Modeling and mining business process variants in cloud environments

Karn Yongsiriwit

université PARIS-SACLAY

TELECOM SudParis

DOCTORAT EN CO-ACCREDITATION
TELECOM SUDPARIS ET L'UNIVERSITE PARIS-SACLAY

Spécialité: Informatique

Ecole doctorale: Sciences et Ingénierie

Présenté par Karn Yongsiriwit

Pour obtenir le grade de DOCTEUR DE TELECOM SUDPARIS

MODELING AND MINING BUSINESS PROCESS VARIANTS IN CLOUD ENVIRONMENTS

Soutenue le 23/01/2017

devant le jury composé de :

Directeurs de thèse :

M.	Walid Gaaloul	Professeur, TELECOM SudParis, France

Co-encadrant de thèse :

M.	Mohamed Sellami	Maître de conférences, ISEP, France

Rapporteurs :

M.	François Charoy	Professeur, University of Lorraine, France
M.	Pascal Poizat	Professeur, Université Paris Ouest, France

Examinateurs :

M.	Khalil Drira	Professeur, Université de Toulouse, France
M.	Djamal Benslimane	Professeur, Université Claude Bernard Lyon 1, France
Mme.	Anne Doucet	Professeur, LIP6, France

Thèse numéro : 2017SACLL002

to my family

Acknowledgment

First and foremost, I would like to express my appreciation and gratitude to my advisor, Dr. Walid Gaaloul, who offered abundantly helpful assistance, support and guidance. His vision, creativeness and enthusiasm inspired me greatly to work. This dissertation would not have been possible without his help.

I am sincerely and heartily grateful to my co-director, Dr. Mohamed Sellami, who gave me a lot of valuable advice. His expertise and experiences influenced and helped me be more mature in doing research.

I acknowledge the TELECOM SudParis institute for offering me a scholarship and providing me good environment and facilities to complete my thesis.

I am obliged to my colleagues, Emna Hachicha, Nour Assy, who made many useful discussions in research.

Finally, I am forever indebted to my beloved parents for their understanding, endless patience and encouragement. They have given me constant supports, needed inspiration and always wish the best things to me. I dedicate this thesis to them.

Abstract

More and more organizations are adopting cloud-based Process-Aware Information Systems (PAIS) to manage and execute processes in the cloud as an environment to optimally share and deploy their applications. This is especially true for large organizations having branches operating in different regions with a considerable amount of similar processes. Such organizations need to support many variants of the same process due to their branches' local culture, regulations, etc. However, developing new process variant from scratch is error-prone and time consuming. Motivated by the "Design by Reuse" paradigm, branches may collaborate to develop new process variants by learning from their similar processes. These processes are often heterogeneous which prevents an easy and dynamic interoperability between different branches.

A process variant is an adjustment of a process model in order to flexibly adapt to specific needs. Many researches in both academics and industry are aiming to facilitate the design of process variants. Several approaches have been developed to assist process designers by searching for similar business process models or using reference models. However, these approaches are cumbersome, time-consuming and error-prone. Likewise, such approaches recommend entire process models which are not handy for process designers who need to adjust a specific part of a process model. In fact, process designers can better develop process variants having an approach that recommends a well-selected set of activities from a process model, referred to as process fragment. Large organizations with multiple branches execute BP variants in the cloud as environment to optimally deploy and share common resources. However, these cloud resources may be described using different cloud resources description standards which prevent the interoperability between different branches.

In this thesis, we address the above shortcomings by proposing an ontology-based approach to semantically populate a common knowledge base of processes and cloud resources and thus enable interoperability between organization's branches. We construct our knowledge base built by extending existing ontologies. We thereafter propose an approach to mine such knowledge base to assist the development of BP variants. Furthermore, we adopt a genetic algorithm to optimally allocate cloud resources to BPs. To validate our approach, we develop two proof of concepts and perform experiments on real datasets. Experimental results show that our approach is feasible and accurate in real use-cases.

Keywords. Business process models, process mining, cloud resource allocation, business process fragments, ontology, knowledge base.

Abstract

De plus en plus les organisations adoptent les systèmes d'informations sensibles aux processus basés sur Cloud en tant qu'un environnement pour gérer et exécuter des processus dans le Cloud dans l'objectif de partager et de déployer leurs applications de manière optimale. Cela est particulièrement vrai pour les grandes organisations ayant des succursales opérant dans des différentes régions avec des processus considérablement similaires. Telles organisations doivent soutenir de nombreuses variantes du même processus en raison de la culture locale de leurs succursales, de leurs règlements, etc. Cependant, le développement d'une nouvelle variante de processus à partir de zéro est sujet à l'erreur et peut prendre beaucoup du temps. Motivés par le paradigme <<la conception par la réutilisation>>, les succursales peuvent collaborer pour développer de nouvelles variantes de processus en apprenant de leurs processus similaires. Ces processus sont souvent hétérogènes, ce qui empêche une interopérabilité facile et dynamique entre les différentes succursales.

Une variante de processus est un ajustement d'un modèle de processus afin de s'adapter d'une façon flexible aux besoins spécifiques. De nombreuses recherches dans les universités et les industries visent à faciliter la conception des variantes de processus. Plusieurs approches ont été développées pour aider les concepteurs de processus en recherchant des modèles de processus métier similaires ou en utilisant des modèles de référence. Cependant, ces approches sont lourdes, longues et sujettes à des erreurs. De même, telles approches recommandent des modèles de processus pas pratiques pour les concepteurs de processus qui ont besoin d'ajuster une partie spécifique d'un modèle de processus. En fait, les concepteurs de processus peuvent mieux développer des variantes de processus ayant une approche qui recommande un ensemble bien défini d'activités à partir d'un modèle de processus défini comme un fragment de processus. Les grandes organisations multi-sites exécutent les variantes de processus BP dans l'environnement Cloud pour optimiser le déploiement et partager les ressources communes. Cependant, ces ressources Cloud peuvent être décrites en utilisant des différents standards de description des ressources Cloud ce qui empêche l'interopérabilité entre les différentes succursales.

Dans cette thèse, nous abordons les limites citées ci-dessus en proposant une approche basée sur les ontologies pour peupler sémantiquement une base de connaissance commune de processus et de ressources Cloud, ce qui permet une interopérabilité entre les succursales de l'organisation. Nous construisons notre base de connaissance en étendant les ontologies existantes. Ensuite, nous proposons une approche pour exploiter cette base de connaissances afin de supporter le développement des variantes BP. De plus, nous adoptons un algorithme génétique pour allouer d'une manière optimale les ressources Cloud aux BPs. Pour valider notre approche, nous développons deux preuves de concepts et effectuons des expériences sur des ensembles de données réels. Les résultats expérimentaux montrent que notre approche est réalisable et

précise dans des cas d'utilisation réels.

Mot clés. Modèle de processus métier, fouille du processus, allocation des ressources Cloud, fragment du processus métier, ontologie, base de connaissances.

Table of contents

List of Tables

List of Figures

Chapter 1

Introduction

Contents

1.1 Research context

In highly competitive business environments, organizations are forced to effectively utilize Information Technology (IT) to achieve excellent performance of their business processes management and execution [10,11]. Organizations are increasingly adopting software systems, so called Process-Aware Information Systems (PAISs). Such systems manage and execute operational processes on the basis of business process (BP) models involving multiple people, applications, and/or information sources [12, 13].

Examples of PAISs are Business Process Management (BPM) systems [14–18] and Enterprise Resource Planning (ERP) systems [19].

PAISs explicitly describe a BP model to represent the logical and the temporal order in which organizational tasks have to be performed to achieve a specific goal [20]. Business process management (BPM) is a field that focuses on improving performance by managing and optimizing BPs [21]. Traditionally, BPM life-cycle can be categorized into four repeated steps to continuously improve BP models. It consists of: (1) process design, (2) process implementation, (3) process execution, and (4) process diagnosis. In the process design phase, a BP is modeled using graphical notations proposed in the literature, such as Event-Driven Process Chain (EPC) [22], Business Process Model and Notation (BPMN) [23], UML Activity Diagram [24], Petri nets [25], Yet Another Workflow Language (YAWL) [22], etc. The designed BP model is thereafter implemented and executed by a PAIS. During the BP execution, the underlying PAIS records and accumulates historical data of the process execution (i.e., process event logs) [26]. Such data is then analyzed in the diagnosis phase to identify possible problems and to improve the designed BP model using process mining techniques [27–30].

Many large organizations with multiple branches situated in different regions are adopting PAIS to support many variants of the same BP due to branches' local culture, regulations, etc [31]. For example, considering a car rental company having branches located in different regions, each branch needs to develop variants of their reservation process to comply with its local regulation and culture. Hence, BPs are shared among different branches and can be flexibly adjusted and executed to branches' specific needs [31].

Cloud computing [32] has rapidly grown for delivering shared ICT services and resources over the internet. According to the National Institute of Standards and Technology, cloud computing is a model that enables providers sharing their computing resources (e.g., networks, servers, storage, applications, and services) and users can access them in ubiquitous, convenient and on-demand way with a minimal management effort [33]. A recent study from Gartner states that PAISs on cloud environments is the biggest segment with 77% of the market and predicts a growth from $\$84.1B$ in 2012 to $\$144.7B$ in 2016, generating a global compound annual growth rate of 15% [34]. Such organization can provision their BP variants in cloud environments to adopt agile, flexible and cost-effective business solutions and also to reduce BP development and maintenance costs.

In fact, more and more organizations are adopting PAIS on cloud environments to manage their considerable amount of BP variants. Thus, a large amount of cloud resources are being consumed by these BP variants. Hence, within an organization, its branches are willing to share their accumulated cloud resources and useful process data (i.e., BP models and process event logs) in order to assist the design of BP variants.

In fact, in a highly competitive environment organizations need to flexibly ad-

just their existing BP models to develop new BP variants in order to meet emerging business needs. BPM life-cycle allows organizations to significantly reduce BP development time and cost savings [35]. The design phase in the BPM life-cycle is the initial and key phase of a BP development [14]. Many approaches have been proposed to facilitate the BP design phase using BP templates [36], reference BPs [37], and measuring the similarity between BPs [38–41]. Existing approaches, however, mainly study conceptual BP models and take into account the entire BP topology which is labor-intensive and often the cause for error-prone and time consuming BP variant development [42]. Moreover, they barely consider BP variant deployment in cloud environments and more specifically cloud resource allocation.

Our thesis focuses on the development of process variants in the process design phase of the traditional BPM lifecycle (Figure 1.1) and the allocation of cloud resources consumed by BPs. To do so, we exploit process data (i.e., collection of BP models and process event logs), and cloud resources shared by an organization's branches to assist respectively the design and deployment of process variants in cloud environment by: (i) facilitating process variant design using heterogeneous process data (i.e., BP models and process event logs), and (ii) solving optimal allocation

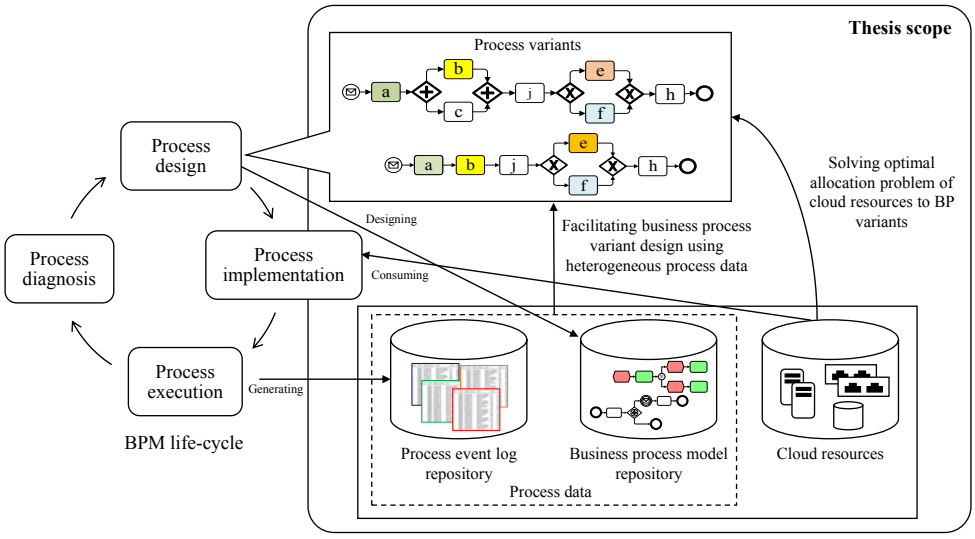


Figure 1.1: BP variant development in the BPM lifecycle inspired from [1]

1.2 Research problem: *How to support business process variant design by exploiting process data*

1.2.1 On assisting business process variant design by exploiting process data

Motivated by the "Design by Reuse" paradigm, many approaches have proposed to take into consideration previous design experience, best practices and how other organization's branches perform similar BPs in order to assist business process variant design.

Several approaches aim at supporting BP design by providing best-practice reference BP models [43, 44] or measuring BP similarity. For example in [38], Dijkman et al. presented and compared four algorithms for calculating the similarity of BP models based on graph matching. Dongen et al. [39] represented behaviors, so-called *causal footprints (e.g. task a is always succeeded by b, but that b can also occur before a)* [45], of BP models to compute similarity between them. In [40], Kunze et al. proposed an indexing approach to search for similar BP models based on metric trees and edit distances. Yan et al. [41] presented an algorithm for classifying similar BP models based on their features, e.g., tasks and task succession. These approaches however do not consider process event logs which reflect closely the behavior of BPs and always exist in information systems [46].

In some circumstances, process designer may need recommendations for some parts of an under design BP, called BP fragments, instead of entire BPs. For example, we consider a process designer designing a BP model as shown in Figure 1.2. The process designer is looking for activities and their relations (i.e., BP fragments) that are suitable for the selected position (with the '?' mark) in the under-design process. In this case, recommending an entire BP will not be helpful. Instead, BP fragment recommendation is more suitable and straightforward. Recent works mainly recommend entire BP which is labor-intensive and often the cause for an error-prone and time consuming BP variant design [42].

In light of these limitations, our objective is to facilitate the design of BP variants without confusing the process designer with large and complex results. We aim at recommending BP fragments that are relevant to selected positions of an under-design BP. To ensure this objective, we propose to reuse heterogeneous process data-sources providing as shared knowledge base of BP models (Section 1.2.2) and process event logs (Section 1.2.3), to recommend focused and comprehensible results (see Figure 1.2). To address this research problem, we need to answer the following question:

- **R1**: How to identify BP fragments that are close to process designer interests from heterogeneous process data?

These recommended BP fragments can be used in three typical cases:

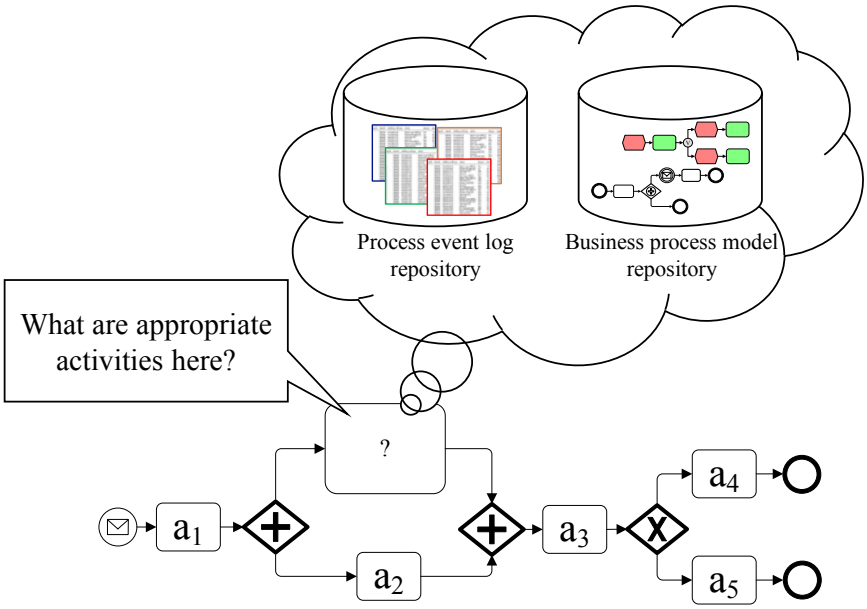


Figure 1.2: Our BP variant development problematic

- when a suitable BP fragment is needed to add for an empty place in an under design BP model;
- when a BP model is needed to be adopted as a new variant to meet new business goals;
- when the execution of a BP fragment fails, and thus needs to be replaced with another BP fragment having similar behavior.

A serious challenge that prevents fostering the use of process data (i.e., BP models and process event logs) between organization's branches is that of *interoperability* [47]. Interoperability is defined as the ability of ICT systems to exchange information and to use the information that has been exchanged [48,49]. Figure 1.3 depicts our problems in which process data (i.e., BP models and process event logs) are heterogeneous and could not be easily interoperated between an organization's branches. Therefore, process data should be semantically modeled in a machine-interpretable way, such that it enables interoperability between an organization's branches on BP model level (Section 1.2.2) and process event log level (Section 1.2.3).

1.2.2 On modeling heterogeneous business process models in a common knowledge base

The variety of BP modeling languages (e.g. EPC [22], BPMN [23], UML [24], Petri nets [25], YAWL [22], etc.) along with model elements' labels usually written in a

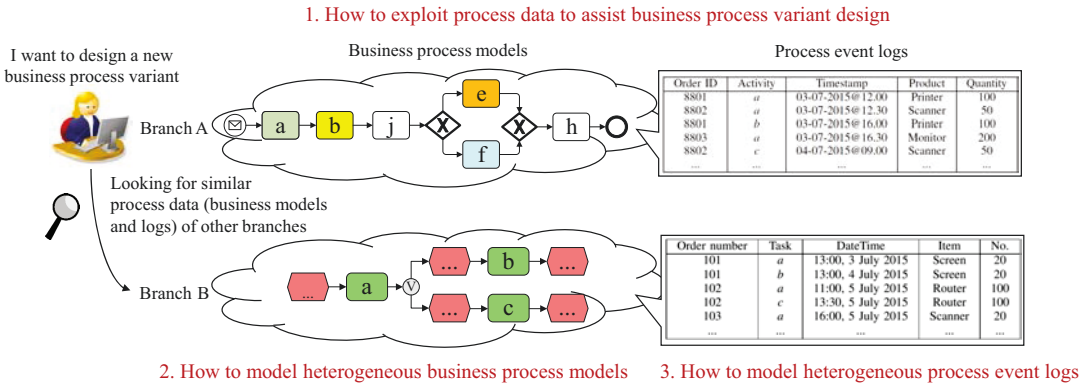


Figure 1.3: Process data heterogeneity

natural language have mostly implied semantic heterogeneity in BP models. In fact, an organization's branches usually define their own BP models with their preferred modeling language and element's labels as they like. This makes BP models diverse, heterogeneous and thus difficult to exploit by other branches without making a proper data transformation.

Ontology engineering is one of the semantic web fields that allows to model the knowledge in organizations. It offers the means for solving semantic heterogeneity problems [50]. Many approaches have been proposed to use ontologies to model BP models described in a certain process modeling language. Concretely, they use ontologies as means to conceptualize and represent BP models in EPC [51–54] and BPMN [55]. These approaches however focus only on certain BP modeling languages. They do not take into account the heterogeneity problems of BP models. Thus, scenarios where an organization's branches describe their BP models using different languages are not supported by these works.

On the other hand, several approaches have proposed generic ontologies to represent BP models. In [56], Cabral et al. have presented the Business Process Modeling Ontology (*BPMO*) as a part of the approach developed by the SUPER European project [57] to construct generic BP models at the semantic level. In [6], Lin et al. proposed to semantically annotate heterogeneous process models to ensure semantic interoperability and to manage process knowledge from different enterprises. They formalized a new process model representation, namely *PSAM* (Process Semantic Annotation Model) to provide links from process models to domain-specific ontologies. In [58], Thomas et al. have presented a semantic language-independent process modeling as an extension of semi-formal process modeling languages using ontologies. Among these works, *BPMO* is the most promising because of its capability to represent BP models in various modeling languages. However, the automatic translate back and forth between heterogeneous BPs to BPMO is still semi-manual.

In our work, we aim at modeling heterogeneous BP models to enable interoper-

ability when exploiting process data. To this end, we aim at representing these BP data using appropriate ontologies. Thereafter, we want to automate the construction of a shared knowledge base which can be used as a data-source to develop BP variants. To address this research problem, we need to answer the following question:

- **R2**: How to model heterogeneous BP models to be shared in a common knowledge base?

1.2.3 On modeling heterogeneous process event logs in a common knowledge base

Typical process-aware information systems (e.g., ERP, CRM, and workflow management systems) always produce and accumulate process event logs. Process event logs contain information on when tasks were executed and by whom [27, 28], which is essential for identifying important business execution paths.

