mathcal{AL}$ (attributive language) allows for atomic negation, concept intersection, universal restrictions and limited existential quantification. More expressive languages can be obtained by adding further constructors and axioms to $\mathcal{AL}$. In the following we list the possible extensions:

$\mathcal{F}$: Functional properties

$\mathcal{E}$: Full existential qualification

$\mathcal{U}$: Concept union

$\mathcal{C}$: Complex concept negation

$\mathcal{H}$: Role hierarchy

$\mathcal{R}$: Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness

$\mathcal{O}$: Nominals

$\mathcal{I}$: Inverse properties

$\mathcal{N}$: Cardinality restrictions

$\mathcal{Q}$: Qualified cardinality restrictions

$^{(\mathcal{D})}$: Use of datatype properties, data values or data types.

Of interest for us is the abbreviation $\mathcal{S}$, that stands for $\mathcal{ALC}$ logic with transitive role. This is the basis for the logics of OWL.

C.3 OWL

OWL DL abstract syntax	DL syntax	Model-theoretic semantics
Descriptions (*C*)		
A (URI reference) owl:Thing owl:Nothing	A $\top$ $\bot$	$A^I \subseteq \Delta^I$ $\text{owl:Thing}^I = \Delta^I$ $\text{owl:Nothing}^I = \varnothing$
restriction(R allValuesFrom(C)) restriction(R minCardinality(n)) restriction(R maxCardinality(n))	$\forall R.C$ $\geq n\, R$ $\leq n\, R$	$(\forall R.C)^I =$ $\{x \mid \forall y. < x, y > \in R^I \rightarrow y \in C^I\}$ $(\geq n\, R)^I =$ $\{x \mid \#\{y. < x, y > \in R^I\} \geq n\}$ $(\leq n\, R)^I =$ $\{x \mid \#\{y. < x, y > \in R^I\} \leq n\}$
restriction(U allValuesFrom(D)) restriction(U minCardinality(n)) restriction(U maxCardinality(n))	$\forall U.D$ $\geq n\, U$ $\leq n\, U$	$(\forall U.D)^I =$ $\{x \mid \forall y. < x, y > \in U^I \rightarrow y \in D^I\}$ $(\geq n\, U)^I =$ $\{x \mid \#\{y. < x, y > \in U^I\} \geq n\}$ $(\leq n\, U)^I =$ $\{x \mid \#\{y. < x, y > \in U^I\} \leq n\}$
unionOf(C_1 C_2 ...)	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^I = C_1^I \cup C_2^I$
Data Ranges (*D*)		
D (URI reference)	D	$D^I \subseteq \Delta_D^I$
Object Properties (*R*)		
R (URI reference)	R ; R^-	$R^I \subseteq \Delta^I \times \Delta^I$; $(R^-)^I = (R^I)^-$
Datatype Properties (*U*)		
U (URI reference)	U	$U^I \subseteq \Delta^I \times \Delta_D^I$
Class Axioms		
Class(A partial $C_1 \ldots C_n$) EquivalentClasses($C_1 \ldots C_n$) DisjointClasses($C_1 \ldots C_n$)	$A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$ $C_1 = \ldots = C_n$ $C_i \sqcap C_j = \bot$, $i \neq j$	$A^I \subseteq C_1^I \cap \ldots \cap C_n^I$ $C_1^I = \ldots = C_n^I$ $C_i^I \cap C_j^I = \varnothing$, $i \neq j$
Property Axioms		
ObjectProperty(R domain(C_1)...domain(C_n) range(C_1)...range(C_m) [inverseOf(R_1)]) DatatypeProperty(U domain(C_1)...domain(C_n) range(D_1)...range(D_m))	 $\geq 1\, R \sqsubseteq C_i$ $\top \sqsubseteq \forall R.C_j$ $R = (^- R_1)$ $\geq 1\, U \sqsubseteq C_i$ $\top \sqsubseteq \forall U.D_j$	 $R^I \subseteq C_i^I \times \Delta^I$, $i = 1,\ldots,n$ $R^I \subseteq \Delta^I \times C_j^I$, $j = 1,\ldots,m$ $R^I = (R_1^I)^-$ $U^I \subseteq C_i^I \times \Delta_D^I$, $i = 1,\ldots,n$ $U^I \subseteq \Delta^I \times D_j^I$, $j = 1,\ldots,m$

Figure C.1: DL summary

The Web Ontology Language (OWL) is a knowledge representing scheme designed specifically for use on the semantic web; it exploits existing web standards (XML and RDF), adding the familiar ontological primitives of object and frame based systems, and the formal rigor of DLs. As exemplified in Table §C.1 and Table §C.2 [1] [2], OWL consists a rich set of knowledge representation constructs that can be used to formally specify RAL-domain knowledge, which in turn can be exploited by DL reasoners for purposes of inferencing, *i.e.*, deductively inferring new facts from knowledge that is explicitly available [21]. As stated in [21], the logical basis of the language means that reasoning services can be provided in order to make OWL described resources more accessible to automated processes thereby allowing one to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base. From a formal point of view, OWL can be seen to be equivalent to a very expressive DL, with an OWL ontology corresponding to a DL terminology (TBox) whereas instance data pertaining to the ontology making up the assertions (ABox). OWL terms *classes*, *properties* and *objects* refer to DL terms *concepts*, *roles* and *individuals*, respectively.