Organization's branches using PAIS execute their heterogeneous BP models and accumulate heterogeneous process event logs which are not easy to exploit by other branches. Several efforts have been made to conceptualize events as ontologies [59, 60]. However, proposed ontologies represent events for general use. Thus they did not describe relationships between events and BPs and hence are not suitable for representing process event logs. In [61], authors defined the Event Ontology (*EVO*) to model events taking place during BP executions. This work is a part of SUPER European project [57], therefore events described using *EVO* correspond to tasks described using *BPMO*. However, authors do not specifically describe how to model heterogeneous process event logs with EVO.

Process mining techniques [28] allow BP discovery using process event logs, BP conformance checking (i.e., comparing a BP model with its process event logs to check if the actual execution conforms to the model or not) and BP enhancement (i.e., using process mining results to improve BP models). By using such techniques [62], we can discover a BP model from process event logs as graphs. Such graphs can be represented as log-based BP graph [29], Causal Net (C-net) [63], etc. However, only C-nets allow modeling XOR, AND and OR logical connectors and hence fit well with various BP modeling languages [28].

Our objective is to model heterogeneous process event logs such that they can be easily used by different organization's branches. We aim at representing process event logs using appropriate ontologies. Thereafter, we want to populate a shared knowledge base with this data as mentioned in Section 1.2.2 which can be later used as a data-source to apply process mining techniques assisting BP variant design. To address this research problem, we need to answer the following question:

- **R3**: How to model heterogeneous process event logs to be shared in a common knowledge base?

1.3 Research problem: *How to optimally allocate heterogeneous cloud resource to business processes*

1.3.1 On modeling heterogeneous cloud resources in a common knowledge base

Interoperability between cloud resources is one of the most substantial challenges in order to effectively share and reuse resources within an organization. As Cloud Computing is relatively a new paradigm, the interoperability issue is being increasingly addressed at different level of IaaS [64], PaaS [65–68] and SaaS [69, 70]. Many cloud resource management standards have also been proposed to cope with this issue by providing description models for cloud resources. Such models describe cloud resources and thus enable interoperability between them. Among the most known stardards, we cite TOSCA [71] (Topology and Orchestration Specification for cloud Applications), OCCI [72] (Open Cloud Computing Interface), and CIMI [73] (Cloud Infrastructure Management Interface).

Organizations' branches are describing their cloud resources by different cloud resource description standards. Such standards are being developed in isolation. For example, TOSCA is developed by the OASIS Technical Committee, OCCI by the Open Grid Forum community, and CIMI by the DMTF Cloud Management Working Group. Hence, cloud resources described using these different standards might not be simply interoperable because of their heterogeneous schema and vocabulary. In a collaborative environment where organizations are willing to share their cloud resources, translating cloud resource descriptions is usually done manually and is therefore a tedious task. This in turn creates cloud silos and non-reusable cloud resources.

With the objective of enabling *interoperability* between cloud resource description standards, we aim at representing cloud resources using appropriate ontologies. Hence, we want to populate a knowledge base with this data allowing the sharing of cloud resources within an organization. To address this research problem, we need to answer the following question:

- **R4**: How to model heterogeneous cloud resources to be shared in a common knowledge base?

1.3.2 On solving optimal allocation of heterogeneous cloud resources to business processes

The tremendous emergence of cloud computing has generated several challenges in order to efficiently provision cloud resources [74]. This is especially true for large organizations having large amount of BP variants executed in cloud environments. A lot of cloud resources are being consumed by BPs. Hence, optimal allocation of cloud resources to BPs is needed for organizations to optimize their costs. However, cloud resource allocation to BPs is a combinatorial problem with large-scale complexity.

Such problem consists of how to select the most suitable resources required by different BP activities to ensure an optimal resource consumption in term of quality of service (QoS).

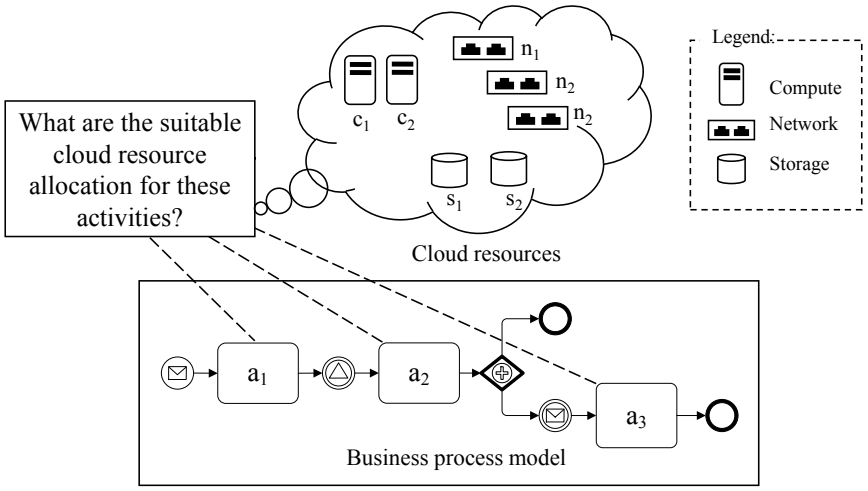


Figure 1.4: Cloud resource allocation to BP problematic

Our objective is to solve the problem of optimal cloud resource allocation to BPs. We aim at proposing a genetic-based approach, which is adequate for combinatorial problems, for selecting an optimal cloud resource allocation using a heterogeneous cloud resource data-source, i.e., a shared knowledge base of cloud resources (Section 1.3.1). To address this research problem, we need to answer the following question:

- **R5**: How to solve optimal allocation of cloud resources to business processes?

1.4 Thesis principles, objectives and contributions

1.4.1 Thesis principles

In this thesis we consider the following principles while developing our approach:

- **Heterogeneous data modeling**: The approach should model heterogeneous process data and cloud resources in a machine-interpretable way, such that it enables interoperability among an organization.
- **Automation**: The approach should propose automated techniques.

- **Implicit knowledge exploitation**: The approach should be driven by the "Design by Reuse" paradigm. Consequently, it should extract and utilize implicit knowledge hidden in existing and accessible cloud-based PAIS data such as already designed BP models, process event logs and cloud resources.

- **Focused results**: To not confuse process designers, the approach should recommend focused results that are close to their interest.

It is noteworthy that the proposed work in this thesis needs to be (*i*) validated through proof of concepts and (*ii*) evaluated through different experiments on real datasets. Therefore, the implementation, experiments, and case study results with end users should be detailed.

1.4.2 Thesis objectives

In this thesis, we aim at **proposing automated support for BP variant design by exploiting existing process data** and **proposing automated support for optimal allocation of cloud resources to business process variants**. Our first goal is twofold: (*i*) model and mine heterogeneous process data by constructing a shared knowledge base and (*ii*) utilize such knowledge base to assist the design of BP variants by proposing fine-grained results that are close to process providers' interests. The second goal is also twofold: (*iii*) model heterogeneous cloud resources to construct a shared knowledge base, and (*iv*) optimally allocate cloud resource to BP variants.

To achieve the first and third objective, we propose a semantic framework tackling this heterogeneity issues. This framework promotes the creation of a semantic knowledge base from heterogeneous process data and cloud resources shared by an organization's branches. Our framework promotes the interoperability and thus enables the reuse of such data within an organization.

For the second objective, we propose to learn from past process data in our knowledge base to assist BP variant design by recommending process fragments. The recommended process fragments are close to process designers' interests and inspire them to flexibly adjust specific parts in their under-design BPs.

To achieve the fourth objective, we allocate cloud resources to BP variants using cloud resource descriptions stored in our knowledge base. Furthermore, we propose a novel approach using a genetic algorithm to solve such combinatorial problem with large-scale complexity.

1.4.3 Thesis contributions

We propose an ontology-based approach to semantically populate a knowledge base with heterogeneous process data (i.e., BP models and process event logs). This knowledge base is developed upon a set of ontologies that we define to represent this data.

Our proposed ontologies extend ontologies developed in the context of the European project SUPER [57] which aims to enable interoperability between enterprises' busi-

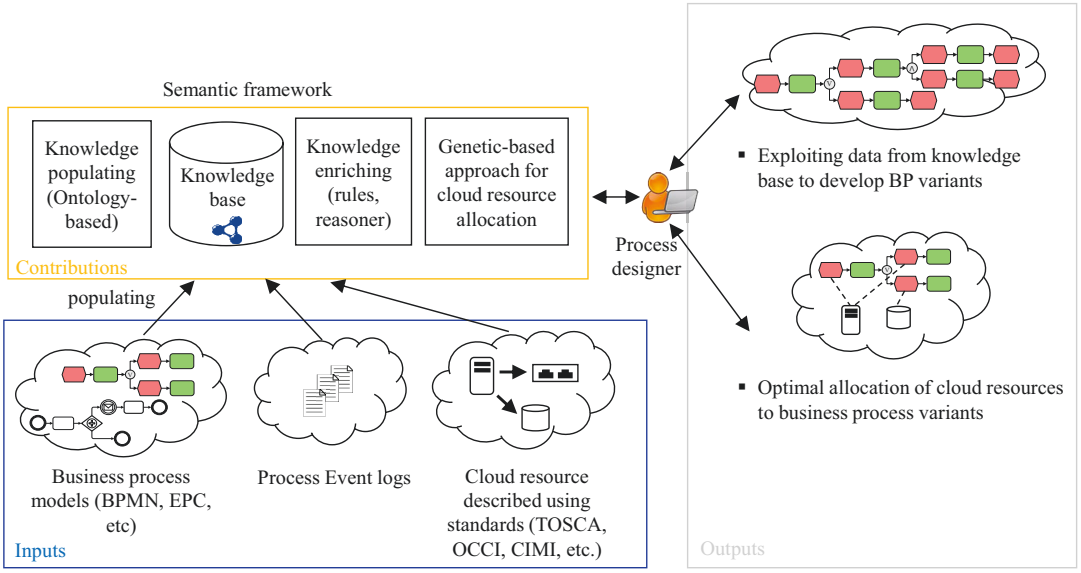


Figure 1.5: Thesis contribution

To promote a flexible adjustment of BPs in the process design phase, we aim to exploit process data (i.e., BP models and process event logs) to assist BP variant design. To do so, we propose to populate a common knowledge base using heterogeneous process data as input (Figure 1.5). Thereafter, we mine process data stored in our knowledge base to recommend suitable mined process fragments to selected positions of an under-design BP.

Considerable amount of cloud resources will be consumed by BP variants. Therefore, we propose to populate a knowledge base with the descriptions of cloud resources consumed by BPs with the aim at enabling interoperability and sharing such resources within an organization. Furthermore, we propose to adopt genetic algorithm to deal with the optimal cloud resource allocation to BP variants.

We validate our work in two steps. Firstly, we developed two proof-of-concepts: (*i*) a BP variant modeling application, and (*ii*) a cloud resource knowledge base application. We performed experiments on modeling and mining BP models for recommendation on two large datasets: (*i*) SAP reference models [75], (*ii*) and IBM BP models proposed in [76]. We also conducted experiments on synthesized process event logs generated from SAP reference models. We then evaluated the *feasibility*, *efficiency* and *accuracy* of our proposed approaches. Regarding modeling heterogeneous cloud resources, we applied our approaches on use cases identified from different standards.

In summary, our contributions in this thesis are as follows:

1. **An automated approach to model heterogeneous process data and cloud resources in a shared knowledge base:**
   - An ontology-based approach to represent heterogeneous process data (i.e., BP models and process event logs) and heterogeneous cloud resources in a shared knowledge base.
2. **An automated approach to assist the design of process variants by mining our shared knowledge base:**
   - *A data mining based approach* to extract process fragments from process data in our shared knowledge base;
   - An algorithm that *measures similarities betweens* process fragments for recommendation.
3. **An automated approach to optimally allocate cloud resources to business process variants:**
   - *A genetic-based algorithm* that *select suitable cloud resource allocation to BPs.*
4. **Validation:**
   - *Two proof-of-concepts* for each contribution implemented as extensions of the Signavio process editor;
   - *Experiments on two large datasets from SAP reference BP models and IBM BP models* to demonstrate the feasibility, efficiency and accuracy of our proposed assisting BP variant design approach;
   - *Experiments on use-cases* to compare our genetic-based resource allocation approach with other approaches;

1.5 Thesis outline

This thesis is organized as follows: Chapter 2 presents a background on our research context. It starts by presenting existing works related to modeling heterogeneous BP models, process event logs, and cloud resource descriptions. Thereafter, we present different approaches for supporting the design of BP variants. We review their models and analyze their solutions. This analysis allows us to justify the need for **proposing an automated support for BP variant design by exploiting existing process data** and **proposing an automated support for optimal allocation of cloud resources to business processes**.

Chapters 3, 4 and 5 are the core of our thesis which elaborate our approach to support the design of BP variants by exploiting heterogeneous process data (i.e., BP models and process event logs), and automated support for interoperating heterogeneous cloud resources.

In Chapter 3, we present our approach to model heterogeneous BP models in a common knowledge base and support the design of BP variants using such knowledge base. We reuse an existing ontology, namely *BPMO*, to model heterogeneous BP models in our knowledge base. We propose the *NCFO* ontology as an extension of *BPMO* to represent fragments of BP models. Then, we present our approach for matching and recommending similar BP fragments to assist the design of BP variants.

In Chapter 4, we present our automated approach for supporting the design of BP variants by exploiting process event logs. We propose the *Linked-CN* ontology as an extension of an existing ontology, namely *EVO*. We present our approach to discover BP models from process event logs and to populate a knowledge base of discovered BP models using *Linked-CN* ontology. We extend the *NCFO* ontology (Chapter 3) to represent fragments of discovered BP models. Thereafter, we present our matching algorithm to recommend similar BP fragment which can be used as means for BP variant design.

In Chapter 5, we present an approach for modeling heterogeneous cloud resource described by different standards. We propose an ontology, namely *Linked-CR*, which defines high level concepts to describe cloud resources modeled with different standards (e.g., TOSCA, OCCI, and CIMI). We populate a common knowledge base of cloud resources using *Linked-CR*. We use semantic inference rules to enable an automated standard translation. Cloud resource descriptions modeled with *Linked-CR* allows organizations to query and customize cloud resources regardless of their representation by cloud providers.

In Chapter 6, we present an approach to optimal allocate of cloud resources to BP variants using genetic algorithm. We encode our problem to a genome then apply genetic algorithm to find the most optimal cloud resource allocation.

In chapter 7, we present the proof of concepts that we implemented, the experiments that we performed and the case study that we conducted to validate our approach.

Finally, Chapter 8 concludes this thesis by summarizing the presented work and discussing possible extensions.

Chapter 2

Related Work

Contents

2.1 Introduction

In this chapter, we review the existing works in the literature relevant to our research problems which are *How to support business process variant design by exploiting process data?* (Section 1.2) *and How to optimally allocate heterogeneous cloud resource to business process variants* (Section 1.3). Firstly, we study existing solutions for facilitating the design of BP variants in Section 2.2. Thereafter, we focus on existing ontology-based approaches for modeling heterogeneous process data (BP models and process event logs) in Section 2.3 and 2.4, respectively. In Section 2.5, we focus on existing ontology-based approaches for modeling heterogeneous cloud resources Section 2.5. Then, we study existing works for cloud resource allocation to BPs (Section 2.6). Finally, we conclude the chapter in Section 2.7.

2.2 Assisting business process variant design

In this section, we review existing approaches that aim at assisting BP variant design. We categorize these approaches into four categories: business process modeling (Section 2.2.1), business process similarity (Section 2.2.2), business process querying (Section 2.2.3), and business process mining (Section 2.2.4).

2.2.1 Business process modeling

Business process modeling refers to techniques that represent BPs by means of appropriate graphical notation [15] in order to filter out the complexity of the real world, and thus the important parts of the system can be focused [77]. Over the past few decades, there has been a large amount of effort to propose process modeling languages to represent BPs, such as Event-Driven Process Chain (EPC) [22], Business Process Model and Notation (BPMN) [23], UML Activity Diagram [24], Petri nets [25], Yet Another Workflow Language (YAWL) [22], etc. Moreover, many tools have been developed to provide the ease of designing BPs with graphical interface, such as Visio, Workflow designer, Process marker, Tibco, Holosofx, Questetra, Bizagi, Bonita, etc. Despite the considerable number of techniques and tools, BP model design is still time-consuming and labor-intensive [78].

In today's rapidly changing business environments, organizations need to flexibly adjust their BPs to quickly adapt to new requirements (i.e., BP variant design) [79,80]. Re-designing or adjusting existing BP without considering similar BP from other community is not effective [81]. Thus, several associations and vendors have defined models, so-called reference models, to capture proven practices and recurring business operations (i.e., common business activities) in a specific application domain. Reference models are generic model to be individualized to fit specific requirements for organizations or IT projects. Several commercial process modeling tools integrate reference models as standardized libraries in order to reuse the proven practices across

process (re-)design projects. They do so by capturing knowledge about common activities, information artifacts and flows encountered in specific application domains. Examples of such tools are the SAP reference models [82] and the Supply Chain Operations Reference (SCOR) models [83].

Reference model has shortcomings when given to BP designers as a reference to design BP variants because (*i*) it is provided manually, thus the design of BP variants from reference model is error-prone and time-consuming and (*ii*) reference model is given as an entire model while sometimes only some parts of the model need to be considered.

2.2.2 Business process similarity

Designing BP variants start by finding similar BP models by measuring the similarity between them. This helps identifying the overlapping and redundant parts between BP models in a BP repository. Many approaches have been proposed to measure such similarity [2, 84–89].

Several existing approaches compute this similarity metric based solely on conceptual BP models. For example, Dijkman et al. [2] have proposed to rank BP models in a repository based on their similarity to a given BP model. Concretely, the authors represent a BP model as a graph (Figure 2.1). Thus, perform matching between graphs by establishing 1-to-1 correspondences between nodes in the compared graphs. Bunke [84] has proposed to represent BP model as a directed attributed graph and then uses graph-edit distance to compute the similarity between BP models. Concretely, the number of deletions/insertions/substitutions of nodes and edges in order to transfer from one model to another is measured as a similarity metric. Yan et. al. [85] compute the similarity between two BP models by focusing on the characteristics of model elements, i.e., activity label and connection elements (e.g., start, end, sequence, split and join). The similarity between two activity labels is computed using the text edit distance [90] while the similarity between two connection element is computed by the average number of input and output paths. Finally, the similarity between two BP models is synthesized from the similarity of their activity labels and connection elements.

Li et. al. [86] have proposed to measure the difference between two BP models based on the number of operations that need to be performed to transform one process to the other. Concretely, they considered execution orders between activities and measured the difference based on the deletions/insertions/movements of activities. Ehrig et. al [87] represent Petri-net BP models by an ontology with a set of properties. The similarity between model elements is computed based on the syntactic and semantic matching of their properties. They use string-edit distance for the syntactic matching and WordNet library for the semantic matching.

Some approaches [88, 89] measure similarity by exploiting process event logs. For example, Aalst et. al. [88] have proposed to compute the similarity value of two BP

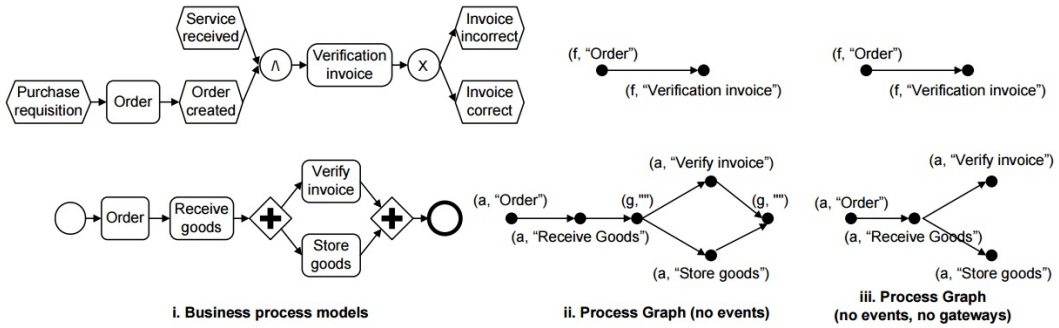


Figure 2.1: Two processes and their graph representations [2]

models based on their process execution logs (i.e., process event logs). BP models are modeled using Petri-Net, while process event logs are modeled as sequence of activities, namely log traces. They defined the fitness notion between two process events based on the enable state (e.g., int, ready, running, terminated, skipped) of activities in the traces. The similarity between two BP models is evaluated based on their respective precision and recall metrics. Such metrics are computed based on the fitness between the log traces of the two BP models. Dongen et. al. [89] measure the similarity between two EPC BP models using vector space model (VSM), i.e. the cosine value of the angle created by the two corresponding vectors. They defined the 'causal footprint' as a vector to represent the execution orders of activities. Finally, the similarity between two causal footprints is computed using VSM which represents the similarity of their corresponding BP models.

Measuring similarity between two entire BPs often leads to the graph-matching problem, which is NP-complete [91], and might need to deal with the trade-off among the complexity, accuracy, and system performance. In our approach, we focus on specific positions of BP models (i.e., process fragments). By measuring the similarity between process fragments, we can retrieve relevant process fragment for recommendation to assist the BP variant design without facing the complexity problem.

2.2.3 Business process querying

In order to accelerate the BP variant design, many research contributions have been made to query desired BPs and activities [92–98].

Awad et. al. [92–96] have developed a BP query language, namely BPMN-Q. Concretely, they extended the BPMN notations to develop BPMN-Q by adding new model elements, such as variable node which represents an unknown activity in a query, generic which indicates an unknown node in a BP, and path which states that there must be a path between two nodes [95]. Such query is considered as a graph. Thus, the query graph is then matched with existing BP models to retrieve BPs that relevant are to the query.

Several query-based approaches adopt text mining techniques. Hornung et. al. have developed an approach to query BPs using text mining techniques [97]. Concretely, They extract model element labels from Petri-net BP models, and translate them to a text-based document. Once the part of the BP is translated to a document, it is matched to other BP in the BP model repository based on the vector space model (VSM) and the term frequency inverse document frequency (TF-IDF) similarity metric. Lincoln et. al. [98] also use text mining techniques to perform matching between BP segments. They defined object grouping model (OGM) to represent a BP segment. OGM represents relationships between a primary object and others in a process segment. Edges connecting between objects are weighted by repetitions. Finally, the similarity between BP segments, represented in OGM is measured based on TF-IDF.

In general, existing query-based approaches perform matching between BP fragments. Therefore, they do not often face the graph-matching problem. However, in term of exploiting implicit knowledge, they barely consider process event logs. In our approach, we exploit both BP model and process event logs to assist the design of business BP variant.

2.2.4 Business process mining

Typical information systems, especially PAIS, record and accumulate their business transactions as process event logs [26]. Examples of such systems are ERP systems (e.g. SAP), case handling systems (e.g. FLOWer), workflow management systems (e.g.Staffware), CRM systems (e.g. Microsoft Dynamics CRM), middle ware (e.g., IBM'sWebSphere), hospital information systems (e.g., Chipsoft), and so on. Their process event logs can be analyzed to identify execution errors, a-priori BP models, and so on, which can be useful as a data-source to design business BP variants.

Several process mining approaches have been proposed to discover BP models from process event logs [99–103]. Some approaches mine process event logs and an a-priori BP model to check the conformance between them [104–107]. Several approaches focus on identifying BP execution errors [108, 109]. Some approaches examine social behaviors between groups of BP users [110,111]. Furthermore, our previous works [29, 30, 112] use process mining techniques to support the BP design.

In our approach, we exploit not only BP models but also process event logs as a data-source to assist the design of BP variants. We exploit process event logs because such logs alway exist in today's information systems and reflect actual behaviors of BPs during execution. By adopting process mining techniques, we can discover BP models from process event logs. Thereafter, we perform matching between fragments of discovered BP models and a selected fragment of an under-design process. Thus, an under-design process can be adjusted based on the matching result.

2.2.5 Synthesis

Existing approaches attempt to assist the (re-)design of BP models in which new BP variants are designed to meet new requirements. Several works proposed to use reference models as proven practices [82, 83] which are provided manually, thus cause error-prone and time-consuming. Many approaches perform matching between BP models based on their conceptual models [2, 84–87] or their process event logs [88, 89]. However, their approaches provide entire BP models while sometimes only some parts of the model need to be considered, and thus often high in computational complexity. Several approaches retrieve specific parts of BP models by using query [94, 96–98] to provide focused results. To sum up, presented approaches do not exploit the implicit knowledge offered by both BP models and process event logs, and only some of them partially assist BP variant design in an automatic manner.

Table 2.1 illustrates a synthesis of the presented approaches in terms of our last principles identified in Section 1.4.1: (*i*) automation, (*ii*) implicit knowledge exploitation, and (*iii*) focused results. None of them can fulfill all principles. '+' indicates that the corresponding principle is fulfilled by the corresponding approach, '−' indicates that the corresponding principle is not fulfilled and '+/−' indicates that the corresponding principle is partially fulfilled.

Approaches	Principles		
	Automation	Implicit knowledge exploitation	Focused results
[82], [83]	−	+/−	−
[2], [84], [85], [86], [87], [88], [89]	+/−	+/−	−
[94], [96], [97], [98]	+/−	+/−	+
[99], [100], [101], [102], [103]	−	+/−	−

Table 2.1: Synthesis of the approaches for assisting BP variant design

In our approach, we focus on a part of a BP (i.e., process fragment). We aim at recommending process fragment that are relevant to a selected position of a BP using our knowledge base constructed from heterogeneous BP models and process event logs (see Section 2.3 and 2.4). We take into account the relation of activities in a process fragment as a similarity metric. We compute similarity values between a selected fragment of an under-design BP model with process fragments represented in a knowledge base. This allows retrieving relevant process fragments which can be flexibly used in an under-design model.

2.3 Modeling heterogeneous business process models

Semantic interoperability problems of BPs have arisen especially in collaborative organizations' PAIS systems where heterogeneous processes are operated by multiple

branches in order to learn best practices from each other [113]. Generally, researches on semantic interoperability are classified into two categories: mapping and intermediary approaches. (*i*) Mapping approaches construct mappings between participant systems (in our case, BPs) and a global schema [114–116]. However, global schema does not allow a flexible adaptation of new participant system. (*ii*) Intermediary approaches model domain-specific knowledge, mapping knowledge, or rules in a machine-interpretable way such as ontologies [117]. Concretely, these approach formalize concepts and relationships between concepts as a shared common understanding within a community. Such formalization is independent of particular applications and schemas.

Semantic Web technology and ontology engineering researches have initiated substantial amount of interests to solve interoperability problems. In this section, we review existing ontology-based approaches for modeling processes.

2.3.1 MIT process handbook

The MIT process handbook project [118] has addressed the semantic interoperability problem of business knowledge by developing an approach to organize and share business activities in an online knowledge base. Concretely, they have proposed a set of fundamental concepts, namely entries, to model business activities in a knowledge base. They adopted the inheritance from the object-oriented programming paradigm to represent the specialization of activities and the dependency of coordination theory [119] to represent control flow between activities. The core entries described in the book are as follows:

- Description. Any information related to the given activity, such as definitions, comments, or links to other entries.
- Use. A set of activities in which the given activity is used as a part.
- Part. A part represents a structure of multiple activities.
- Generalization. The set of activities are type of the given activity.
- Specialization. The level specialized activities of the given activity.
- Bundle. A group of related specializations based on questions of an activity: what? when? where? why? who? and how?.

The MIT process handbook mainly describes generic reference models for typical business activities (e.g., buying, selling, and making). Their approach, however, do not consider the heterogeneity of BPs in term of process modeling languages. Hence, modeling heterogeneous BPs using their generic models might be error-prone and time consuming.

2.3.2 Process Specification Language (PSL)

The PSL [3] is one of the National Institute of Standards and Technology (NIST) projects that tackles the interoperability issues of manufacturing process information. Concretely, they have developed an ontology, namely PSL-Core, based on the Knowledge Interchange Format (KIF) specification [120]. PSL-Core defines four basic classes (Object, Activity, Activity Occurrence, and Timepoint) and four basic relations (Participates-in, Before, BeginOf, and Endof). Based on these generic classes and relations, they have developed an extension of PSL-Core to model business activities and their ordering relations (Figure 2.2). Other extensions can also be made based on the specific domain, such as the process planning domain described in the project.

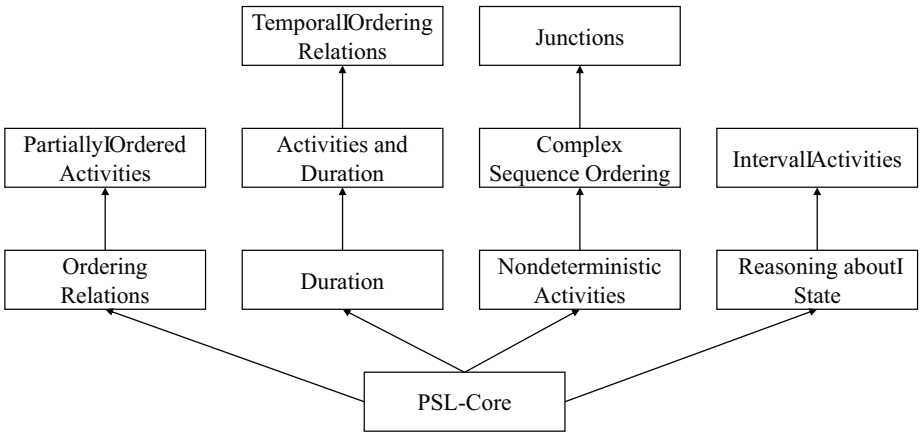


Figure 2.2: PSL-Core extension for modeling activities and ordering relations [3]

PSL-Core allows modeling BPs. However, they do not consider the heterogeneity of BPs in term of process modeling languages. Thus, modeling heterogeneous BPs using PSL-Core might be tedious and complex.

2.3.3 Process Interchange Format (PIF)

PIF [4] has a goal to develop an interchange format to automate process description exchange among different process modeling languages and support systems such as workflow software, flow charting tools, process repositories, etc. Concretely, PIF is a process modeling language developed based on the Knowledge Interchange Format (KIF) specification [120]. PIF defines three core classes for process description: Activity, Actor and Resource. PIF main classes and their relationships are illustrated in Figure 2.3.

PIF provides a common language through which different process representations can be automatically translated back and forth to PIF. However, the mechanism to achieve such automatic translation is the responsibility of individual groups who

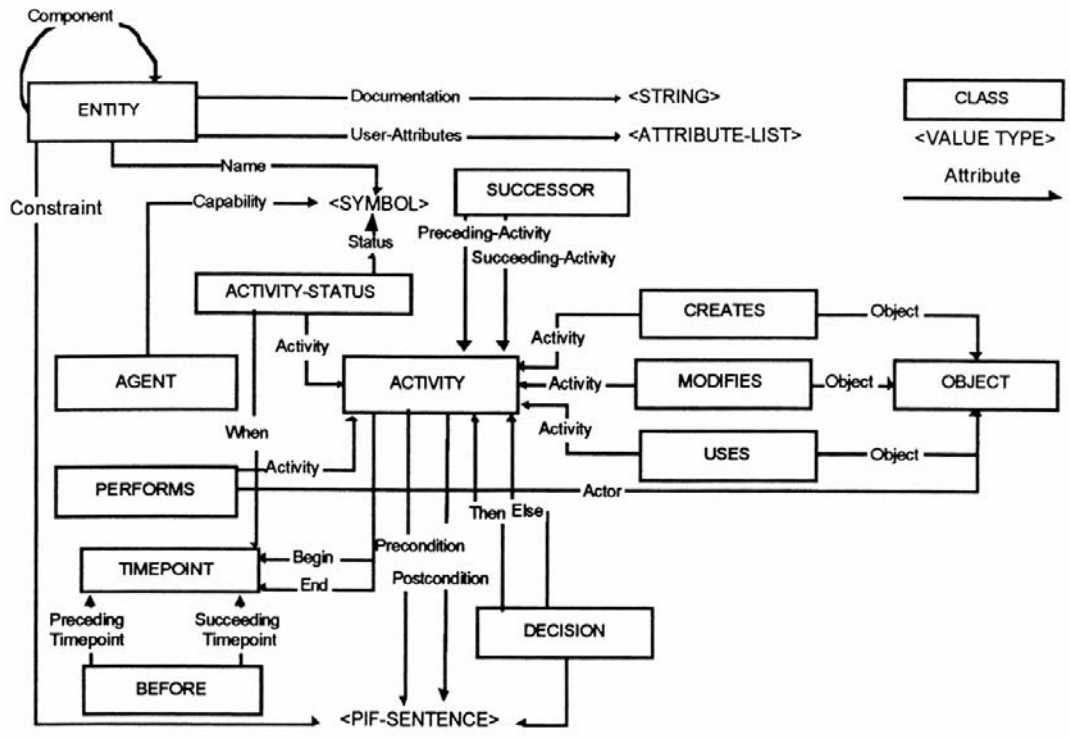


Figure 2.3: PIF classes and their relations [4]

want to use PIF. Therefore, PIF has not yet provided a tool or mechanism to model heterogenous BPs.

2.3.4 Web Ontology Language for Web Services (OWL-S)

OWL-S [5] is an ontology built on top of Web Ontology Language (OWL) for describing Semantic Web Services. It allows users and software agents to automatically discover, invoke, compose, and monitor Web resources offering services. The process ontology has been defined as a part of the OWL-S ontology to describe how to interact with a service that can be viewed as a process. Figure 2.4 illustrates the process ontology modeled in OWL classes, properties and axioms. They have defined 3 types of processes: (*i*) atomic processes representing the actions a service can perform in a single interaction; (*ii*) composite processes representing actions that require multi-step protocols and/or multiple actions; and (*iii*) simple processes providing abstract multiple views of the process.

OWL-S is criticized as suffers conceptual ambiguity, and offer an overly narrow view on Web Services [121]. Hence, using OWL-S to model heterogeneous BPs is not suitable.

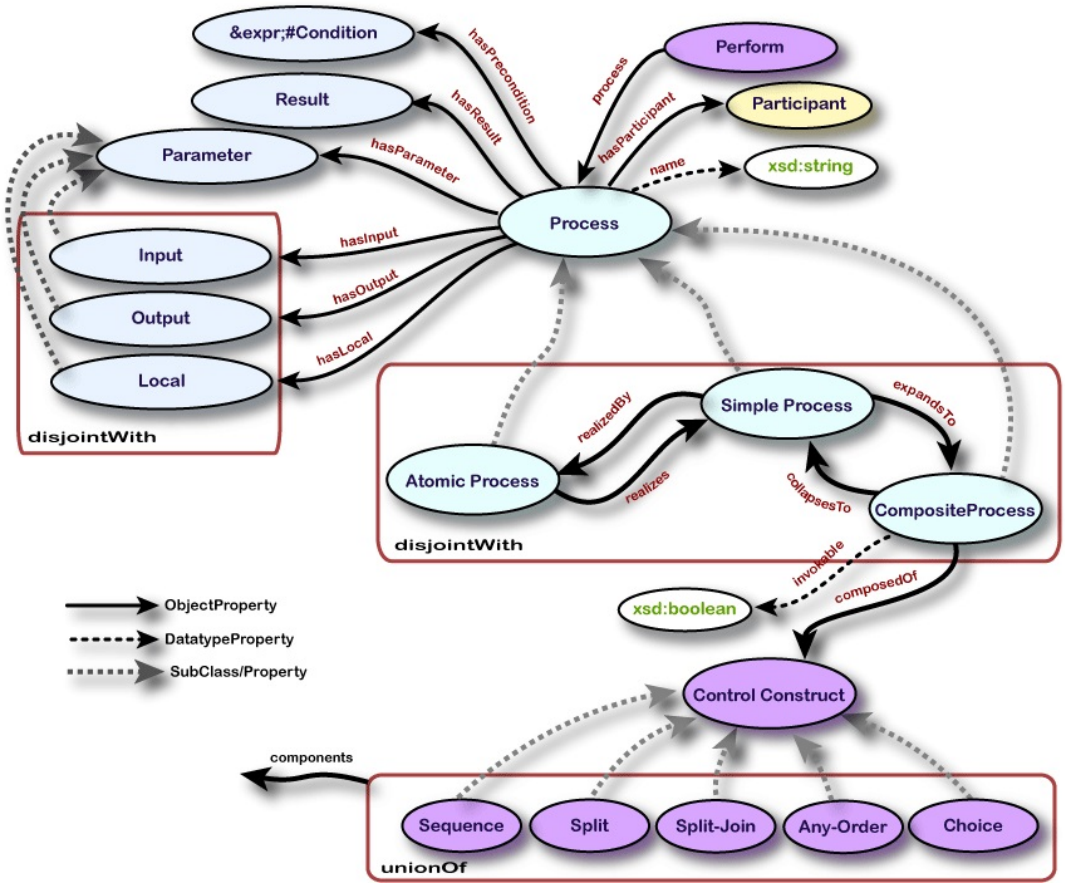


Figure 2.4: The process ontology of OWL-S [5]

2.3.5 General Process Ontology (GPO)

GPO [6] is an explicit and formal specification of concepts to model BPs in general. Concretely, it provides a common semantic annotation schema to represent BPs. Thus, GPO assists human and machines to understand heterogeneous BP models using semantic annotations. Example of the main concepts in GPO (Figure 2.5) are:

- Activity is a concept which composes a process.
- Artifact represents something involved in an activity such as information, software, etc.
- WorkflowPattern represents the ordering of activities. WorkflowPattern can be specialized into multiple specific patterns based on typical workflow patterns [122], such as Sequence, Choice (ExclusiveChoice, MultipleChoice, ParallelSpit), and Merge (SimpleMerge, MultipleMerge, Synchronization), which

are typical workflow control patterns supported by most process modeling languages.

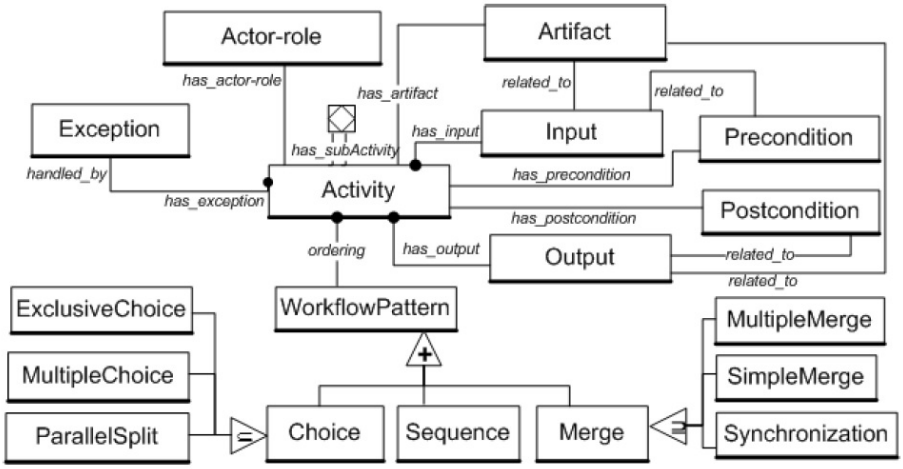


Figure 2.5: General process ontology (GPO) [6] represented using RML [7]

GPO is also defined as the semantic mediator for different process modeling languages. In article [123], authors provide case studies where two BPs (one described in BPMN [124] and another one described in EEML [125]) are translated into GPO. Hence, their approach has a capability to model heterogeneous BPs. However, their approach is relatively immature to model heterogeneous BPs.

2.3.6 Business Process Modelling Ontology (BPMO)

The BPMO [56] ontology is one of the research contributions of the European project SUPER [57]. It is capable of representing workflow elements from various process modeling languages. Concretely, BPMO abstracts different business process modeling notations such as BPMN [124] and EPC [125]. Figure 2.6 depicts BPMO concepts and relationships. BPMO core concepts are:

- *bpmo:Process* is an abstraction of a BP model.
- *bpmo:WorkflowElement* is a model element in a BP model. It has three sub-concepts: *bpmo:Task*, *bpmo:WorkflowEvent* and *bpmo:GraphPattern*. It is related to a process model through the property *bpmo:hasHomeProcess*.
- *bpmo:Task* is an atomic unit of work or activity in a business process model.
- *bpmo:WorkflowEvent* is an event that occurs during the execution of a business process. It has three sub-concepts *bpmo:StartEvent*, *bpmo:IntermediateEvent* and *bpmo:EndEvent*.

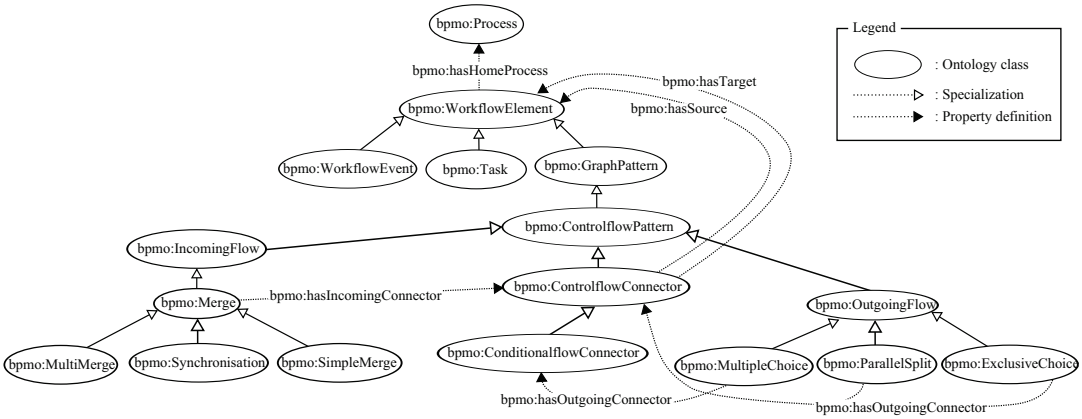


Figure 2.6: Business Process Modelling Ontology (BPMO)

- *bpmo:GraphPattern* represents the connection between workflow elements in a business process model. It has three sub-concepts:
  - *bpmo:ControlflowConnector* is the control flow (sequence) connecting two model elements through the properties *bpmo:hasSource* and *bpmo:hasTarget*.
  - *bpmo:IncomingFlow* represents variant types of logical connector having a merging behavior (i.e. join behavior). The *bpmo:hasIncomingConnector* property relates a merging connector to the workflow elements in its incoming flow. A *bpmo:Merge* connector has three main sub-concepts: *bpmo:MultipleMerge*, *bpmo:SimpleMerge* and *bpmo:Synchronization*.
  - *bpmo:OutgoingFlow* represents variant types of logical connector having a split behavior. The *bpmo:hasOutgoingConnector* property relates a split connector to the workflow elements in its outgoing flow. *bpmo:OutgoingFlow* connector has three primary sub-concepts: *bpmo:ExclusiveChoice*, *bpmo:ParallelSplit* and *bpmo:Multiplechoice*.