Axiom	**DL Syntax**	**Example**
Sub-class	$C_1 \sqsubseteq C_2$	$OrgTeam \sqsubseteq OrgUnit$
Equivalent class	$C_1 \equiv C_2$	$Capability \equiv Skill$
Disjoint with	$C_1 \sqsubseteq \neg C_2$	$Person \sqsubseteq \neg Position$
Same Individual	$x_1 \equiv x_2$	$Carol \equiv Caroline$
Different from	$x_1 \sqsubseteq \neg x_2$	$BusinessManager \sqsubseteq \neg Secretary$
Sub property	$P_1 \sqsubseteq P_2$	$hasExperience \sqsubseteq hasCapability$
Equivalent property	$P_1 \equiv P_2$	$hasCapability \equiv hasSkill$
Inverse	$P_1 \equiv P_2^-$	$occupies \equiv isOccupiedBy^-$
Transitive property	$P^+ \sqsubseteq P$	$extReportsTo^+ \sqsubseteq extReportsTo$
Functional property	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq reportsTo$
Inverse functional property	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq 1 isReportedBy^-$

Table C.1: OWL axioms

[1] In both tables a syntax commonly used for DLs [18] is utilised.

[2] Note that these tables are no complete, but contain only those elements useful fin the scope of this thesis.

Constructor	**DL Syntax**	**Example**
Intersection	$C_1 \sqcap \cdots \sqcap C_n$	$map(expr1) \sqcap map(expr2)$
Union	$C_1 \sqcup \cdots \sqcup C_n$	$map(expr1) \sqcup map(expr2)$
Complement	$\neg C$	$\neg map(expr)$
One of	$x_1 \sqcup \cdots \sqcup x_n$	$Alex \sqcup Anna$
All values from	$\forall P.C$	$\forall participatesIn.Role$
Some values	$\exists P.C$	$\exists participatedIn.Role$
Has value	$P.\{x\}$	$participatedIn.\{Technician\}$
Max cardinality	$\leq nP$	$\leq 1 isMemberOf$
Min cardinality	$\geq nP$	$\geq 1 hasCapability$

Table C.2: OWL class constructors

BIBLIOGRAPHY

[1] Web Services Business Process Execution Language v2.0. Technical report, OASIS, 2007. (Pages 18, 44).

[2] WS-BPEL Extension for People (BPEL4People). Technical report, OASIS, 2009. (Pages 33, 39, 44, 47, 49, 50, 58, 75, 76, 194, 226).

[3] Web Services-Human Task (WS-HumanTask) v1.1. Technical report, OASIS, 2010. (Pages 33, 39, 44, 47, 49, 50, 58, 76, 226).

[4] XACML 3.0 RBAC Profile. Specification, OASIS, 2010. (Pages 41, 47, 49, 50, 76, 209, 210).

[5] W. M. P. V. D. Aalst and B. F. V. Dongen. ProM: The Process Mining Toolkit. *Industrial Engineering*, 489:1–4, 2009. (Pages 58, 184).

[6] G. Abrami, O. Barreteau, and F. Cernesson. An Agent-Group-Role based modelling framework for participative water management support. In *iEMSs*, 2002. (Page 32).

[7] R. Acerbis, A. Bongio, M. Brambilla, S. Butti, S. Ceri, and P. Fraternali. Web Applications Design and Development with WebML and WebRatio 5.0. In *TOOLS (46)*, pages 392–411, 2008. (Page 202).

[8] M. Adams. The Resource Service. In *Modern Business Process Automation*, pages 261–290. 2010. (Pages 4, 21, 35, 58, 219, 220, 243).

[9] M. Adams. *YAWL v2.3-User Manual*, 2012. (Pages 24, 31, 35, 43, 47, 49, 50, 54, 58, 60, 62, 63, 76, 77, 78, 201, 218, 243).

[10] G. D. Alexander Grosskopf and M. Weske. *The Process. Busisness Process Modeling using BPMN*. Megan-kiffer Press, March 2009. (Page 16).

[11] I. American National Standards Institute. Role-Based Access Control. ANSI INCITS 359-2004. http://csrc.nist.gov/rbac, February 2004. (Pages 30, 207).

[12] V. Andrikopoulos, S. Benbernou, M. Bitsaki, O. Danylevych, M. Hacid, W. van den Heuvel, D. Karastoyanova, B. Kratz, F. Leymann, M. Mancioppi, K. Mokhtari, C. Nikolaou, M. Papazoglou, and B. Wetzstein. Survey on Business Process Management. Technical report, 2008. (Pages 17, 21, 23, 24, 26).

[13] ARIS. RACI, ARIS Community Website. http://www.ariscommunity.com/raci, 2012. (Pages 40, 106).

[14] A. Awad. *A compliance management framework for business process models*. PhD thesis, University of Potsdam, 2010. (Page 21).

[15] A. Awad, G. Decker, and M. Weske. Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In M. Dumas, M. Reichert, and M.-C. Shan, editors, *BPM*, volume 5240 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2008. ISBN 978-3-540-85757-0. (Pages 79, 142, 202).

[16] A. Awad, G. Decker, and N. Lohmann. Diagnosing and Repairing Data Anomalies in Process Models. Technical report, Potsdam, Germany, 2009. (Pages 59, 163, 164, 165).

[17] A. Awad, A. Grosskopf, A. Meyer, and M. Weske. Enabling Resource Assignment Constraints in BPMN. Technical report, BPT, 2009. (Pages xvii, 30, 33, 36, 40, 43, 44, 47, 49, 50, 69, 76, 218, 224, 225).

[18] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logics Handbook: Theory, Implementations, and Applications*. Cambridge University Press, 2003. ISBN 0521781760. (Pages 121, 124, 249, 251).

[19] D. Benavides, S. Segura, and A. R. Cortés. Automated analysis of feature models 20 years later: A literature review. *Inf. Syst.*, 35(6):615–636, 2010. (Pages 7, 52).

[20] E. Bertino, E. Ferrari, and V. Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, 2:65–104, February 1999. ISSN 1094-9224. (Pages 40, 41, 42, 47, 49, 50, 54, 59, 61, 62, 63, 76, 77, 78, 210, 226, 239, 244).

[21] M. Bhatt, W. Rahayu, S. P. Soni, and Carlo. Ontology driven semantic profiling and retrieval in medical information systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(4):317–331, 2009. (Page 251).

[22] E. Bottazzi and R. Ferraio. Preliminaries to a DOLCE ontology of organisations. *Int. J. of Business Process Integration and Management*, 4(4):225–238, 2009. (Page 32).

[23] M. Brambilla, P. Fraternali, and C. K. V. Ruiz. Combining social web and BPM for improving enterprise performances: the BPM4People approach to social BPM. In *WWW (Companion Volume)*, pages 223–226, 2012. (Page 201).

[24] J. Bronkhorst. RACI matrices - how difficult can it be? HP's Website, June 2010. http://h30507.www3.hp.com/t5/ITILigent-Service-Management/RACI-matrices-how-difficult-can-it-be/ba-p/41138. (Page 106).

[25] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Hints on how to face business process compliance. In *III Taller de Procesos de Negocio e Ingeniería de Servicios (PNIS'10) en JISBD'10*, volume 4, pages 26–32, 2010. (Pages 10, 21, 199).

[26] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. On the identification of data-related compliance problems in business processes. In *VI Jornadas Científico-Técnicas en Servicios Web y SOA (JSWEB'10)*, pages 89–102, 2010. (Pages 10, 59, 198).

[27] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Defining and Analysing Resource Assignments in Business Processes with RAL. In *ICSOC*, volume 7084, pages 477–486, 2011. (Pages 9, 10, 59, 198).

[28] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Exploring Features of a Full-Coverage Integrated Solution for Business Process Compliance. In *CAiSE 2011 Workshops (GRCIS'11)*, volume 83, pages 218–227, 2011. (Pages 10, 186, 199).

[29] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Mixing RASCI Matrices and BPMN Together for Responsibility Management. In *VII Jornadas en Ciencia e Ingeniería de Servicios (JCIS'11)*, volume 1, pages 167–180, 2011. (Pages 10, 195, 197).

[30] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes. In *Business Process Management Workshops (BPD'11)*, pages 50–61, 2011. (Pages 8, 10, 179, 196).

[31] C. Cabanillas, M. Resinas, A. Ruiz-Cortés, and A. Awad. Automatic Generation of a Data-Centered View of Business Processes. In *CAiSE*, volume 6741, pages 352–366, 2011. (Pages 10, 160, 198, 201, 202).