In the context of SUPER project, the *sBPMN* [126] and *sEPC* [53] ontologies have been also defined to formalize BPMN and EPC, respectively. By employing these ontologies, processes can be translated from BPMN and EPC to BPMO and vice versa [127]. Therefore, using such ontologies is suitable for modeling heterogeneous BPs.

2.3.7 Synthesis

Some existing approaches [3–6, 56, 118] tried to examine common properties of BPs, and thus define ontologies to semantically describe BPs in order to ensure the interoperability of such data within an organization. Most of them [3–6] developed tools

to assist the modeling of BPs to realize their ontologies. Some of them applied their approaches on use cases [6, 56, 118] to evaluate their solutions.

Table 2.2 shows a synthesis on the presented approaches in terms of the first two principles identified in Section 1.4.1: (*i*) heterogeneous data modeling, and (*iii*) automation. We review existing works regarding the *heterogeneous data modeling* principle by looking at BPs as data-source, and *automation* principle by focusing on the capability to automatically translate back and forth between heterogeneous BPs to the representation with their proposed ontologies.

Approaches	Principles	
	Heterogeneous BP modeling	Automatic translation
The MIT process handbook project [118]	+/−	−
PSL [3]	+/−	−
PIF [4]	+/−	−
OWL−S [5]	+/−	−
GPO [6]	+	+/−
BPMO [56]	+	+/−

Table 2.2: Synthesis of approaches for modeling heterogeneous BPs

In general, existing ontology-based approaches attempted to solve interoperability problems of BPs [3–6,56,118]. However, most of these approaches have not considered the heterogeneity problem of BP modeling languages [3–5,118], thus it might be error-prone and time consuming to model heterogeneous BPs using such approaches. On the other hand, some approaches [6, 56] have studied the BP heterogeneity problem and provide mediated mechanisms to facilitate the translation of heterogeneous BPs to constitute a shared knowledge base using their proposed ontologies.

Our approach aims at constructing a shared knowledge base of heterogeneous process data using ontologies. Reusing and extending existing ontologies is extremely valuable [128], thus we have decided to reuse and extend BPMO in our approach because BPMO has sufficient capability to model heterogeneous BP models. Moreover, this ontology is highly mature and is part of the SUPER European project [57] where a lot of research contribution has been done regarding to the interoperability of BPs.

2.4 Modeling heterogeneous process event logs

PAIS systems accumulates historical data of process execution (i.e., process event logs). In a large organization with multiple branches, each branch executes heterogeneous BPs on PAIS, and thus generates heterogeneous process event logs. Hence, the semantic interoperability problem of process event logs have arisen when organization branches are willing to share such data to learn from each other in order to improve their BPs. In addition, process event logs can be useful data-source to improve BPs using process mining techniques [27].

2.4.1 Event ontology and Time-Annotated RDF

As we mentioned previously in Section 2.3, using an intermediary approach (i.e., ontology-based) can help modeling process event logs and enabling interoperability within an organization. Some efforts have been made to conceptualize events as ontologies (Event ontology [59] and Time-Annotated RDF [60]). However, these proposed ontologies represent events for general use. Thus they do not describe relationships between events and BPs, and thus are not suitable for representing process event logs. In this section, we review existing ontology-based approaches for modeling process event logs.

2.4.2 Event Ontology (EVO)

Event Ontology (EVO) [129] is defined as a part of the European SUPER project [57]. Unlike other existing ontologies, events described with EVO can have explicit link to BPs by its defined relationships to BPMO ontology which is also part of the SUPER project. Figure 2.7 depicts those concepts and relationships. EVO main concepts are:

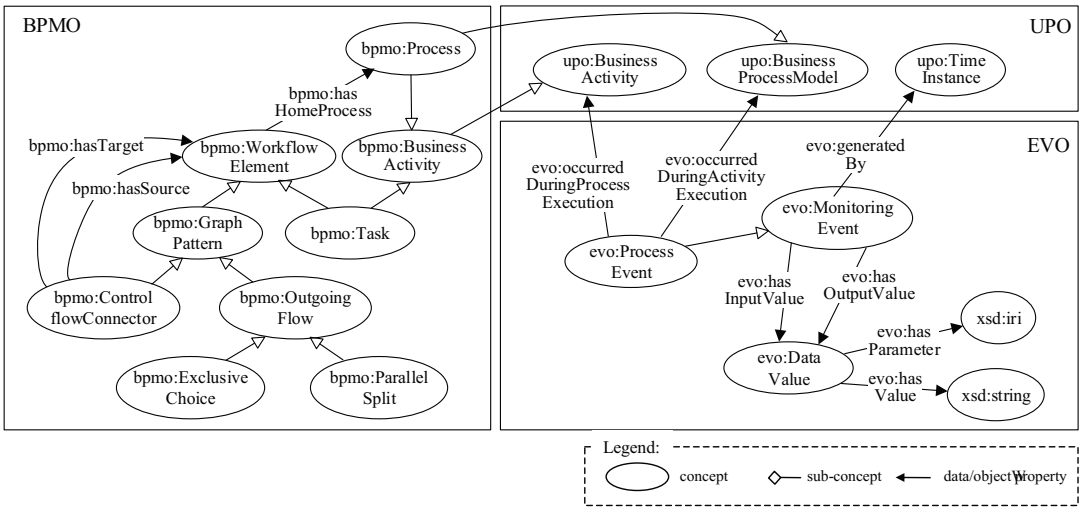


Figure 2.7: SUPER Event Ontology (EVO)

- *evo:MonitoringEvent* represents event taking place during the runtime of PAIS system. An agent which generates an event and a timestamp are described through the property evo:generatedBy and *evo:hasCreationTimestamp* respectively.

- *evo:ProcessEvent* represents an event recorded during the execution of a BP. *evo:occurred-*

DuringProcessExecution and *evo:occurredDuringActivityExecution* properties respectively describe a BP model and an activity which have been executed and consecutively event has been recorded.

- *evo:DataValue* is a generic attribute stored in an event. It represents a characteristic of an event, such as, title, description and so on.

EVO links to BPMO through UPO ontology because UPO acts as an integration point for all other SUPER ontologies. Note that UPO is an ontology defined by the SUPER project to formalize top-level concepts of BP and organizational-related entities [56]. The main concepts defined in UPO are: *upo:TimeInstance* represents a point in time, i.e. timestamp where event can take place during the execution of a BP; *upo:BusinessProcessModel* and *upo:BusinessActivity* respectively represents a BP model and an business activity.

2.4.3 Synthesis

Table 2.3 depicts a synthesis of the presented approaches in terms of the first two principles described in Section 1.4.1: (*i*) heterogeneous data modeling, and (*iii*) automation. We study the *heterogeneous data modeling* principle, and *automation* principle at process event log level by looking on the capability of automatic discover of BPs from process event logs using their proposed ontologies.

Approaches	Principles	
	Heterogeneous process event log modeling	Automatic translation
Event ontology [59]	+/−	−
Time−Annotated RDF [60]	+/−	−
Event Ontology (EVO) [129]	+	−

Table 2.3: Synthesis of approaches for modeling heterogeneous process event logs

Existing ontology-based approaches have capability to model events in general [59, 60, 129]. Only EVO ontology [129], part of SUPER project, can explicitly model process event logs because of its defined relationships to other SUPER ontologies (BPMO and UPO). Moreover, using SUPER ontologies allows modeling process event logs and discovered BP model when using process mining techniques. However, there is still lack of automatic mechanism to do such translation.

Our goal is to construct a shared knowledge base of heterogeneous process data using ontologies. Thus, we have decided to reuse and extend related SUPER ontologies (i.e., UPO, BPMO, and EVO) in our approach to semantically represent process event logs in a shared knowledge base. However, without proper data translation, process event logs cannot be directly compared to an under-design process for assisting BP variant design. Therefore, we propose to a discover BP model from process event logs, and thus store in the knowledge base. To do so, we apply the heuristic mining

technique introduced in [62] to discover BP model as Causal Net (C-net) [63]. We extend SUPER ontologies to define *Linked C-Net* ontology for representing discovered BP models in our knowledge base. This knowledge is later used as a data-source for assisting BP variant design.

2.5 Modeling heterogeneous cloud resources

In this section, we review existing approaches that aim at modeling heterogeneous cloud resources such that they enable interoperability between different cloud providers and users. Firstly, we discuss standards that allow describing cloud resources (Section 2.5.1). Therefore, we describe existing works in the literature regarding the modeling of cloud resources (Section 2.5.5).

2.5.1 Cloud resources description standards

In a large organization, each branch may adopt different cloud resource description standards to describe their cloud resources. Hence, this prevents the interoperability between an organization's branches at cloud resource level. In this section, we briefly introduce existing cloud resource description standards: TOSCA (Section 2.5.2), OCCI (Section 2.5.3), and CIMI (Section 2.5.4).

2.5.2 TOSCA

TOSCA [8] is an OASIS standard that provides a description specification in order to enable ted deployment and management of cloud-based application while being portable between different cloud environments [130]. A TOSCA application description consists of a Topology Template and a Plan (Fig. 2.8). The Topology Template defines the structure of the application and the Plan defines process models used to manage an application during its lifecycle (e.g., create, configure, and terminate). In this work, we focus on the topology template since it allows the description of cloud resources at the infrastructure level.

The Topology Template is a directed graph with Node Templates as its nodes and Relationship Templates as its edges. Node Templates represent an application component (e.g., compute and network) and are defined by a Node Type. A *Node Type* defines the properties and the offered operations (i.e., interfaces) to manage the application. Normative Node Types are defined in [131]. Examples of *Node Types* at infrastructure level are as follows: (*i*) Compute, a real or virtual server, (*ii*) Network, a simple, logical network service, (*iii*) BlockStorage, a block storage device, and (*iv*) Port, a logical entity that associates between Compute and Network.

Also, relations between these nodes might exist and are referred to as Relationship templates. The semantics and properties of these relationships are defined by Relationship type. Relationship type also indicates connected nodes and the direction of the

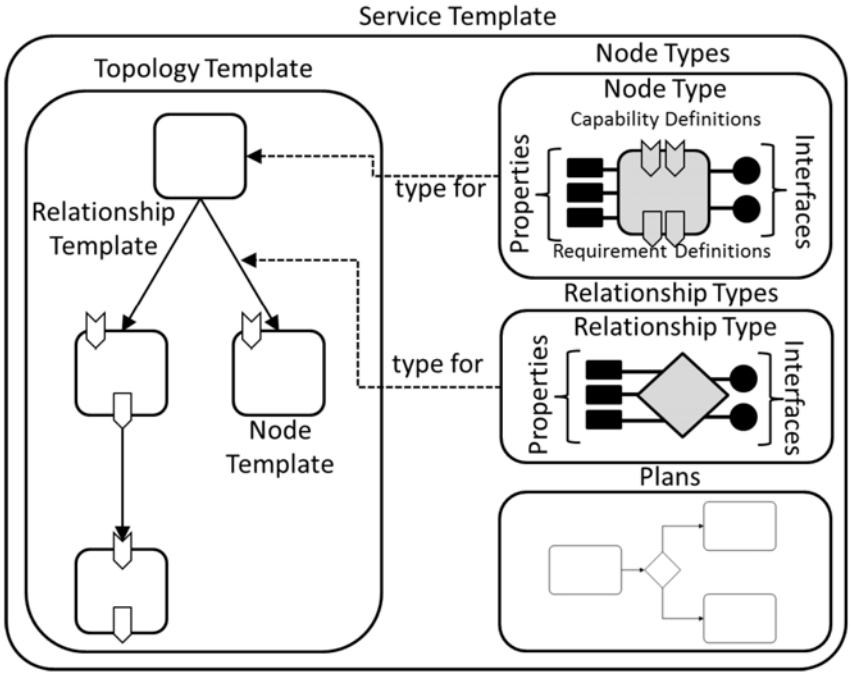


Figure 2.8: TOSCA topology and its relations [8]

relationship (source and target nodes). Examples of Relationship type at infrastructure level defined in [131] are BindsTo and LinksTo which are network association relationships from port to compute and to network node types, respectively. AttachesTo is a relationship between a Storage node to a *Compute* node.

2.5.3 OCCI

OCCI is an Open Grid Forum (OGF) standard that provides a meta-model for abstracting cloud resources and a RESTful protocol for their management. It focus on interoperability and offer a high degree of extensibility. OCCI provides an extensible model, called OCCI core model, for cloud resource description [132]. Concretely, the model defines a Resource class to expose any cloud resource and a Link class which describes associations from one Resource to another. In order to describe resources at the infrastructure level, OCCI infrastructure [9] is proposed as an extension of the OCCI core model.

In OCCI infrastructure, the Resource from the core model is specialized into: (*i*) Compute, representing a generic processing resource, (*ii*) Network, representing a networking entity (e.g. a virtual switch), and (*iii*) Storage, representing a resource that record information to a data storage device. Link is also specialized in OCCI infrastructure to: (*i*) NetworkInterface, representing an interaction between Compute and Network (e.g. network adapter), and (*ii*) StorageLink, representing an interaction be-

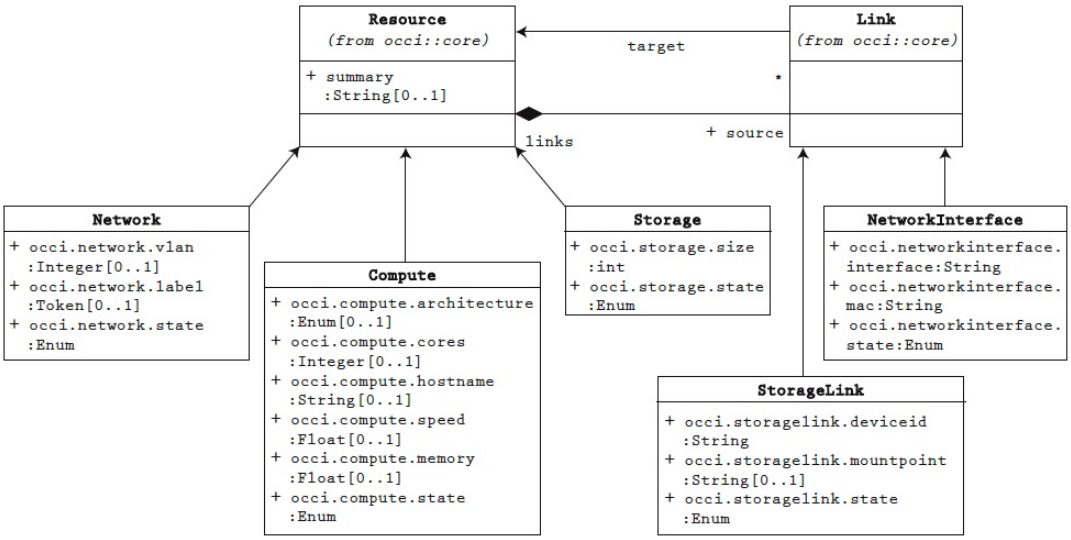


Figure 2.9: UML class diagram of OCCI infrastructure model [9]

tween Resource to a Storage.

2.5.4 CIMI

CIMI is a Distributed Management Task Force (DMTF) standard that describes standardized ways to set up and manage infrastructure resources on the cloud. It provides a specification to describe and manage cloud resources [133,134]. Some uses cases are provided in [135]. CIMI defines the following basic resource types at the infrastructure level: (*i*) Machine, representing a compute resource that encapsulates both CPU and Memory, (*ii*) Network, representing an interconnected logical service for forwarding data traffic between end points, and (*iii*) Volume: representing a storage resource either at the block level or the file-system level. CIMI also defines associations for connecting these resource types: MachineNetworkInterface between Machine and Network, and MachineVolume between Machine and Volume.

2.5.5 Cloud resources modeling

In the literature, several works have been proposed to standardize cloud resource management between different cloud provider. For example, the authors from [136] deliberate the need for a generic API that manages cloud resources across different cloud providers. They have investigated that most of available cloud management APIs use similar concepts, properties, and actions but different names and schema. Therefore, they propose modeling of a common cloud API based on semantic technologies. In [137], the authors suggest common API (an API for all APIs) that specifies

a set of core functionalities needed for any cloud provider. The IEEE P2302 Working Group is working on defining a standard to support the management of cloud resources among different providers, namely Intercloud [138]. Concretely, they aim to define a topology, a set of functionalities, and a model to support cloud interoperability among different providers. In [139], authors propose an ontology to semantically represent cloud resources modeled with Intercloud. However, Intercloud is a relatively new work-in-progress and focuses on interoperability between cloud providers. Moreover, several cloud brokers (i.e., CompatibleOne [140]) have been proposed to assists cloud end users managing cloud resource in their cloud provider choices. Such brokers: (*i*) do not consider standard or comply with only one standard, and (*ii*) do not offer means for discovery, managing and interconnecting cloud resources. Thus, we propose a semantic framework that allows populating a knowledge base of heterogeneous cloud resources to enable interoperability between them.

Several approaches focus on defining taxonomies or ontologies to describe cloud resources. Authors from [141] proposed a taxonomy for IaaS cloud resources. They developed their taxonomy based on the analysis of cloud providers and Web hosting providers. Another work from [142] also proposed a taxonomy to describe cloud elements at the infrastructure level. Their taxonomy discusses many aspects of cloud, such as resources, security, Service-Level Agreements (SLAs), etc. In fact, their taxonomy is built based on three cloud standards: Open Virtualization Format (OVF), OCCI and Cloud Data Management Interface (CDMI). These three standards have been defined for different cloud aspects. OVF [143] is a preliminary standard of DMTF for packaging and distributing of Virtual Applications. CIMI [73] is a Distributed Management Task Force (DMTF) standard that describes infrastructure resources in the cloud. CDMI [144] is a Storage Networking Industry Association's (SNIA) standard that provides a specification for rich cloud storage management interfaces. [145] proposed a semantic framework for managing grid and cloud resources. Their framework is based on an ontology developed from their previous work [146]. However, they built an ontology without considering cloud resource description standards. Authors from [147] proposed an ontology to model cloud resources, SLA and their configurations. They studied only one cloud resource description standard which is TOSCA standard. In [148], the authors presented an ontology-based approach to describe cloud resources. They developed a set of ontologies based upon OCCI standard. Existing approaches, however, model cloud resource description by taking into account only one standard. In this work, we focus solely on cloud resource description modeling by looking at three well-known and mature cloud resource description standards (TOSCA, OCCI and CIMI) to develop our ontologies based upon these three standards.

2.6 Cloud resource allocation to business processes

In this section, we review existing approaches that provide means to allocate cloud resources to BPs. Recently, organizations have investigated the deployment of service-based BPs in cloud environments. S. Schulte, E. Duipmans and M. Wang et al. in [149–151] discuss the advantages and disadvantages of deploying BPs in cloud environments. Few works handle the resource perspective of BPs. For example, some works focus on human resources behavior and allocation [152, 153]. In our work, we focus on the cloud resource allocation to BP activities.

Many approaches have been proposed to optimally allocated resource to BPs. For example, Hoenisch et al. [154, 155] have proposed a prediction and reasoning algorithm to allocate cloud resources to BP activities under user-defined non-functional requirements. Schulte et al. [156] have presented an optimization model and heuristic for workflow scheduling and resource allocation for BP execution with the ability to allocate and release cloud resources. However, solving optimal resource allocation might be combinatorial problem due to the amount of possible allocation variants. Genetic algorithm is one of the powerful means to deal with such problem. It has been preferred than other approaches in large-scale complexity problems. Existing approaches, however, have not address the optimal resource allocation problem using genetic algorithm.

2.7 Conclusion

In this chapter we present approaches that support the design of BP variants, in which we classified approaches into four categories: business process modeling, business process similarity, business process querying and business process mining. Furthermore, we review approaches that model heterogeneous process data (i.e., BP models and process event logs) and cloud resources. We briefly introduced these approaches and identified their principles. We present that most of the existing ontology-based approaches are not capable of modeling heterogeneous cloud resources and process data. Such approaches also lack of automatic mechanism to transform heterogeneous process data and cloud resources to a shared common knowledge. On the other hand, most of existing approaches that assist the design of BP variants do not generate focused results, encounter the computational complexity problem, and are currently manual. Moreover, we present the difference between existing approaches and our approach.

We start presenting in detail our approach in the next chapters. We present our approach to support the design of BP variants using heterogeneous BP models (Chapter 3) and process event logs (Chapter 4) in which we model in a common knowledge base. In Chapter 5, we present an approach for modeling heterogeneous cloud resource described by different standards. Thereafter, we present an approach to optimal allocate of cloud resources to BP variants using genetic algorithm in Chap-

ter 6.

Chapter 3

Assisting BP Variant Design by Exploiting BP Models

Contents

3.1 Introduction

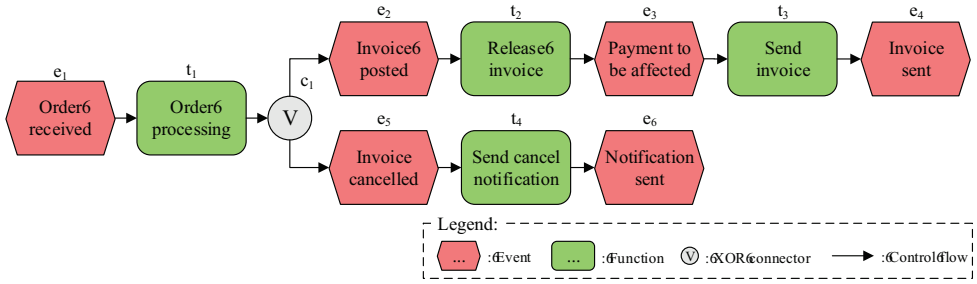
This chapter address the research questions: *R1: How to identify BP fragments that are close to process designer interests from heterogeneous process data?* and *R2: How to model heterogeneous BP models to be shared in a common knowledge base?* We presents our approach for assisting the design of BP variants by exploiting heterogeneous BP models shared within an organization. We start the chapter by introducing an illustrating example (Section 3.2) and overview of our approach (Section 3.3). In section 3.4, we present our ontologies and algorithms to construct a knowledge base from heterogeneous BP models. In section 3.5, we detail the semantic representation of BP fragments from our knowledge base. Section 3.6 continues with the matching

of BP fragments for recommendation to assist the design of BP variants. Finally, we conclude the chapter in Section 3.7.

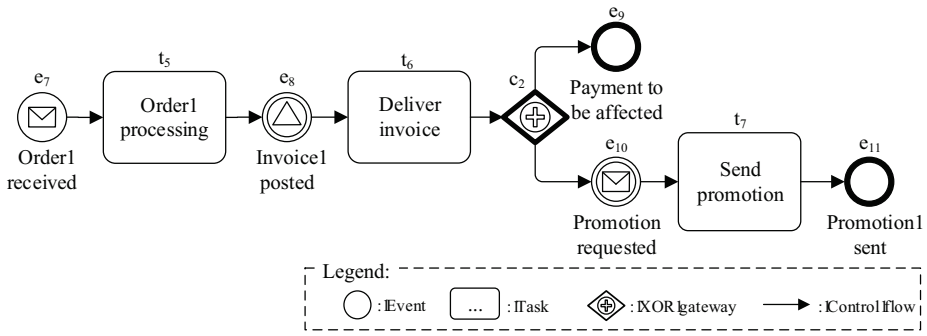
The work in this chapter was published in conference proceedings [157].

3.2 Illustrating example

We present in the following a scenario to illustrate our approach. We consider an e-commerce company that has multiple branches in different cities and countries. These branches execute different variants of the same process that may differ in their structure and behavior according to their branches' local culture, regulations, and customer needs. We show two variants of an order processing BP executed by two branches: $branch_1$ (Figure 3.1a) and $branch_2$ (Figure 3.1b) which are modeled in EPC and BPMN, respectively. EPC is a graphical modeling language for business process workflows [22]. It has been applied in ERP system describing workflow, e.g. SAP R/3 [82]. The main elements of EPC notations are events, functions and connectors. On the other hand, BPMN is a standard of modeling BPs [23]. The basic elements of BPMN are event, task, process/subprocess, gateways, etc.



(a) BP_1 of $branch_1$



(b) BP_2 of $branch_2$

Figure 3.1: Two BP variants of an order processing BP

The first variant BP_1 (Figure 3.1a) corresponds to an order processing BP modeled in EPC. Upon instantiation of BP_1, the order data is first received (event e_1) from

a customer and processed (task t_1) by a member. Then, the invoice can be either automatically posted (event e_2) or cancelled (event e_5) based on some criteria. If the invoice is posted, a member will release/issue the invoice (task t_2) to the customer, thereafter waiting for payment to be affected (event e_3). Once it is affected, a member will send the invoice to the customer (task t_3 and event e_4). However, if the invoice is cancelled (event e_5) after processing, a member will send the cancel notification to the customer (task t_4 and event e_6). The second variant BP_2 (in Figure 3.1b) is modeled in BPMN, and corresponds also to an order processing BP. The steps are approximately the same as the first variant with the addition of the promotion offering functionality (event e_{10} and task t_7).

Suppose now that $branch_1$ wants to design a new variant of BP_1 to support more requirements. It may need to search for BP models representing good practices, then carefully extract adequate prcess fragments, and apply them on the designed BP model. However, this might be a cumbersome and time consuming task when ensured manually. As stated in Chapter 2, existing approaches (such as reference models, searching for similar processes, etc.) that propose entire process models are not suitable in this circumstance because the process designer only looks for adjusting a selected BP fragment. Hence, recommending BP fragments during a BP variant design could alleviate a process designer's workload. For example, $branch_1$ BP designer might look for suitable BP fragment (i.e., a set of tasks and their connection flows) that can be executed after releasing an invoice task. Thereafter, he selects a group of tasks and their relations, called BP fragment, including tasks t_2 and t_3, then ask a recommendation system for recommendation.

Based on this illustrating example, we notice that the process designers might need to search for appropriate BP fragments that can be flexibly inserted with minimum adjustment in a given BP model. By capturing the interactions between tasks in processes, we can detect that the fragment of BP_2 including tasks t_6 and t_7 at $branch_2$ is similar to the selected fragment of BP_1 (see Figure 3.2). Such recommendation allows process designers to easily get new ideas for designing new BP variants as shown in Figure 3.3.

In this example, we consider a use case where organization's branches are willing to collaborate and share their BP models. Therefore, we propose to recommend BP fragments based on similarity to other fragments using interaction between functions/tasks. However, an issue that prevents fostering the use of BP models between different organization's branches is their heterogeneity. An organization's branches might have similar BPs in term of functionalities, however their BP models could be defined in different modeling notations and using different element's labels. For example, BP_1 is modeled with EPC, while BP_2 is modeled with BPMN (see Figure 3.1).

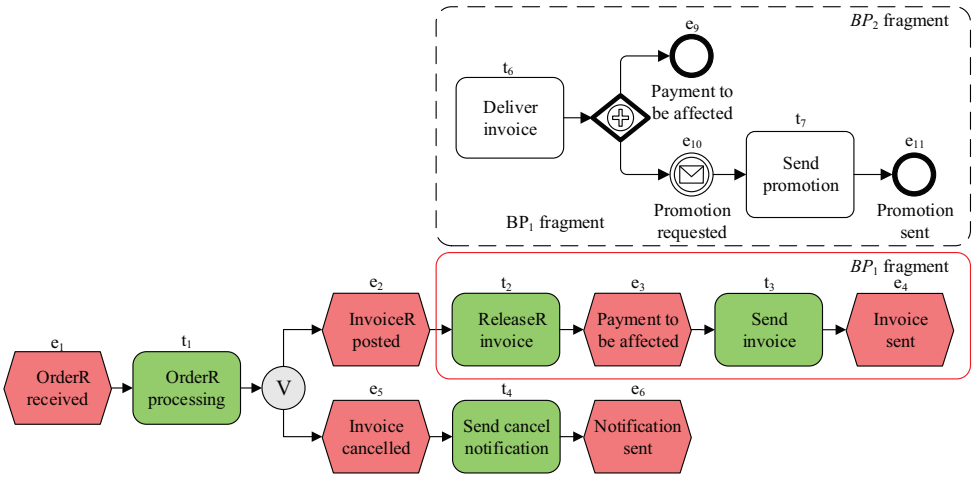


Figure 3.2: BP_2 fragment recommendation for the selected BP_1 fragment

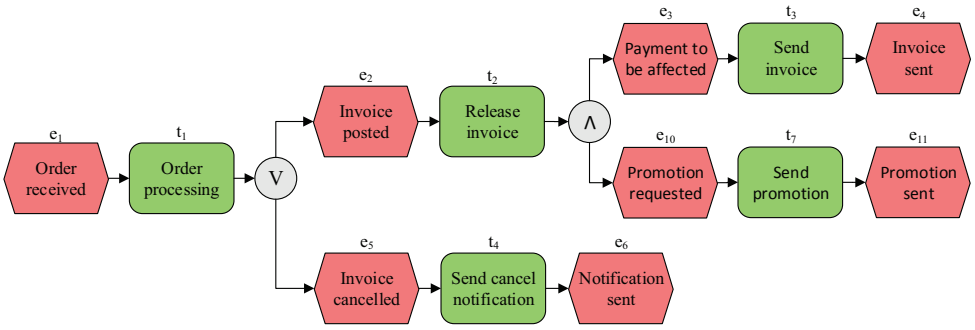


Figure 3.3: New BP variant for $branch_1$

3.3 Approach overview

To foster BP variant design, we provide a comprehensive semantic framework for managing the heterogeneity of BP models (Figure 3.4). Based on this framework, we propose an approach which includes four main steps:

1. Semantically populate a shared knowledge base of heterogeneous BP models (Section 3.4). This knowledge base will serve as a common repository for BP models. Our knowledge base is built based upon a mature ontology describing BP models, namely BPMO [56].

2. Retrieve BP fragments from the shared knowledge base modeled as neighborhood context graphs (Section 3.5). We semantically describe such fragments using our proposed ontology, namely NCFO, which is an extension of BPMO.

3. Compute the matching between neighborhood context graphs using vector space

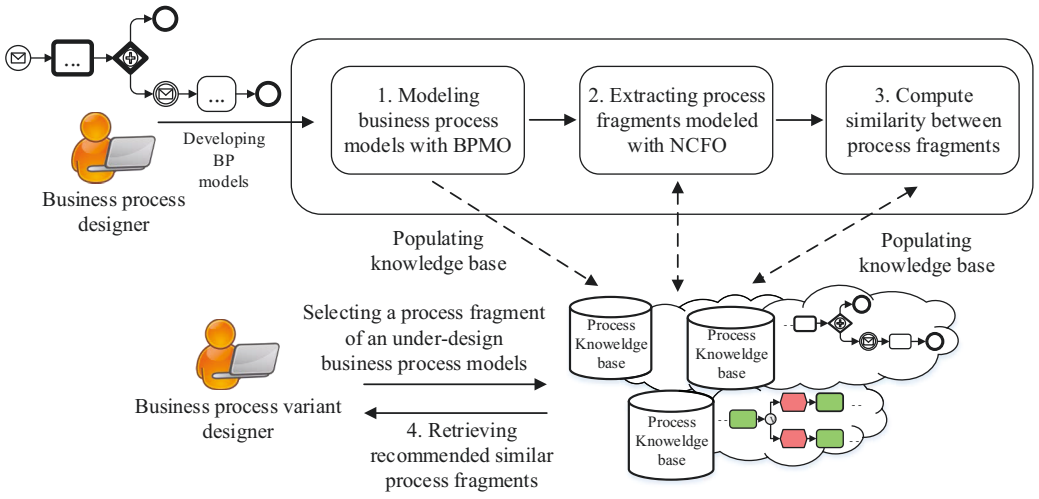


Figure 3.4: An overview of our approach

model (Section 3.6). This matching presents the similarity between two fragments in term of relations between the two root tasks to their neighbors.

4. Finally, a process designer may select a BP fragment of an under-design process. Thereafter, we retrieve and sort top-n similar fragments from our knowledge base in descending order of similarity for recommendation.

In the next section, we show how to populate a shared knowledge base of heterogeneous BP models.

3.4 Knowledge Base of Business Process Models

We define a BP model, also called BP graph, as a directed graph with labeled nodes. This definition is an abstraction of the common elements of existing graphical BP modeling notations [158] such as Event-driven Process Chain (EPC) [125] and Business Process Modeling Notation (BPMN) [124].

Definition 3.4.1 (BP graph)**.** *A BP graph $P = (N, S, T, L)$ is a labeled directed graph where:*

- *N is a set of nodes representing BP modeling notation (i.e., workflow element), for example, a function in EPC and an activity in BPMN.*
- *$S \subseteq N \times N$ is the set of arcs connecting two nodes;*
- *$T : N \to t$ is a function that assigns for each node $n \in N$ a type t. t depends on the element type for each standard notation (e.g., function and event for EPC);*

- $L : N \rightarrow label$ *is a function that assigns for each node* $n \in N$ *a given label. label is a string representing the functionality of its workflow element.*

An example of a BP graph modeled with the EPC notation is illustrated in Figure. 3.1a). This BP graph model is represented by $P_1 = (N, S, T, L)$ where $N = \{e_1, e_2, ..., e_6, t_1, t_2, ..., t_4, c_1\}$, $S = \{s_1, s_2, ..., s_{10}\}$. EPC workflow elements type e, f, and c in N are represented respectively by $T = \{event, function, connector\}$. Give the workflow elements labels are defined as follows: for $n \in N$, if $T(n) =$ event $\vee$ function, then $L(m)$ is its label, for example $L(t_1) =$ *"Order processing"*. If $T(n) = connector$ then $L(n) \in \{\vee, \wedge, \times\}$ representing OR, AND, and XOR connectors, respectively.

As stated above, process models can be described in many different languages (e.g., EPC, BPMN, UML, Petri nets, etc.). Furthermore, workflow element labels added by process designers in natural language imply more semantic heterogeneity in BP models. The analysis of such heterogeneity in [58] shows that workflow elements (i.e., symbols) do not directly represent the things they stood for. To avoid such heterogeneity, workflow elements can be annotated through concepts formalized by experts to explicitly establish their semantics of a BP modeling notation. In this thesis, we propose to semantically annotate workflow elements with concepts from dedicated ontologies (section 3.4.1).

3.4.1 Business Process Modeling Ontology (BPMO)

We use a dedicated ontology to formally represent BP models in our knowledge base. Using an ontology allows capturing the semantics of BP models such that they are represented by reusing conceptual models [159]. It also handles heterogeneity of workflow element labels. Furthermore, it supports automated reasoning which allows knowledge base enriching (i.e., new knowledge discovery). In this work, we formally define an ontology as follow:

Definition 3.4.2 (Ontology)**.** *An ontology* $\mathcal{O}$ *is defined as* $\mathcal{O} = (C, A, R, I)$ *where* C *is a set of concepts,* A *is a set of data properties that define the relationship between concepts and primitive data type (e.g., string, integer),* R *is a set of object properties that define the relationship between different concepts and* I *is the set of object instantiated from concepts (i.e., instance).*

To cope with the problems of semantic heterogeneity between different process modeling languages, we use Business Process Modeling Ontology [56] (*BPMO*) as mentioned in Section 2.3.6 to semantically annotate BP models from existing heterogeneous processes among organization's branches. BPMO is an existing ontology formalized by the european project SUPER [57]. BPMO is capable of representing notations from various process modeling languages. It abstracts different business process modeling notations such as BPMN and EPC. In the context of SUPER

project, the *sBPMN* [126] and *sEPC* [53] ontologies have been also defined to formalize BPMN and EPC, respectively. By employing these ontologies, processes can be translated from BPMN and EPC to BPMO and vice versa [127].

The core concepts of BPMO that abstract the representation of BP models are *bpmo:Process*, *bpmo:WorkflowElement*, *bpmo:Task*, *bpmo:WorkflowEvent* and *bpmo:GraphPattern* (see Section 2.3.6 for details). Figure. 3.5 depicts these concepts and their relationships. In this Figure, NCFO ontology representing fragments of BP models will be introduced later in Section 3.5.1

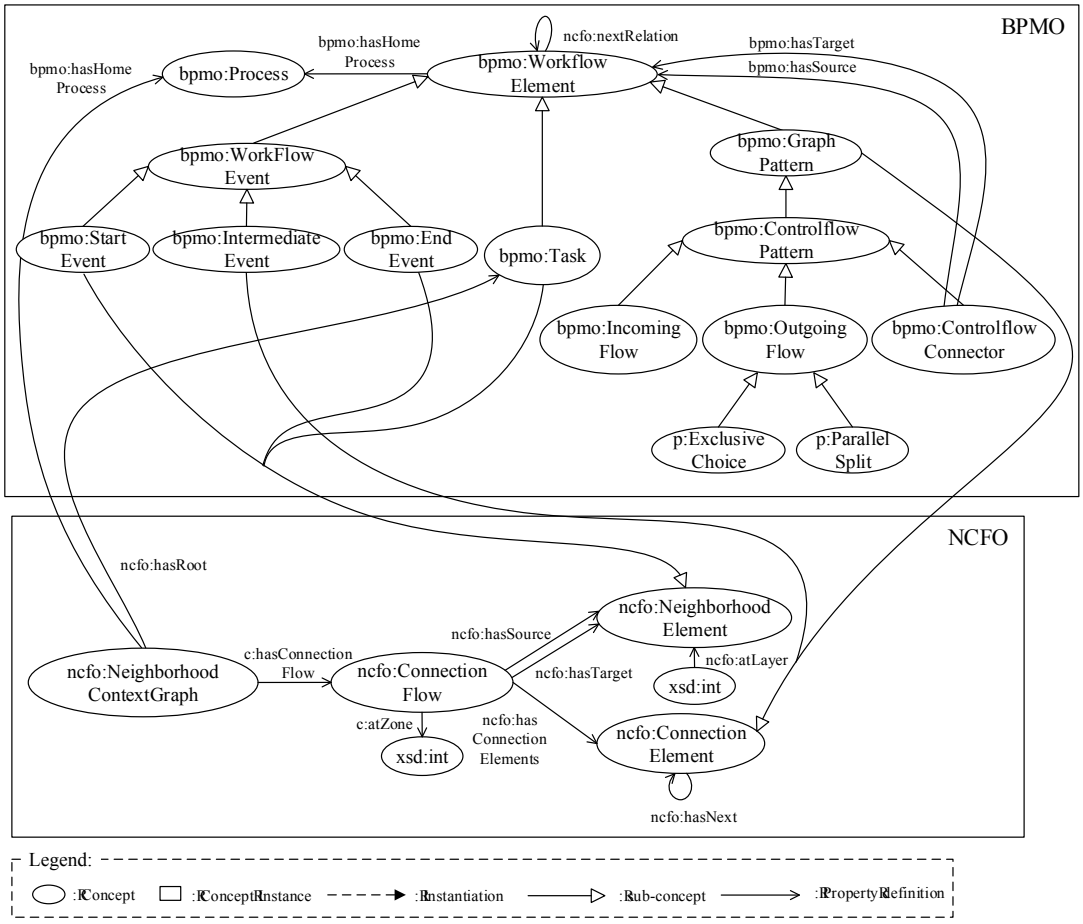


Figure 3.5: BPMO and NCFO

3.4.2 Semantic Business Process Model Annotation

To build a knowledge base from heterogeneous BP models. We semantically annotate semantic BP models which enables: (*i*) interoperability between different BP modeling notations through BPMO ontology, and (*ii*) formalization of business understand-

ing in a given domain through an ontology, i.e., domain-specific ontology. Concretely, we conceptualize the abstraction of BPMO concepts to other modeling notations [127] to annotate heterogeneous process models with BPMO concepts. Firstly, we define a BP model annotated with BPMO concept as a *semantic BP graph*. A semantic BP graph is *semantically annotated with concepts from an ontology* $\mathcal{O}$ and it is defined as:

Definition 3.4.3 (Semantic BP graph)**.** *A semantic BP graph* $P_S = (P, \mathcal{O}, I_N, I_O)$ *is defined as:*

- P *is a BP graph as specified in Definition 3.4.1;*
- $\mathcal{O}$ *is an ontology as specified in Definition 3.4.2;*
- $I_N : \mathcal{O}.I \rightarrow P.N$ *is a function that assigns for each instance* $i \in \mathcal{O}.I$ *a workflow element* $n \in P.N$ *if* i *annotates it; and*
- $I_O : \mathcal{O}.I \rightarrow \mathcal{O}.C$ *is a function that assigns for each instance* $i \in \mathcal{O}.I$ *a concept* $c \in \mathcal{O}.C$ *from which* i *is instantiated.*

In order to verify that two notations described with different BP modeling languages are equivalent, we define the notion of equivalence between BPMO concepts and elements from other standard modeling languages (see Definition 3.4.4).