[32] C. Cabanillas, A. del Río-Ortega, M. Resinas, and A. Ruiz-Cortés. CRISTAL: Collection of Resource-centrIc Supporting Tools And Languages. In *BPM 2012 Demos*, volume 940, pages 51–56, 2012. (Pages 10, 178, 199).

[33] C. Cabanillas, A. del Río-Ortega, M. Resinas, and A. Ruiz-Cortés. RAL Solver: a Tool to Facilitate Resource Management in Business Process Models. In *VIII Jornadas de Ciencia e Ingeniería de Servicios (JCIS'12)*, 2012. (Pages 10, 180, 199).

[34] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Automated Resource Assignment in BPMN Models using RACI Matrices. In *OTM 2012 (CoopIS'12)*, volume 7565, pages 56–73, 2012. (Pages 9, 10, 182, 197).

[35] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Designing Business Processes with History-Aware Resource Assignments. In *BPM 2012 Workshops (BPD'12)*, volume 132, pages 101–112, 2012. (Pages 8, 10, 198).

[36] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Integrando las matrices RASCI en BPMN para la Gestión de la Responsabilidad. *Novática: Revista de la Asociación de Técnicos de Informática*, 216:62–68, 2012. (Pages 10, 197).

[37] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Introducing a Mashup-Based Approach for Design-Time Compliance Checking in Business Processes. In *CAiSE 2012 Workshops (GRCIS)*, volume 112, pages 337–350, 2012. (Pages 10, 186, 200).

[38] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Summary of "Defining and Analysing Resource Assignments in Business Processes with RAL". In *VIII Jornadas de Ciencia e Ingeniería de Servicios (JCIS'12)*, 2012. (Page 198).

[39] G. Cabri, L. Leonardi, and F. Zambonelli. BRAIN: A Framework for Flexible Role-Based Interactions in Multiagent Systems. In R. Meersman, Z. Tari, and D. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 145–161. Springer Berlin / Heidelberg, 2003. (Page 32).

[40] F. Casati, P. Grefen, B. Pernici, G. Pozzi, and G. Sanchez. WIDE workflow model and architecture, 1996. (Pages 31, 42, 47, 49, 50, 54, 60, 62, 63, 76, 77, 78, 213, 214, 215, 242).

[41] F. Casati, S. Castano, and M. Fugini. Managing Workflow Authorization Constraints through Active Database Technology. *Information Systems Frontiers*, 3:

319–338, September 2001. ISSN 1387-3326. (Pages xvii, 30, 42, 47, 49, 50, 76, 215, 216).

[42] D. D. Clark and D. R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *1987 IEEE Symposium on Security and Privacy*, pages 184–194. IEEE Computer Society Press, 1987. (Page 207).

[43] W. M. Coalition. The Workflow Management Coalition Specification, Web site. http://www.aiai.ed.ac.uk/project/wfmc/ARCHIVE/DOCS/glossary/glossary.html, Last accessed in December, 2012 1996. (Page 26).

[44] D. O. Conchúir. Human Resource Management Processes. In *Overview of the PMBOK® Guide*, pages 129–145. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-19122-0. (Pages 33, 46, 49, 50, 76, 102, 237).

[45] G. E. Consulting. Process Analysis, Web site. http://www.bpmcenterofexcellence.com/id58.html, Last accessed in December 2012. (Page 5).

[46] T. H. Davenport. *Process innovation: reengineering work through information technology*. Harvard Business School Press, Boston, MA, USA, 1993. ISBN 0-87584-366-2. (Page 16).

[47] M. de Leoni, W. M. P. van der Aalst, and B. F. van Dongen. Data- and Resource-Aware Conformance Checking of Business Processes. In *BIS*, pages 48–59, 2012. (Page 58).

[48] G. Decker. *Design and Analysis of Process Choreographies*. PhD thesis, University of Potsdam, 2009. (Pages 4, 16, 19).

[49] G. Decker, H. Overdick, and M. Weske. Oryx - An Open Modeling Platform for the BPM Community. In *Proceedings of the 6th International Conference on Business Process Management*, pages 382–385. Springer-Verlag, 2008. ISBN 978-3-540-85757-0. (Page 180).

[50] A. del Río-Ortega, M. Resinas, and A. R. Cortés. Defining Process Performance Indicators: An Ontological Approach. In *OTM Conferences*, pages 555–572, 2010. (Page 201).

[51] A. del Río-Ortega, M. Resinas, C. Cabanillas, and A. Ruiz-Cortés. On the Definition and Design-Time Analysis of Process Performance Indicators. *Information*

Systems, page In Press. Available at http://www.sciencedirect.com/science/article/pii/S0306437912001469, Accepted 2012. (Page 200).

[52] J. Desel and J. Esparza. *Free choice Petri nets*. Cambridge University Press, New York, NY, USA, 1995. (Page 142).