Definition 3.4.4 (BPMO notation equivalence)**.** *Let* $\mathcal{N}$ *be a set of modeling notations where* $\mathcal{N}_x$ *is the set of notation elements in the modeling language* x*. Let* O_{bpmo} *be the BPMO ontology. The BPMO notation equivalence, defined as* $\Pi_{bpmo} = \{(c, n) : c \in O_{bpmo}, n \in \mathcal{N}_x\}$ *is the pairs of semantically equivalent mapping of BPMO concepts to the notation elements in the modeling language* $\mathcal{X}$*.*

For example, let $\mathcal{X} = EPC$ denoting the EPC language and $N_{EPC} = \{$*Function*, *StartEvent*,*IntermediateEvent*,....$\}$ the notation elements in EPC. The *Function* in EPC is equivalent to the concept *bpmo:Task* in BPMO and the *StartEvent* is equivalent to *bpmo:StartEvent* in BPMO; therefore (*Function*,*bpmo:Task*) and (*StartEvent*, *bpmo:StartEvent*) $\in \Pi_{bpmo}$. Table 3.1 illustrates the mapping between the general connectors $(\times, \vee, \wedge)$ presented in most process modeling languages to BPMO connector concepts. Note that BPMO classifies connectors based on their behavior into 2 types: (i) merging: a point where two or more sequence flow paths are combined into one, and (ii) splitting: the dividing of a sequence flow path into two or more. More details on the BPMO mapping equivalence to other modeling languages can be found in [127].

Given a BP graph P, the BPMO ontology $\mathcal{O}_{bpmo}$ and the BPMO equivalence Π_{bpmo}, Algorithm 1 generates the associated semantic BP graph P_S. In P_S, all workflow elements $n \in N$ are annotated with a set of instances of $\mathcal{O}_{bpmo}$ concepts.

To do this, firstly, we iterate over each workflow element $n \in N$ in P_S and we retrieve its BPMO equivalence from Π_{bpmo} (lines 2-4). Using the retrieved concept c

	BPMO concepts	
Connectors	**Merging behavior**	**Splitting behavior**
$\vee$	MultiMerge	MultipleChoice
$\times$	SimpleMerge	ExclusiveChoice
$\wedge$	Synchronisation	ParallelSplit

Table 3.1: Concepts of BPMO connectors by types and behaviors

Algorithm 1 Semantic BP graph construction algorithm

Input: $P, \mathcal{O}_{bpmo}, \Pi_{bpmo}$
Output: P_S

1: $N \leftarrow GetModelElements(P)$
2: **for each** n **in** N **do**
3: **for each** (c, x) **in** Π_{bpmo} **do**
4: **if** $T(n) = x$ **then**
5: $i \leftarrow Instantiate(n, c)$ ▷ Create a new instance with assocaited data/object properties based on the workflow element n and the concept c
6: $I_N \leftarrow i$
7: $I_O \leftarrow c$
8: **end if**
9: **end for**
10: **end for**
11: $P_S.P \leftarrow P$
12: $P_S.\mathcal{O}_{bpmo} \leftarrow \mathcal{O}_{bpmo}$
13: $P_S.I_N \leftarrow I_N$
14: $P_S.I_O \leftarrow I_O$
15: **return** P_S

from BPMO equivalence, we instantiate a new instance i with asserted data/object properties c from the model element n (line 5 and 6). We add the created instance to the set of instances I_M (Line 6). Finally, a semantic BP graph P_s is constructed from P, $\mathcal{O}_{bpmo}$, I_N, and I_O (line 11-14).

Figure. 3.6 shows examples of two semantic BP graphs (partially annotated with BPMO concepts) constructed using Algorithm 1. The semantic process models described in EPC (BP_1) and BPMN (BP_2) have the instances t_2 and t_6, respectively. t_2 and t_6 are semantically annotated to the function "Release invoice" in the EPC model BP_1 and the task "Deliver invoice" in the BPMN model BP_2, respectively. Consequently, these two workflow elements are considered as tasks in BPMO. Therefore, they are semantically BPMO notation equivalent.

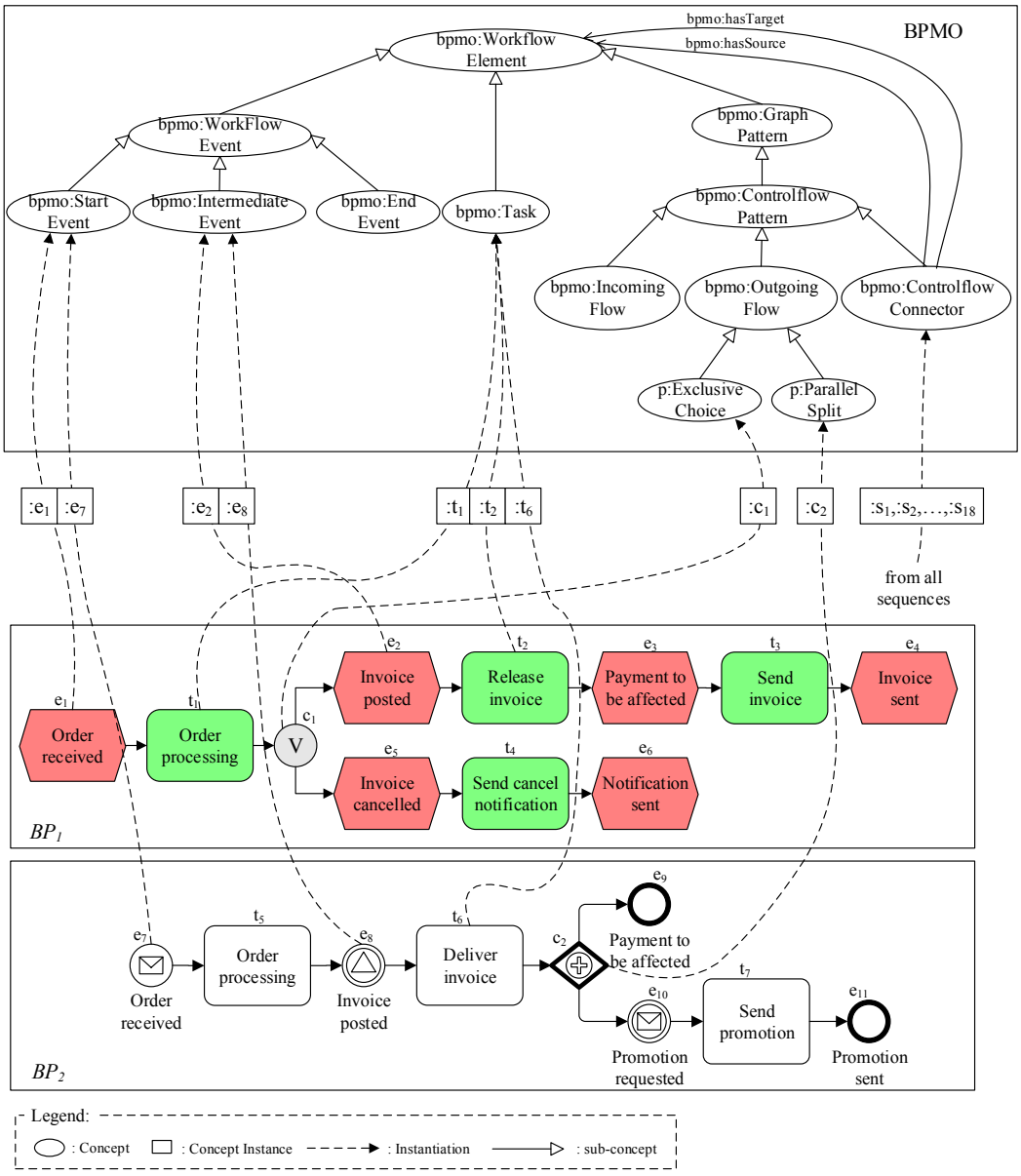


Figure 3.6: Partially semantic annotation of process models with BPMO ontology

After the BPMO annotation, we annotate semantic BP models with a domain-specific ontology to enable common understanding of BPs models in a given domain. Figure. 3.7 shows BP_1 and BP_2 annotated with a domain-specific ontology. The prefix "d" represents the namespace of the domain-specific ontology. For example, the task "Release invoice" in BP_1 and the task "Deliver invoice" in BP_2 are annotated

with the instances t_2 and t_6 respectively having *bpmo:hasBusinessDomain* relations to the concept *d:release_invoice*. The relation *bpmo:hasBusinessDomain* of BPMO represents the functionality of a workflow element by its annotated concept from a domain-specific ontology. Therefore, tasks t_2 and t_6 having different labels are semantically equivalent, i.e. they have the same functionality represented by the concept *d:release_invoice*.

Figure 3.7: Partially semantic annotation of processes with a domain ontology

A knowledge base can serve as a common repository to model and share heterogeneous data. Hence, we populate a shared knowledge base from semantic BP graphs. For example, a knowledge base of semantic BP graphs constructed from

BP_1 is shown in Listing 3.1. In this thesis, we represent the knowledge base using Turtle RDF triple[1] for simplicity. For example, *t2* is instantiated from the BPMO concept *bpmo:Task* which is in the BP model *bp1* describing through the object property *bpmo:hasHomeProcess*. Its label "Release invoice" is described through the data property *bpmo:hasName*. Its functionality is described by the specific-domain ontology concept *d:release_invoice* through the object property *bpmo:hasBusinessDomain*.

Listing 3.1: A BP model described with BPMO

```
:t2 a bpmo:Task ;
  bpmo:hasHomeProcess :bp1 ;
  bpmo:hasName "Release_invoice"^^xsd:string ;
  bpmo:hasBusinessDomain d:release_invoice ;
  ...
:t6 a bpmo:Task ;
  bpmo:hasHomeProcess :bp2 ;
  bpmo:hasName "Deliver_invoice"^^xsd:string ;
  bpmo:hasBusinessDomain d:release_invoice ;
  ...
:e1 a bpmo:StartEvent;
  bpmo:hasHomeProcess :bp1 ;
  bpmo:hasName "Order recieved"^^xsd:string ;
  ...
:e2 a bpmo:IntermediateEvent;
  bpmo:hasHomeProcess :bp1 ;
  bpmo:hasName "Invoice posted"^^xsd:string ;
  ...
...
```

3.5 Neighborhood Context Fragment Ontology

In this section, we define the *Neighborhood Context Fragment Ontology* (NCFO) as an extension of BPMO to semantically represent fragments of a BP model (Section 3.5.1). Some concepts in NCFO are inspired from our previous work [112]. Furthermore, we show how to populate fragments of semantic BP models using NCFO in the knowledge base (Section 3.5.2).

3.5.1 Neighborhood Conext Fragment Ontology (NCFO)

We represent a BP fragment by a graph, called neighborhood context graph [112], in which one task (namely root task) is located at the center. Its neighbors (i.e., directly connected tasks) are located in layers according to their shortest path lengths (i.e., consecutive nodes connected by an arc) to the root task. This graph represents the interaction of the root task to its neighbors. Based on this graph representation of

[1]The Resource Description Framework (RDF) is a World Wide Web Consortium (W3C) data model for web resources [160]

BP fragments, we propose an approach to retrieve similar BP fragment for recommendation by selecting a root task and its neighborhood tasks in a process model. Figure. 3.5 depicts NCFO concepts and their properties for describing a BP fragment as a neighborhood context graph.

- **ncfo:NeighborhoodContextGraph** represents a fragment of a semantic BP model that contains a root task and connections to its neighbors. Concretely, the concept *ncfo:NeighborhoodContextGraph* is represented by a root task through the relation *ncfo:hasRoot* to the concept *bpmo:Task* of BPMO and consists of multiple *ncfo:ConnectionFlow* defined by the relation *ncfo:hasConnectionFlow*.

  For example, Figure. 3.8 shows a neighborhood context graph retrieved from BP_1 (Figure. 3.6) having t_3 as a root task and another one retrieved from BP_2 (Figure. 3.6) having t_7 as a root task. The two neighborhood context graphs are annotated with the instances g_1 and g_2, respectively from the concept *ncfo:NeighborhoodContextGraph* of NCFO to represent BP fragments in the knowledge base.

- **ncfo:NeighborhoodElement** represents a workflow element in a neighborhood context graph. We consider tasks, start and end events having a connection path to the root task of the neighborhood context graph and the root task itself as neighborhood elements. Therefore, we model *ncfo:NeighborhoodElement* as a super-concept of *bpmo:Task*, *bpmo:StartEvent* or *bpmo:EndEvent* of BPMO. Furthermore, the data property *ncfo:atLayer* is used to indicate the shortest connection path distance from the root task, called layer number.

  For example, in Figure. 3.8, $t_1, t_2, t_3, t_4, e_1, e_4, e_6$ are neighborhood elements of g_1 and t_5, t_6, t_7, e_9, e_1 are neighborhood elements of g_2 by their annotation to the *ncfo:NeighborhoodElement* of NCFO.

- **ncfo:nextRelation** represents a object property from a *bpmo:NeighborhoodElement* to another *bpmo:NeighborhoodElement*. *ncfo:nextRelation* can be inferred an assertion with the inference rule to enrich the knowledge base (i.e., when a control flow connector s has as source a and as target b, we can infer that a and b are connected through s). The inference rule in Listing 3.2 using SWRL rules[2] to refer *ncfo:nextRelation* between a and b from *bpmo:ControlflowConnector* s having *bpmo:hasSource* a and *bpmo:hasTarget* b.

Listing 3.2: SWRL rule for nextRelation assertion

```
1 bpmo:ControlflowConnector(?s) ∧ bpmo:hasSource(?s,?a)
2 ∧ bpmo:hasTarget(?s,?b) ⟶ ncfo:nextRelation(?a,?b)
```

[2]SWRL is a language expressing rules for Semantic Web [161]

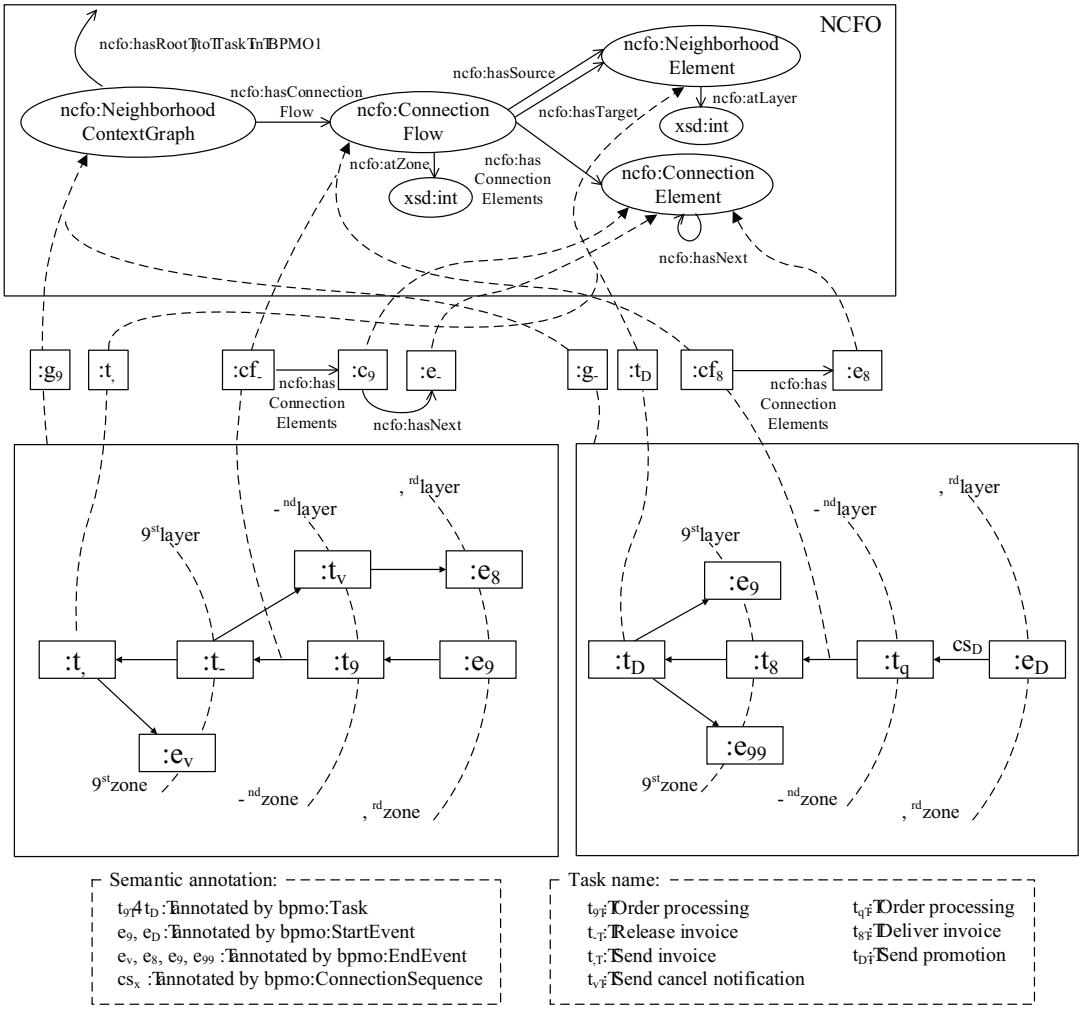


Figure 3.8: Example: neighborhood context graph

- **ncfo:ConnectionFlow** represents a connection path from a neighborhood element to its closest neighbor. It consists of a source and a target neighborhood element specified through properties *ncfo:hasSource* and *ncfo:hasTarget*. The data property *ncfo:atZone* shows its distance (namely zone number) from the root task determined through the number of layers between the source and the target neighborhood elements through the property *ncfo:atLayer*. Furthermore, it has a sequence of connection sequences from the source to the target specified by the property *ncfo:hasConnectionElements*.

  For example, in Figure. 3.6, the connection sequence between task t_1 and t_2 in BP_1 is formed by a sequence of c_1 and e_2, while the connection sequence between task t_6 and t_7 in BP_2 is one connection element e_8. The two connec-

tion sequences are represented in the neighborhood context graphs g_1 and g_2 (Figure. 3.8) as the instances cf_2 and cf_8, respectively, which are instantiated from the concept *ncfo:ConnectionFlow*.

3.5.2 Neighborhood Conext Fragment Annotation

Based on NCFO, to generate the neighborhood context graph from a selected root task, we first determine the shortest connection path from a root task to its neighborhood elements. This undirected path is represented as a sequence of neighborhood elements. It is retrieved by the recursive procedure presented in Algorithm 2. In this procedure, a and b are neighborhood elements, CP is a sequence of neighborhood elements and *isRelation* (line 2), *getRelation* (line 6) procedures are implemented by the execution of SPARQL queries from Listing 3.3 and 3.4, respectively. We check if a and b has direct relation (line 1), then we add b to CP and return CP. Otherwise, we get a set of neighborhood elements that has an undirected next relation from a as X (line 6), then for each neighborhood elements, we recursively call this procedure to continue obtaining its next neighborhood element.

Algorithm 2 Algorithm for retrieving shortest connection paths

```
1: procedure getConnectionPath(a,b,CP)
2:     if isRelation(a, b) then
3:         CP add b
4:         return CP
5:     else
6:         X in getRelation(a)
7:         for each x in X do
8:             getConnectionPath(a,b,CP)
9:         end for
10:    end if
11: end procedure
```

Listing 3.3: SPARQL query for verifying undirected relation

```
1 ASK {
2   :a ncfo:nextRelation :b || :b ncfo:nextRelation :a .
3 }
```

Listing 3.4: SPARQL query for retrieving undirected relation elements

```
1 SELECT ?x
2 WHERE {
3   :a ncfo:nextRelation ?x .
4   ?x ncfo:nextRelation :a .
5 }
```

For example, in BP_1 (Figure. 3.6), the shortest connection path from t_1 to t_3 is the sequence t_1, t_2 and t_3. The shortest connection path from t_3 to t_4 is the sequence t_3, t_2 and t_4.

Algorithms 3 shows how to generate connections flows from a shortest connection path CP. We iterate neighborhood elements in CP (line 3), then get the current and next neighborhood elements (line 4 and 5) and their layer number (line 6 and 7). We retrieve a connection sequence cs between these two neighborhood elements (line 8) and the number of zones (line 9). Thereafter, we instantiate a new connection flow cf (line 10) and add it to a set of connection flow CF. Finally, we return CF (line 14).

Algorithm 3 Algorithm for creating connection flow

```
1:  procedure createConnectionFlow(CP)
2:      i ← 1
3:      for i < size of CP do
4:          source ← CP[i − 1]
5:          target ← CP[i]
6:          source.atLayer ← i − 1
7:          target.atLayer ← i
8:          cs ← getConnectionSequence(source, target)
9:          zone ← i
10:         cf ← instatiateCF(source, target, cs, zone)
11:         add cf to CF
12:         i ← i + 1
13:     end for
14:     return CF
15: end procedure
```

For example, Figure. 3.9 constructed from BP_1 (Figure. 3.6) using Algorithm 3 shows examples of connection flows from t_3 to t_2 and from t_2 to t_1 having t_3 as a root task in BP_1 (see Figure. 3.6).