[53] V. Dignum. *A model for organizational interaction: based on agents, founded in logic*. PhD thesis, 2004. (Page 32).

[54] R. M. Dijkman, M. Dumas, B. F. van Dongen, R. Käärik, and J. Mendling. Similarity of business process models: Metrics and evaluation. *Inf. Syst.*, 36(2):498–516, 2011. (Pages 58, 202).

[55] H. Dresner. Business Activity Monitoring: BAM Architecture, 2003. (Page 23).

[56] W. Du, J. Davis, Y.-N. Huang, and M.-C. Shan. Enterprise Workflow Resource Management. In *RIDE*, pages 108–115, 1999. (Pages 40, 45, 47, 49, 50, 54, 60, 62, 63, 76, 77, 78, 230, 231, 243).

[57] J. Eder and W. Liebhart. Workflow Recovery. In *CoopIS*, pages 124–134, 1996. (Pages 4, 19).

[58] H. Enderton. *Elements of Set Theory*. Acad. Press, 1977. (Page 70).

[59] J. Ferber and O. Gutknecht. Aalaadin: a meta-model for the analysis and design of organizations in multi-agent systems. In Y. Demazeau, editor, *Proceedings of the Third International Conference on Multi-Agent Systems, ICMAS'98*, pages 128–135, Paris, France, July 1998. IEEE Computer Society. (Page 32).

[60] D. F. Ferraiolo and D. R. Kuhn. Role-Based Access Controls. In *15th National Computer Security Conference*, 1992. (Page 207).

[61] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4:224–274, August 2001. ISSN 1094-9224. (Pages 41, 49, 50, 76, 207, 208, 209).

[62] U. R. T. Force. OMG Unified Modeling Language Specification, Version 1.4 (final draft), Februrary 2001. (Pages 5, 18).

[63] J. Fraser and A. Tate. The Enterprise Tool Set - An Open Enterprise Architecture. In *Workshop on Intelligent Manufacturing Systems (IJCAI'95)*, 1995. (Page 229).

[64] J. M. García, D. Ruiz, and A. R. Cortés. Improving semantic web services discovery using SPARQL-based repository filtering. *J. Web Sem.*, 17:12–24, 2012. (Page 202).

[65] A. Ghose and G. Koliadis. Auditing Business Process Compliance. In *ICSOC*, pages 169–180, 2007. (Page 202).

[66] G. Governatori and S. Sadiq. The Journey to Business Process Compliance. In *Handbook of Research on BPM*, pages 426–454. IGI Global, 2009. (Pages 59, 202).

[67] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.-C. Shan. Business Process Intelligence. *Computers in Industry*, 53(3):321–343, 2004. (Page 23).

[68] K. Grigorova. Process modelling using Petri nets. In *Proceedings of the 4th international conference conference on Computer systems and technologies: e-Learning*, CompSysTech '03, pages 95–100, 2003. ISBN 954-9641-33-3. (Page 18).

[69] A. Grosskopf. An Extended Resource Information Layer for BPMN. Technical report, BPT, 2007. (Pages xvii, 33, 36, 39, 43, 47, 49, 50, 69, 76, 220, 221).

[70] G. Guizzardi. *Ontological Foundations for Structural Conceptual Models*. PhD thesis, 2005. (Page 32).

[71] T. Halpin and T. Morgan. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann, 2nd edition, 2008. ISBN 0-12-373568-8. (Page 217).

[72] M. Hammer and J. Champy. *Reengineering the corporation: a manifesto for business revolution*. HarperBusiness, New York, 1st ed. edition, 1993. ISBN 186448392 0887306403 088730687 186448392 1863737065 0887306403 1857880293 088730687. (Page 16).

[73] D. Harel and B. Rumpe. Meaningful modeling: what's the semantics of "semantics"? *Computer*, 37(10):64–72, 2004. ISSN 0018-9162. (Page 163).

[74] A. H. M. T. Hofstede and H. Proper. How to Formalize It? Formalization Principles for Information System Development Methods. *Information and Software Technology*, 40:519–540, 1998. (Page 120).

[75] Y.-N. Huang and M.-C. Shan. Policies in a Resource Manager of Workflow Systems: Modeling, Enforcement and Management. In M. Kitsuregawa, M. P. Papazoglou, and C. Pu, editors, *ICDE*, page 103. IEEE Computer Society, 1999. ISBN 0-7695-0071-4. (Page 230).

[76] U. Hustadt, R. A. Schmidt, and L. Georgieva. A Survey of Decidable First-Order Fragments and Description Logics. *Journal of Relational Methods in Computer Science*, 1:2004, 2004. (Page 120).

[77] T. Jaeger and J. E. Tidswell. Rebuttal to the NIST RBAC model proposal. In *Proceedings of the fifth ACM workshop on Role-based access control*, RBAC '00, pages 65–66, New York, NY, USA, 2000. ACM. ISBN 1-58113-259-X. (Page 207).

[78] A. Koschmider, L. Yingbo, and T. Schuster. Role Assignment in Business Process Models. In *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 37–49. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-28108-2. (Pages 30, 31, 33, 45, 47, 49, 50, 58, 76, 235, 236).

[79] V. Künzle and M. Reichert. Integrating Users in Object-Aware Process Management Systems: Issues and Challenges. In *Business Process Management Workshops*, pages 29–41. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12186-9. (Pages 55, 70, 78).

[80] M. La Rosa, M. Dumas, A. H. ter Hofstede, J. Mendling, and F. Gottschalk. Beyond Control-Flow: Extending Business Process Configuration to Resources and Objects. 2007. (Page 202).

[81] C. E. Landwehr, C. L. Heitmeyer, and J. Mclean. A Security Model for Military Message Systems. *ACM Transactions on Computer Systems*, 2:198–222, 1984. (Page 207).

[82] Y. A. Liu, S. D. Stoller, Y. A. Liu, and S. D. Stoller. Role-Based Access Control: A Corrected and Simplified Specification. In C. Wang, S. King, R. Wachter, R. Herklotz, C. Arney, G. Toth, D. Hislop, S. Heise, T. Combs, C. Wang, S. King, R. Wachter, R. Herklotz, C. Arney, G. Toth, D. Hislop, S. Heise, and T. Combs, editors, *Department of Defense Sponsored Information Security Research: New Methods for Protecting Against Cyber Threats*. Wiley, 2007. (Pages 41, 47, 49, 50, 76, 207).

[83] R. Lu, S. Sadiq, and G. Governatori. Compliance aware business process design. In A. H. M. ter Hofstede, B. Benatallah, and H.-Y. Paik, editors, *5th International Conference on Business Process Management (BPM 2007)*, pages 120–131. Springer, 2008. (Page 58).

[84] N. Luhmann. *System as Difference*, volume 13. Organisation, 2006. (Page 32).

[85] J. Mendling, M. L. Rosa, and A. H. ter Hofstede. Correctness of Business Process Models with Roles and Objects. 2008. (Page 55).

[86] A. Meyer. Resource Perspective in BPMN - Extending BPMN to Support Resource Management and Planning. Master's thesis, Hasso Plattner Institute, Potsdam (Germany), 2009. (Pages xvii, 32, 33, 34, 36, 39, 43, 44, 47, 49, 50, 68, 69, 76, 223, 224, 226).

[87] B. Motik and R. Rosati. Reconciling description logics and rules. *J. ACM*, 57: 30:1–30:62, June 2008. ISSN 0004-5411. (Page 120).

[88] B. Motik, P. F. Patel-Schneider, and B. C. Grau. OWL 2 Web Ontology Language Direct Semantics. http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/, 2009. (Pages 120, 182).

[89] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541 –580, apr 1989. ISSN 0018-9219. (Pages 5, 18, 58, 163, 164).

[90] D. Nardi and R. J. Brachman. An Introduction to Description Logics. In *Description Logic Handbook*, pages 1–40, 2003. (Pages 247, 248).

[91] M. Netjes, H. A. Reijers, and W. M. P. van Der Aalst. Supporting the BPM lifecycle with FileNet. In *CAiSE 2006*, pages 497–508, 2006. (Page 21).

[92] O. Nicolae and G. Wagner. Modeling and Simulating Organisations. In J. Barjis, T. Eldabi, and A. Gupta, editors, *EOMAS*, volume 88 of *Lecture Notes in Business Information Processing*, pages 45–62. Springer, 2011. ISBN 978-3-642-24174-1. (Page 32).

[93] A. Oberweis. A meta-model based approach to the description of resources and skills. In *AMCIS*, page 383, 2010. (Pages 31, 45, 235).

[94] OMG. BPMN 2.0. Recommendation, OMG, 2011. (Pages 5, 8, 17, 18, 19, 20, 21, 39, 41, 44, 47, 49, 50, 75, 76, 82, 86, 99, 102, 103, 104, 105, 113, 161, 163, 164, 194, 201, 225).

[95] X. F. Pan and G. Y. He. *Simulate your business process using IBM FileNet Process Simulator*. IBM Corporation, October 2008. (Page 56).

[96] Pellet. (OWL) 2 Reasoner for Java. http://clarkparsia.com/pellet, 2011. (Page 140).

[97] J. Peña, M. G. Hinchey, R. Sterritt, A. R. Cortés, and M. Resinas. A Model-Driven Architecture Approach for Modeling, Specifying and Deploying Policies in Autonomous and Autonomic Systems. In *DASC*, pages 19–30, 2006. (Page 7).