In this example, neighbors are located on a circle whose center is the root task with the number of layers as radius. Figure. 3.8 shows examples of neighborhood context graphs of t_3 and t_7 from BP_1 and BP_2 (Figure. 3.6), respectively. Thus, NCFO is used to populate fragments of semantic BP models in the knowledge base. An example of neighborhood context graph of t_3 from BP_1 (Figure. 3.6) modeled in NCFO using a triplet notation is given in Listing 3.5.

Listing 3.5: A process fragment modeled with NCFO

```
1 :g1 a ncfo:NeighborhoodContextGraph ;
2   ncfo:hasRoot :t3 ;
3   ncfo:hasConnectionFlow :cf1 ;
4   ncfo:hasConnectionFlow :cf2 ;
```

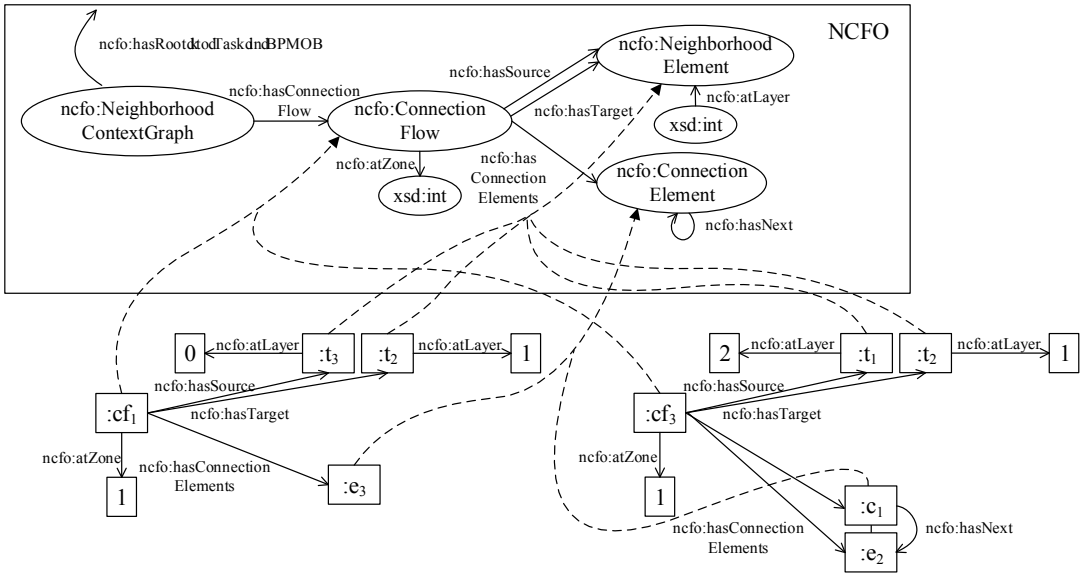


Figure 3.9: Example: Connection flows

```
5   ...
6  :cf5 a ncfo:ConnectionFlow ;
7    ncfo:hasSource :t2;
8    ncfo:hasTarget :t4;
9    ncfo:atZone "2"^^xsd:integer ;
10   ncfo:hasConnectionElement e2 ;
11   ncfo:hasConnectionElement c1 ;
12   ncfo:hasConnectionElement e5 ;
13 ...
14 :t2 a ncfo:NeighborhoodElement ;
15   ncfo:atLayer "1"^^xsd:integer ;
16 ...
17 :t4 a ncfo:NeighborhoodElement ;
18   ncfo:atLayer "2"^^xsd:integer ;
19 ...
20 :e2 ncfo:hasNext :c1 ;
21 :c1 ncfo:hasNext :e5 ;
22 ...
```

3.6 Neighborhood Context Graph Matching

In order to assist the design of BP variants, we propose to recommend similar neighborhood context graphs (i.e., BP fragments) populated in our knowledge base to an under design BP model. To do so, we propose a semantic-based approach to identify semantically similar BP fragments populated in a knowledge base for recommenda-

tions. Concretely, we measure the matching degree (i.e., similarity of tasks and their relations) between two BP fragments. To do so, firstly, we consider the matching between the elements of a BP fragments annotated by a common domain-specific ontology and we introduce the domain equivalence metric for matching them (Section 3.6.1). Thereafter, we describe how to compute the matching degree between two connection flows (Section 3.6.2). Finally, we compute the matching degree between two BP fragments by matching all connection flows that belong to the same zone (Section 3.6.3). We later on use such computed matching degrees for recommending similar BP fragments.

3.6.1 Neighborhood Element matching

In our approach, the element of a neighberhood context graph, called neighborhood element, can be a task, a start event, or an end event annotated by *bpmo:Task*, *bpmo:StartEvent*, and *bpmo:EndEvent*, respectively. For tasks, we consider that two tasks are similar, called domain equivalent, if they are annotated with the same concept from a domain-specific ontology (Definition 3.6.1).

Definition 3.6.1 (Domain equivalent $\equiv_d$)**.** *Let $a \in g_1$ and $b \in g_2$ be two tasks in two different neighborhood context graphs g_1 and g_2. a and b are domain equivalent, denoted as $a \equiv_d b$ if they are linked through an object property bpmo:hasBusinessDomain to the same concept of a domain-specific ontology.*

The SPARQL query to verify the domain equivalence is provided in Listing 3.6. Given task a and b are linked through an object property *bpmo:hasBusinessDomain* x and y, respectively, the query returns true if x is equivalent to y.

Listing 3.6: SPARQL query for verifying domain equivalence

```
1 ASK {
2   :a rdf:type bpmo:Task .
3   :a bpmo:hasBusinessDomain ?x .
4   :b rdf:type bpmo:Task .
5   :b bpmo:hasBusinessDomain ?y .
6   FILTER ( ?x = ?y )
7 }
```

For example, considering the two BP graphs BP_1 and BP_2 of Figure. 3.7, $t_1 \equiv_d t_5$ because they are connected through the *bpmo:hasBusinessDomain* object property to the same concept *bpmo:order_processing_with_reference* of a domain-specific ontology. We also have $t_6 \equiv_d t_2$ as they are related to the same concept *d:release invoice* through *bpmo:hasBusinessDomain* property.

In the same way, examine the domain equivalence of events in BP graphs, we consider two events are similar if they are annotated with the same meta-model concept from BPMO (see Definition 3.6.2).

Definition 3.6.2 (Meta-model equivalent $\equiv_m$)**.** *Let task $a \in g_1$ and task $b \in g_2$ be two neighborhood elements in two different neighborhood context graphs. a and b are meta-model equivalent, denoted as $a \equiv_m b$, if they are annotated with the same meta-model concept from BPMO.*

For example in Figure. 3.6, $e_1 \equiv_m e_7$ because they are annotated with the same meta-model concept 'StartEvent' from BPMO.

Based on these two definitions we consider neighborhood elements to be equivalent by Definition 3.6.3

Definition 3.6.3 (Neighborhood element equivalent $\equiv_n$)**.** *Let $a \in g_1$ and $b \in g_2$ be two neighborhood element in two different neighborhood context graphs. a and b are equivalent, denoted as $a \equiv_n b$ if $a \equiv_d b$ when both a and b are tasks.*

3.6.2 Neighborhood Connection Flow matching

Since each connection flow represents a sequence of connection elements, it can be easily mapped to a sequence of characters formed by their annotating BPMO concepts. We propose to use the Levenshtein distance [90] to compute the matching degree between two connection flows. It computes the minimum number of edits (i.e. insertions, deletions or substitutions) required to change one character sequence to the other. Given two connection sequences of connection elements cs_1 and cs_2, the matching $M(cs_1, cs_2)$ between them is computed by Equation. 3.1 where $size$ is a function returning the number of connection elements in a connection sequence, LD is a function returning the Levenshtein distance between two connection sequences, and the Max function returns the maximum number.

$$M(cs_1, cs_2) = 1 - \frac{LD(cs_1, cs_2)}{Max(size(cs_1), size(cs_2))} \tag{3.1}$$

For example, we show how to compute the matching between connection flows cf_2 and cf_8 of two neighborhood context graphs g_1 and g_2, respectively (Figure. 3.8) as follows:

$$\begin{aligned} M(cf_2, cf_8) &= 1 - (LD(\texttt{IntermediateEvent ParallelSplit}, \\ &\qquad \texttt{IntermediateEvent})/Max(2,1)) \\ &= 1 - (1/2) = 0.5 \end{aligned}$$

Levenshtein distance LD between "*IntermediateEvent ParallelSplit*" and "*IntermediateEvent*" is 0.5 as we consider one connection element as one character. Therefore, the minimum number of edits required to change between them is 1 by removing "*IntermediateEvent*".

3.6.3 Neighborhood Context matching

To compute the neighborhood context matching degree between two fragments, we compute the matching degree of the connection flows of the two neighborhood context graphs. We consider two cases: matching in the first zone and matching in other zones. In the first zone, we match the connection flows of the root tasks and the domain equivalent neighborhood elements ($\equiv_d$). In other zones, we match the connection flows that connect domain equivalent neighborhood elements. The matching degree between connection flows is computed using Equation 3.1. To compute the neighberhood context matching degree, we sum all matching values then divide them by the number of connection flows in the considered zones. Concretely, the matching between two neighborhood context graphs g_1 and g_2 within k zones, denoted by $\mathbb{MC}^k(g_1, g_2)$, is computed by Equation 4.2.

$$\mathbb{MC}^k(g_1, g_2) = \frac{\sum_{t=1}^{k} \sum_{{}^{e_v}_{e_u}cf_{g_1} \in Z^t_{g_1}, {}^{e_n}_{e_m}cf_{g_2} \in Z^t_{g_2}} \mathsf{MF}^t({}^{e_v}_{e_u}cf_{g_1}, {}^{e_n}_{e_m}cf_{g_2})}{\sum_{t=1}^{k} |Z^t_{g_1}|} \tag{3.2}$$

where k is the number of considered zones, Z is the set of connection flows, ${}^{b}_{a}cf_g \in Z^t_g$ is a connection flow from neighborhood element a to b within t^{th} zone of the neighborhood context graph g, and $\mathsf{MF}^t({}^{e_v}_{e_u}cf_{g_1}, {}^{e_n}_{e_m}cf_{g_2})$ is the matching between two connection flows ${}^{e_v}_{e_u}cf_{g_1}$ and ${}^{e_n}_{e_m}cf_{g_2}$ in t^{th} zone where:

$$\mathsf{MF}^t({}^{e_v}_{e_u}cf_{g_1}, {}^{e_n}_{e_m}cf_{g_2}) = \begin{cases} M({}^{e_v}_{e_u}cf_{g_1}, {}^{e_n}_{e_m}cf_{g_2}), & \text{if } (t = 1, (e_u \equiv_n e_m) \\ & \vee (e_v \equiv_n e_n) \\ & \vee (e_u \equiv_n e_v \wedge e_m \equiv_n e_n)) \\ & \wedge (t \neq 1, (e_u \equiv_n e_m \wedge e_v \equiv_n e_n)) \\ 0 & \text{otherwise} \end{cases}$$

For example, the neighborhood context matching between the two neighborhood graphs g_1 and g_2 (Figure. 3.8) within 1 zone is computed by Equation 4.2 as follow:

$$\begin{aligned} \mathbb{MC}^1(g_1, g_2) &= (M({}^{t_3}_{t_2}cf_{g_1}, {}^{t_7}_{t_6}cf_{g_2}) + M({}^{e_4}_{t_3}cf_{g_1}, {}^{e_{11}}_{t_7}cf_{g_2})/(3) \\ &= (0.5 + 1)/3 = 0.5 \end{aligned}$$

Another example of the neighborhood context matching between the two neighborhood graphs g_1 and g_2 within 3 zones is as follow:

$$\begin{aligned} \mathbb{MC}^3(g_1, g_2) = (&M({}^{t_3}_{t_2}cf_{g_1}, {}^{t_7}_{t_6}cf_{g_2}) + M({}^{e_4}_{t_3}cf_{g_1}, {}^{e_{11}}_{t_7}cf_{g_2}) + \\ &M({}^{t_2}_{t_1}cf_{g_1}, {}^{t_6}_{t_5}cf_{g_2}) + M({}^{t_1}_{e_1}cf_{g_2}, {}^{t_5}_{e_7}cf_{g_2}))/(3 + 2 + 2) \\ = &(0.5 + 1 + 0.5 + 1)/7 = 0.428 \end{aligned}$$

3.7 Conclusion

In this chapter, we answered the research questions raised in the thesis problematic which are: *R1: How to identify BP fragments that are close to process designer interests from heterogeneous process data?* and *R2: How to model heterogeneous BP models to be shared in a common knowledge base?*.

A shared knowledge base can serve as a common repository within an organization. To model heterogeneous BP models in a knowledge base, we proposed to adopt an existing mature BPMO ontology developed by the european project SUPER. BPMO allows modeling BP models described in various BP modeling languages such as BPMN and EPC. We presented an algorithm to construct BP models described in BPMO from heterogeneous BP models. Hence, by using such algorithm, BP models can be translated from any BP modeling lanaguage to BPMO which enables the interoperability of BPs within an organization.

We also proposed the NCFO ontology to represent fragments of BP models extracted from our knowledge base. We developed NCFO as an extension of BPMO to describe a BP fragment around a selected task as graphs. This graph represents the task (i.e., the root task) and the relations to its neighbors (i.e., tasks having relation to the root task). Based on this representation, we proposed an approach to retrieve BP fragments that are similar to a selected one in an under-design BP model. Such retrieved BP fragments can be used for BP variant design.

The approach we present in this section respects our principles introduced in Section 1.4.1:

- **Heterogeneous data modeling**: We propose to adopt an existing BPMO ontology to model heterogeneous BP models. Furthermore, we define an ontology as an extension of BPMO, namely NCFO, to model BP fragments. Some concepts in NCFO are inspired from our previous work [112]. However, the main difference is the use of semantics in our work.

- **Automation**: We propose an automated approach to construct a knowledge base from heterogeneous BP models. Moreover, similar BP fragments can be automatically retrieved from our knowledge base to facilitate the design of BP variants.

- **Implicit knowledge exploitation**: We utilize existing and accessible BP models shared within an organization.

- **Focused results**: We exploit the relation between activities to measure the similarity matrices between BP fragments. Such relation, so called neighborhood context, is implicit knowledge hidden in BP models to represent our BP fragments.

We validate our approach by demonstrating our proof of concept application to support the BP variant design by recommending appropriate BP fragments in Section 7.2.1. Thereafter, we evaluate our approach by performing several experiments to prove that our approach is feasible in real use cases in Section 7.3.1

Chapter 4

Assisting BP Variant Design by Exploiting Process Event Logs

Contents

4.1 Introduction

In this chapter, we address the research questions: *R1: How to identify BP fragments that are close to process designer interests from heterogeneous process data?* and *R3: How to model heterogeneous process event logs to be shared in a common knowledge base?* We presents our approach for assisting the design of BP variants by using on process event logs. As mentioned in Chapter 1, BP models do not always exist in information systems. However, typical systems always produce and accumulate process event logs which contain useful information related to the important business execution paths. We start the chapter by introducing an illustrating example and overview of our approach (Section 4.2). In section 4.4, we present our ontologies and algorithms to discover BP models from heterogeneous process event logs. In section 4.4.2, we

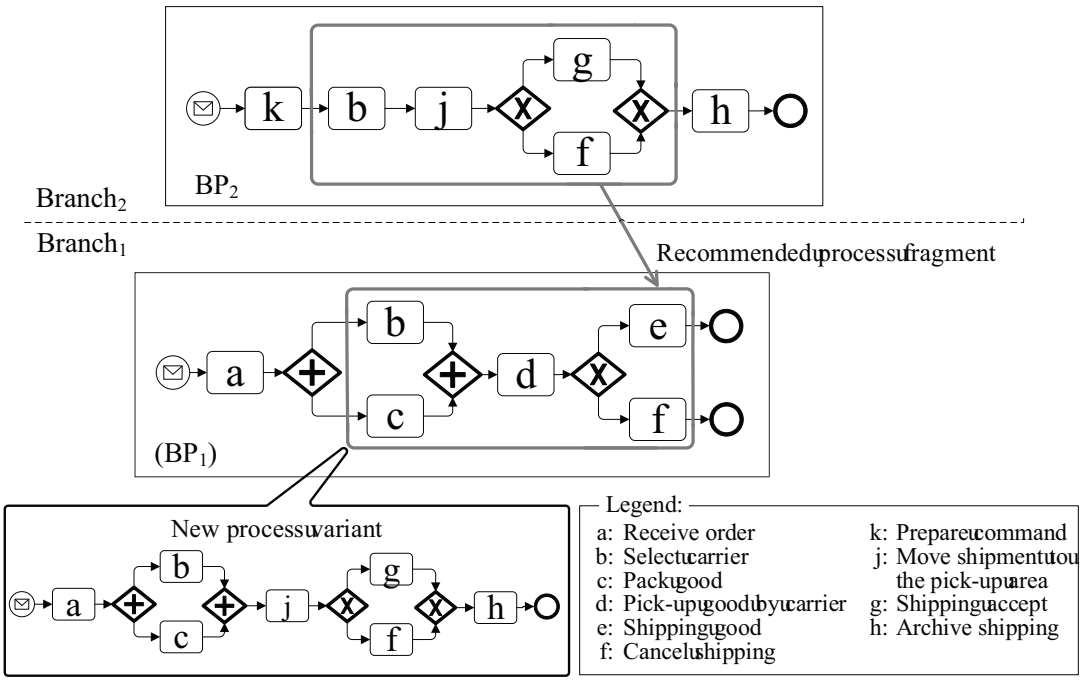


Figure 4.1: Scenario of BP variant development

detail the semantic representation of process event logs in our knowledge base. Section 4.5 continues with the matching of BP fragments for recommendation to assist the design of BP variants. Finally, we conclude the chapter in Section 4.6.

The work in this chapter was published in conference proceedings [162].

4.2 Illustrating example

We extend our example presented in Section 3.2 to illustrate our approach. Figure 4.1 shows that an organization have two branches $branch_1$ and $branch_2$ have developed and executed their BP variants BP_1 and BP_2, respectively. Upon instantiation of BP_1, the order data is first stored (task a). Thereafter, the following concurrent activities take place: (b) a carrier is automatically selected based on the order data, (c) the good is manually packed by a staff person, (d) the good is picked up by the selected carrier. Finally, (e) the good is either shipped or (f) refused to ship by the selected carrier. BP_1 provider may need to design a new variant of his BP to support more prospective customers' needs. Examples of needs could be: move goods to the pick-up area, archive details related to the shipping at the end of the BP, etc.

Suppose that $branch_1$ needs to adjust BP_1 to support more requirements. To do so, the process designer may look for appropriate BP fragments that can be flexibly plugged in the selected position. A recommendation system can assist such task by

recommending that the fragment of BP_2 including activities b, j, g and f at $branch_2$ is similar to the selected fragment of BP_1. This recommendation allows process designer to easily get new ideas for designing new BP variants of BP_1 as shown in the figure.

We consider a use case where organization's branches are willing to share their process event logs. Therefore, we propose to recommend BP fragments based on similarity between fragment of process event logs. However, an issue that prevent fostering the use of such data between different organization's branches is their heterogeneity (syntactic or semantic). In this work, we assume that process data have already been preprocessed to solve the syntactic problem, thus we focus only on semantic level. Organization's branches execute similar BPs, but accumulate heterogeneous process event logs (see examples of BP_1 and BP_2 process event logs in Table 4.1 and Table 4.2 respectively). We need to resolve this heterogeneity issue in order to benefit from such logs.

Order ID	Task	Timestamp	Product	Quantity
8801	Receive order (a)	03-07-2015@12.00	Printer	100
8802	Receive order (a)	03-07-2015@12.30	Scanner	50
8801	Select carrier (b)	03-07-2015@16.00	Printer	100
8803	Receive order (a)	03-07-2015@16.30	Monitor	200
8802	Pack good(c)	04-07-2015@09.00	Scanner	50
...	...	...	...	...

Table 4.1: Excerpt of BP_1 process event logs

Order number	Activity	DateTime	Item	No.
101	Prepare command (k)	13:00, 3 July 2015	Screen	20
101	Select carrier (b)	13:00, 4 July 2015	Screen	20
102	Prepare command (k)	11:00, 5 July 2015	Router	100
101	Move shipment... (j)	13:30, 5 July 2015	Router	100
103	Prepare command (k)	16:00, 5 July 2015	Scanner	20
...	...	...	...	...