[98] (RACER). Renamed ABox and Concept Expression Reasoner. http://www.racer-systems.com, 2011. (Page 140).

[99] H. O. Reasoner. The new kid on the (OWL) Block. http://hermit-reasoner.com, 2011. (Pages 140, 184).

[100] T. Reuters. Journal Citation Reports, Thomson Reuters's Web site. http://thomsonreuters.com/products_services/science/science_products/a-z/journal_citation_reports/, Last accessed in August 2012. (Page 196).

[101] M. L. Rosa, M. Dumas, A. H. M. ter Hofstede, and J. Mendling. Configurable multi-perspective business process models. *Inf. Syst.*, 36(2):313–340, 2011. (Page 55).

[102] A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95, 2008. (Page 58).

[103] A. Ruiz-Cortés, O. Martín-Díaz, A. D. Toro, and M. Toro. Improving the Automatic Procurement of Web Services Using Constraint Programming. *Int. J. Cooperative Inf. Syst.*, 14(4):439–468, 2005. (Page 7).

[104] J. E. Rumbaugh, I. Jacobson, and G. Booch. *The unified modeling language reference manual*. Addison-Wesley-Longman, 1999. ISBN 978-0-201-30998-0. (Page 18).

[105] N. Russell, A. ter Hofstede, D. Edmond, and W. van der Aalst. Workflow Resource Patterns. Technical report, BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004. (Pages xvii, 8, 43, 46, 47, 49, 50, 76, 79, 82, 83, 110, 216, 217, 218).

[106] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In *CAiSE*, pages 216–232, 2005. (Pages 19, 31, 35, 69, 98, 218, 220, 223).

[107] S. Ryll. Querying the Data Perspective of Business Process Models. Master's thesis, HPI Potsdam, March 2009. (Page 59).

[108] K. Ryndina, J. Kuster, and H. Gall. Consistency of Business Process Models and Object Life Cycles. In *Models in Software Engineering*, pages 80–90. 2007. (Page 59).

[109] S. Sadiq, M. E. Orlowska, W. Sadiq, and C. Foulger. Data Flow and Validation in Workflow Modelling. In K. Schewe and H. E. Williams, editors, *Fifteenth Australasian Database Conference (ADC2004)*, volume 27 of *CRPIT*, pages 207–214, Dunedin, New Zealand, 2004. ACS. (Page 59).

[110] S. W. Sadiq, G. Governatori, and K. Namiri. Modeling Control Objectives for Business Process Compliance. In G. Alonso, P. Dadam, and M. Rosemann, editors, *BPM*, volume 4714 of *Lecture Notes in Computer Science*, pages 149–164. Springer, 2007. ISBN 978-3-540-75182-3. (Page 202).

[111] S. Sakr and A. Awad. A framework for querying graph-based business process models. In *WWW*, pages 1297–1300, 2010. (Page 22).

[112] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *Computer*, 29:38–47, February 1996. ISSN 0018-9162. (Page 207).

[113] A.-W. Scheer. *ARIS-Business Process Modeling*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 2000. ISBN 3540658351. (Pages 31, 44, 47, 49, 50, 54, 60, 62, 63, 76, 77, 78, 228).

[114] A.-W. Scheer, O. Thomas, and O. Adam. *Process Modeling using Event-Driven Process Chains*, pages 119–145. John Wiley and Sons, Inc., 2005. ISBN 9780471741442. (Pages 17, 21).

[115] S. Schefer, M. Strembeck, J. Mendling, and A. Baumgrass. Detecting and Resolving Conflicts of Mutual-Exclusion and Binding Constraints in a Business Process Context. In *OTM Conferences (CoopIS'12)*, pages 329–346, 2011. (Pages 58, 60, 61, 62, 63, 78, 241, 242).

[116] J. Searle. *The Construction of Social Reality*. Free Press, New York, first edition, 1995. (Page 32).

[117] M. Sierhuis, W. J. Clancey, and R. van Hoof. Brahms: A multiagent modeling environment for simulating work practice in organizations. *Int. J. of Simulation and Process Modelling*, pages 1–20, 2006. (Page 32).

[118] M. Smith. Role And Responsibility Charting (RACI). In *Project Management Forum (PMForum)*, page 5, 2005. (Pages 9, 33, 106, 237).

[119] *ARIS Process Governance*. Software AG, 2012. (Pages 33, 39, 44, 49, 50, 228, 243).

[120] M. Strembeck and J. Mendling. Modeling process-related RBAC models with extended UML activity models. *Inf. Softw. Technol.*, 53:456–483, May 2011. ISSN 0950-5849. (Pages 33, 40, 41, 42, 47, 49, 50, 58, 60, 68, 69, 76, 77, 78, 211, 240, 241, 242).

[121] M. Strembeck and G. Neumann. An integrated approach to engineer and enforce context constraints in RBAC environments. *ACM Trans. Inf. Syst. Secur.*, 7(3):392–427, 2004. ISSN 1094-9224. (Page 207).

[122] S. X. Sun, J. L. Zhao, J. F. Nunamaker, and O. R. L. Sheng. Formulating the Data-Flow Perspective for Business Process Management. *Info. Sys. Research*, 17(4): 374–391, 2006. (Page 59).

[123] A. Susi, A. Perini, J. Mylopoulos, and P. Giorgini. The Tropos Metamodel and its Use. *Informatica (Slovenia)*, 29(4):401–408, 2005. (Page 32).

[124] K. Tan, J. Crampton, and C. A. Gunter. The Consistency of Task-Based Authorization Constraints in Workflow Systems. In *Proceedings of the 17th IEEE workshop on Computer Security Foundations*, pages 155–169, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2169-X. (Pages 45, 47, 49, 50, 60, 61, 62, 63, 69, 76, 77, 78, 234, 244).

[125] P. Trinidad and A. Ruiz-Cortés. Abductive Reasoning and Automated Analysis of Feature Models: How are they connected? In *VaMoS*, pages 145–153, 2009. (Page 203).

[126] M. Uschold, M. King, S. Moralee, and Y. Zorgios. *The Enterprise Ontology*, volume 13. The Knowledge Engineering Review, 1998. (Pages 31, 44, 47, 49, 50, 76, 229).

[127] W. M. van der Aalst, A. H. ter Hofstede, and M. Weske. Business process Management: A survey. In *Business Process management*, volume 2678, pages 1–12. Springer, 2003. (Pages 4, 16, 21).

[128] W. M. P. van der Aalst. Verification of workflow nets. In *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer Berlin / Heidelberg, 1997. ISBN 978-3-540-63139-2. (Page 18).

[129] W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998. (Pages 18, 58, 142, 162, 163).

[130] W. M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011. ISBN 978-3-642-19344-6. (Pages 5, 23).

[131] W. M. P. van der Aalst and A. Kumar. A reference model for team-enabled workflow management systems. *Data Knowl. Eng.*, 38(3):335–363, 2001. (Pages 26, 32, 45, 47, 49, 50, 68, 69, 76, 232, 233).

[132] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: Yet Another Workflow Language. *Inf. Syst.*, 30(4):245–275, 2005. (Pages 18, 21, 30, 31, 43, 49, 50, 54, 58, 62, 76, 77, 218).

[133] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: a survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, Nov. 2003. ISSN 0169-023X. (Pages 23, 58).

[134] W. M. P. van der Aalst, M. Pesic, and M. Song. Beyond Process Mining: From the Past to Present and Future. In *CAiSE*, pages 38–52, 2010. (Page 23).

[135] I. Weber, G. Governatori, and J. Hoffmann. Approximate Compliance Checking for Annotated Process Models. Technical report, 2008. (Page 58).

[136] M. Weidlich, J. Mendling, and M. Weske. Efficient Consistency Measurement Based on Behavioral Profiles of Process Models. *IEEE Trans. Software Eng.*, 37(3): 410–429, 2011. (Pages 58, 79, 142, 148).

[137] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Verlag, 2012. (Pages 4, 5, 15, 16, 17, 19, 21, 24, 26).

[138] B. Wetzstein, Z. Ma, A. Filipowska, M. Kaczmarek, S. Bhiri, S. Losada, J.-M. Lopez-Cob, and L. Cicurel. Semantic Business Process Management: A Lifecycle Based Requirements Analysis. In *SBPM*, 2007. (Page 21).

[139] C. Wolter and A. Schaad. Modeling of Task-Based Authorization Constraints in BPMN. In G. Alonso, P. Dadam, and M. Rosemann, editors, *Business Process Management*, volume 4714 of *Lecture Notes in Computer Science*, pages 64–79. Springer Berlin/Heidelberg, 2007. (Pages xvii, 33, 36, 39, 43, 47, 49, 50, 76, 221, 222, 223).

[140] M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia Methodology For Agent-Oriented Analysis And Design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3:285–312, 2000. (Page 32).

[141] H. Yang, C. Wang, Y. Liu, and J. Wang. An Optimal Approach for Workflow Staff Assignment Based on Hidden Markov Models. In *OTM Workshops*, pages 24–26, 2008. (Page 58).

[142] J. Yu, B. Benatallah, F. Casati, and F. Daniel. Understanding Mashup Development. *IEEE Internet Computing*, 12:44–52, 2008. (Page 186).