Table 4.2: Excerpt of BP_2 process event logs

4.2.1 Approach overview

Our approach aims to solving interoperability issues of heterogeneous process event logs by proposing a semantic framework as depicted in Figure 4.2. Concretely, our approach includes four main steps:

1. Discover BP models from heterogeneous process event logs, and thus populate a shared knowledge base of such models (Section 4.4). Our knowledge base

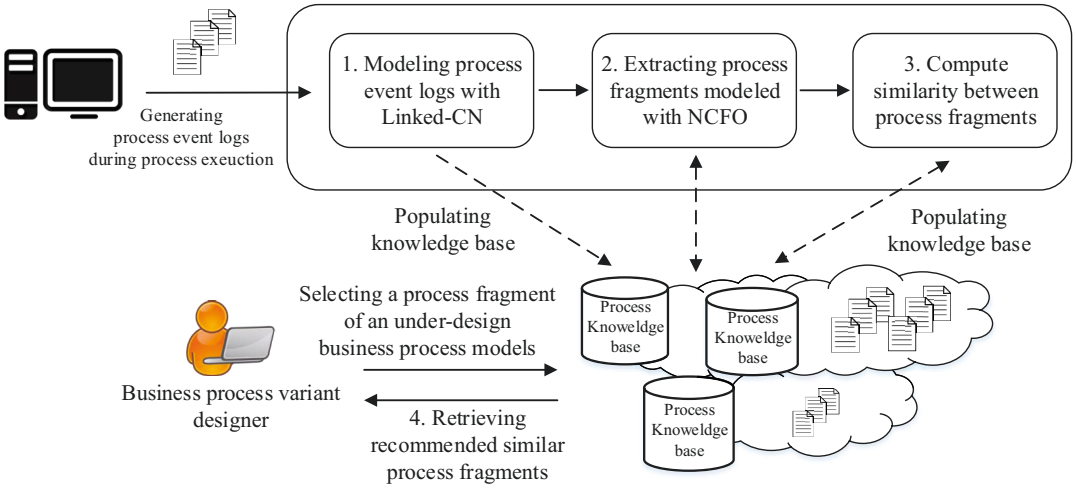


Figure 4.2: An overview of our approach

serves as a common repository for discovered BP models. We populate our knowledge base based upon our proposed ontology, namely Linked-CN, which is an extension of mature ontologies describing BP models and process event logs from European project SUPER [57].

2. Retrieve fragments of discovered BP models from the shared knowledge base as neighborhood context graphs (Section 4.4.2). We extend our NCFO ontology (Section 3.5) to describe such fragments. Thereafter, we compute the similarity between BP fragments for recommendation.

3. Compute the matching between neighborhood context graphs (Section 4.5). This matching presents the similarity between two fragments in term of relations between activities' execution.

4. Lastly, a process designer may select a BP fragment of an under-design process. Then, we retrieve and sort similar BP fragments to the selected one in descending order of similarity and retrieve top-n fragment for recommendation.

4.3 Preliminaries

To cope with the heterogeneity problem between process event logs between organization's branches, we define an ontology based on Linked Data principles [159], namely Linked Causal Net (*Linked-CN*). We develop Linked-CR as an extension of existing ontologies of European project SUPER and our ontology NCFO (Section 3.5). In *Linked-CN*, we adopt C-net modeling language to represent discovered BP models from process event logs. Hence, we firstly briefly describe C-net in this section.

Thereafter, we introduce *Linked-CN* in the next section.

4.3.1 Causal Net (C-net)

C-net [63] is a BP modeling language representing concurrency of activities and their causal dependencies, i.e., relationships between activities where one is triggering the other one. C-net is a dependency graph where nodes represent activities, and arcs correspond to causal dependencies. Input and output bindings denote the occurrence of task. Each arc is annotated with a set of input and output bindings, represented by black dots. A set of dots connected on the input (resp. output) arcs of a node is an input (resp. output) binding. Input and output bindings allow modeling typical logical connectors in various modeling languages such as, BPMN, EPC, YAWL, etc (Figure 4.3(a)). Nodes can be annotated with numbers [63] which represent occurrences of activities and reflect the importance of a task (Figure 4.3(b)). Arcs can be annotated with frequencies of bindings which show the strength of a relations between activities [28].

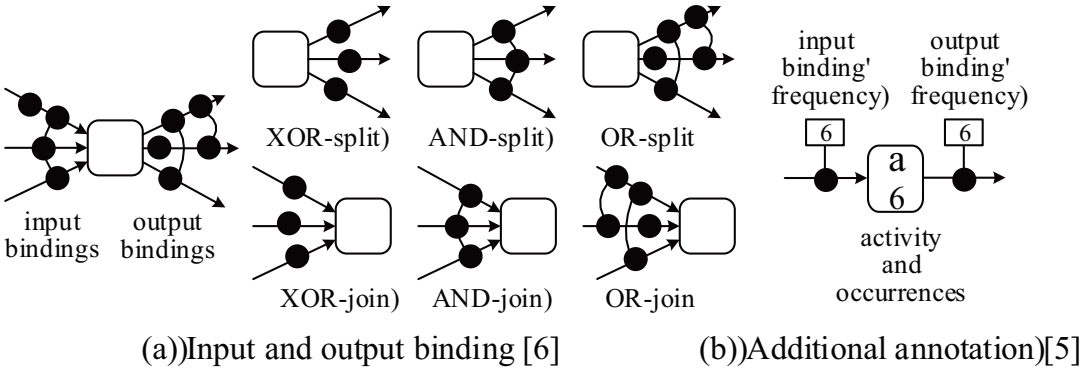
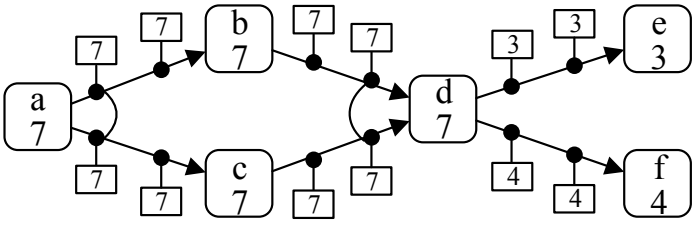


Figure 4.3: C-net modeling language

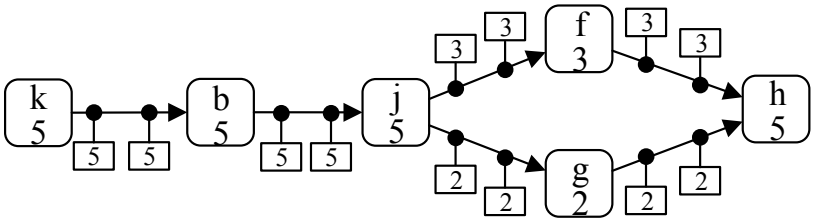
In our approach, we use C-nets to model discovered BP from process event logs because: (*i*) they can be discovered using existing process mining languages since they use similar representations, (*ii*) they allow to model XOR, AND and OR logical connectors and hence they fit well with various BP modeling languages [28], and (*iii*) they consider occurrence of activities unlike other BP modeling languages (e.g. EPC, BPMN, UML).

To illustrate C-nets, we consider BP_1 from Fig. 4.1 and a process event log collected after several executions (An excerpt is provided in Table 4.1). Nodes and arcs of the C-net represented in Fig. 4.4(a) are discovered by applying the heuristic mining technique introducted in [62]. The nodes of this C-net represent the activities *a*, *b*, *c*, *d*, *e* and *f* extracted from the process event log and annotated with their number of occurrence from the log (e.g. *a* occured 7 times). The arcs of this C-net represent the relations between activities. For example, the arc from *a* to *b* represents

a precedence dependency (i.e. a is executed before b). Input and output bindings represent split and join logical connectors, respectively. Arcs are annotated by input and output bindings which are discovered by applying a process mining technique, such as heuristics using time window before and after each task [28]. For example, in BP_1 a has one output binding therefore there is only one possibility that both b and c (i.e., AND-split) occur 7 times after a. On the other hand, d has two output bindings therefore e or f (i.e., XOR-split) occur 3 or 4 times after d, receptively.



(a) C-net of BP_1



(b) C-net of BP_2

Figure 4.4: C-nets

4.4 Knowledge Base of Process Event Logs

To enable the interoperability between different organization's branches at the log level, we propose to construct a knowledge base containing C-nets discovered from process event logs. Such C-nets are represented using the *Linked-CN* ontology (Section 4.4.1). We reuse the *NCFO* ontology (described in Section 3.5) to represent a C-net fragment (Section 3.5.1), thus promoting similarity computation between BP fragments.

4.4.1 Linked Causal Net (*Linked-CN*)

Fig. 4.5 depicts our proposed ontology *Linked-CN* which allows representing process event logs and C-nets discovered from these logs. *Linked-CN* offers the following concepts and properties:

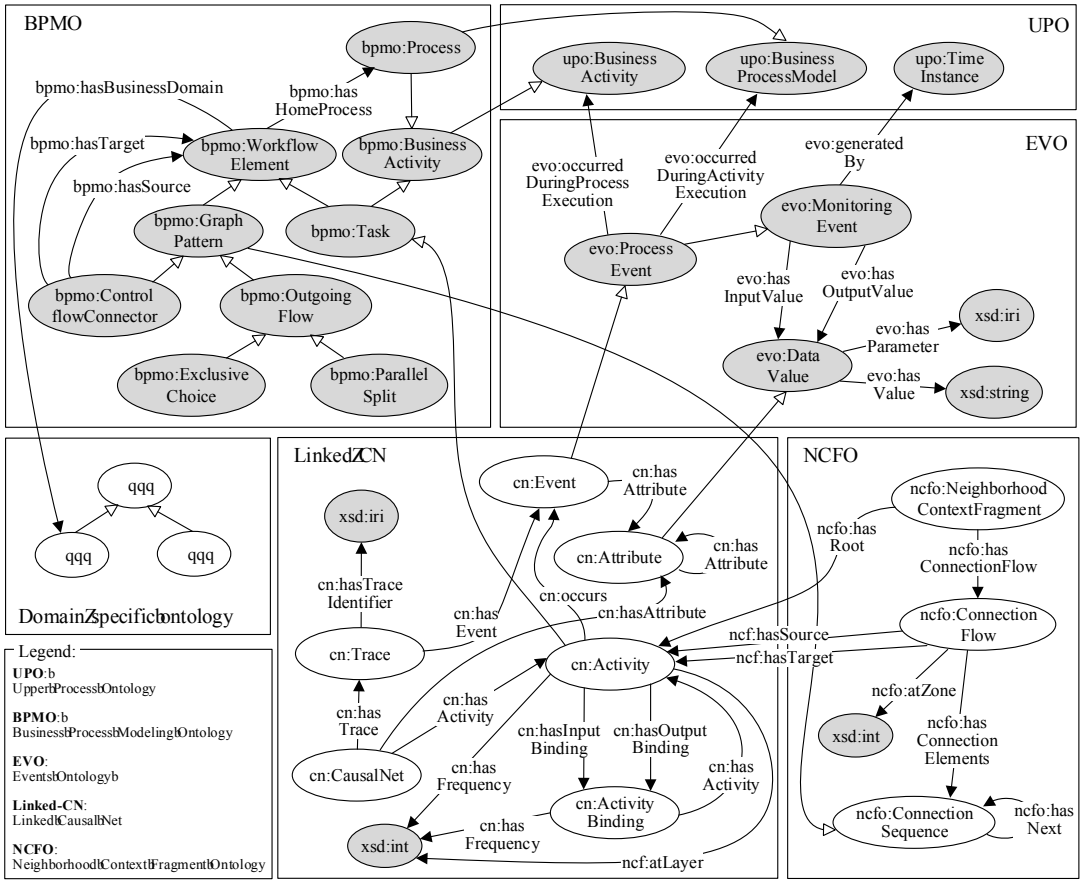


Figure 4.5: SUPER ontologies, *NCFO* and *Linked-CN*

- **cn:Event** concept represents an event in a process event log. Inspired by the extensibility and flexibility of the XES [163], we define cn:Event as a refinement of evo:ProcessEvent with an object property cn:hasAttribute to describe generic event attributes. For example, an event representing the execution of *Receive order* (*a*) (line 1 in Table 4.1) is modeled in *Linked-CN* as shown in Listing 4.1 (line 1-4). Such event is used to populate a BP execution knowledge base (In this work, we represent the knowledge base using Turtle RDF triple).

- **cn:Attribute** concept is a refinement of the evo:DataValue concept. Based on XES, we define a reflexive object property cn:hasAttribute in order to describe nested attributes (i.e. attributes themselves might have child attributes).

- **cn:Activity** concept is a refinement of the bpmo:BusinessActivity concept. Different events might occur during an activity execution and are recorded in process event logs. Therefore, we define an object property cn:occurs to describe events

Listing 4.1: BP_1 execution events modeled with *Linked-CN*

```
:8801_a a cn:Event ;
  evo:hasCreationTimestamp :03-07-2015@12.00 ;
  cn:hasAttribute :order_number_8801 ,
  ...
:03-07-2015@12.00 a upo:TimeInstance ;
  upo:yearOf "2015"^^xsd:integer ,
  upo:monthOf "07"^^xsd:integer ,
  ...
:order_number_8801 a evo:DataValue ;
  evo:hasParameter "www.example.com/order_id"
  ^^xsd:iri ;
  evo:hasValue "8801"^^xsd:string ;
...
```

occurring during an activity execution and a data type property cn:hasFrequency to store its number of occurrence. Activities may have input and output bindings represented by the cn:hasActivityInputBinding and cn:hasActivityOutputBinding object properties, respectively. Hence, activities with no input or output bindings are considered as start or end activity, respectively. For example, events in Table 4.1 are described by activities that triggered them using *Linked-CN* (Listing. 4.5, lines 4-13).

- **cn:Trace** concept represents an ordered sequence of activities. We consider events having identical specific attribute (e.g, *Order ID*) as belonging to the same trace. Therefore, we define the data type property cn:hasTraceIdentifier to associate an attribute (e.g., IRI) to a trace. We also define the object property cn:hasEvents to represent events belonging to a trace. Algorithm 4 depicts how to retrieve traces from our knowledge base given a trace attribute ($traceIRI$) and an activity attribute ($activityIRI$). First, all trace identifiers, traceIDs are retrieved (line 2). This done by executing the SPARQL query *getTraceIDs*. Afterwards, for every *traceID* in *traceIDs* (lines 3-6), we execute the *getActivitySequence* SPARQL query to retrieve the sequence of activities (*actSeq*) from $traceID$, $traceIRI$ and $activityIRI$. This sequence will be added to a list *traces* (Line 5). Finally, the algorithm returns traces as output (Line 7). For example, Listing 4.4 illustrates traces retrieved from events modeled with *Link-CN* in Listing 4.1.

- **cn:CausalNet** concept represents discovered C-net from process event logs. Fig. 4.4 depicts examples of C-nets built from traces of BP_1 (Listing 4.4) and BP_2. We associate to cn:CausalNet three object properties: cn:hasTrace representing the traces used to discover the C-net, cn:hasActivity representing the activities in the C-net and cn:hasAttribute representing global attribute, i.e. an attribute that is

Listing 4.2: *getTraceIDs* SPARQL query

```
SELECT ?traceIDs WHERE {
  ?event cn:hasDataValue ?dataValue .
  ?dataValue evo:hasParameter :traceIRI ;
  evo:hasValue ?traceIDs . } GROUP BY ?traceIDs
```

Listing 4.3: *getActivitySequence* SPARQL query

```
SELECT ?activity WHERE {
  ?event cn:hasDataValue ?dataValue .
  ?dataValue evo:hasParameter :traceIRI ;
  evo:hasValue :traceID .
  ?dataValue evo:hasParameter :activityIRI ;
  evo:hasValue ?activity .
  ?event evo:hasCreationTimestamp ?timestamp .
  OPTIONAL { ?timestamp upo:yearOf ?year } .
  OPTIONAL { ?timestamp upo:monthOf ?month } .
  OPTIONAL { ?timestamp upo:dayOf ?day } . ... }
ORDER BY ?year ?month ?day ...
```

Listing 4.4: Traces of BP_1 process event logs modeled with *Linked-CN*

```
:8801 a cn:Trace ;
  cn:hasTraceIdentifier "www.example.com/order_id" ;
  cn:hasEvent :8801_a ,
  cn:hasEvent :8801_b ,
  cn:hasEvent :8801_c ,
  cn:hasEvent :8801_d ,
  cn:hasEvent :8801_e ,
:8802 a cn:Trace ;
  cn:hasTraceIdentifier "www.example.com/order_id" ;
  cn:hasEvent :8801_a ,
  cn:hasEvent :8801_c ,
  cn:hasEvent :8801_b ,
  cn:hasEvent :8801_d ,
  cn:hasEvent :8801_e ,
...
```

Algorithm 4 Retrieving traces

```
1: procedure getTraces(traceIRI)
2:     traceIDs ← getTraceIDs(traceIRI)
3:     for each traceID in traceIDs do
4:         actSeq ← getActivitySequence(traceID, traceIRI, activityIRI)
5:         traces ← traces.add(actSeq)
6:     end for
7:     return traces
8: end procedure
```

Listing 4.5: BP_1 activity binding modeled with *Linked-CN*

```
1 :c_net a cn:CausalNet ;
2   cn:hasActivitiy :a ,
3   ...
4 :a a cn:Activity ;
5   cn:hasFrequency "7"^^xsd:integer ,
6   cn:occurs :8801_a ,
7   cn:occurs :8802_a ,
8   ...
9   cn:hasActivityOutputBinding :a_binding1 ;
10 :a_binding1 a cn:ActivityBinding ;
11   cn:hasFrequency "7"^^xsd:integer ,
12   cn:hasActivity b ,
13   cn:hasActivity c ;
14 ...
```

available for every event. Listing. 4.5 shows the C-net from Fig. 4.4(a) modeled with *Linked-CN*.

- **cn:ActivityBinding** concept describes causal dependency, i.e, relationship between activities where one is triggering the other one. The data type property cn:hasFrequency reflects the number of occurrence of an activity, and the cn:hasActivity object property represents activities that are part of the binding. An example of an activity binding is shown in Listing. 4.5 at lines 9-13.

4.4.2 Neighborhood Context Fragment Ontology (*NCFO*)

We have introduced our NCFO ontology in Section 3.5 to represent a fragment of BP models. *NCFO* is developed based up on a graph, namely neighborhood context graph, in which a selected activity (i.e., root activity) is located at the center. Its neighbors (i.e., directly connected tasks) are located in layers according to their shortest path lengths (i.e., consecutive nodes connected by an arc) to the root activity. Therefore, we extend our *NCFO* ontology in order to represent a C-net fragment.

C-net fragment can be retrieved from our knowledge base by selecting an activity and indicating the size of the fragment (i.e. zoneth number). The selected activity is considered as a root activity of the fragment. Its neighboring activities and their activity bindings are included in the fragment based on the zoneth number. Fragments of C-nets captured from BP_1 and BP_2 (Figure 4.4(a) and Figure 4.4(b)) having root activities a and k are shown in Figure 4.6(a) and Figure 4.4(b), respectively.

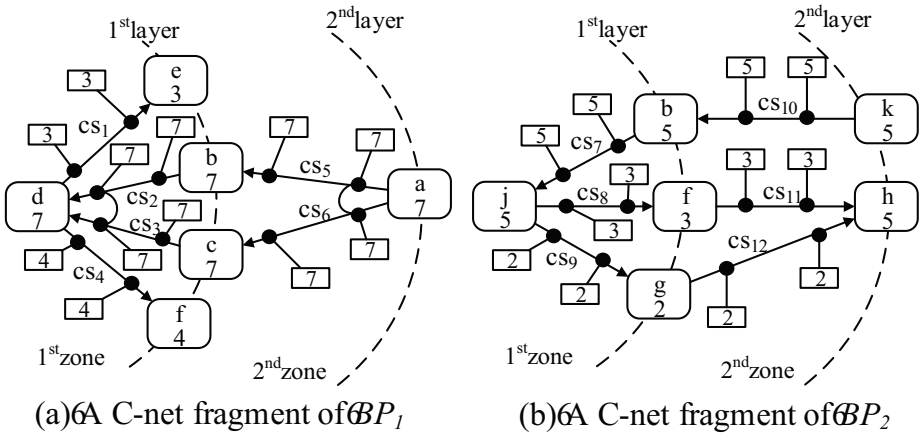


Figure 4.6: C-net fragments

Fig. 4.5 depicts our *NCFO* ontology offering the following concepts and associated properties:

- **ncf:NeighborhoodContextFragment** concept represents a fragment of a C-net that contains a root activity and connections to its neighbors. The fragment's root activity, of type cn:Activity, is defined through the object property ncf:hasRoot. An ncf:NeighborhoodContextFragment has also multiple ncf:ConenctionFlows defined using the ncf:hasConnectionFlow object property. Listing. 4.6 shows an example of a neighborhood context fragment of the C-net fragment from Fig. 4.6(a).

- **ncf:ConnectionFlow** concept represents a dependency relation from an activity to its closest neighbor. It consists of a source and a target activity defined through the object properties ncf:hasSouce and ncf:hasTarget. It has a sequence of ncf:ConnectionSequence defined through the object property ncf:hasConnectionElements. Furthermore, the data property ncf:atZone shows the maximum connection path distance (namely zone number) between source and target activities to the root activity. We extract connection flows from by finding the shortest path from a root activity to its neighborhood activity in C-net[1]. An example of a connection flow is depicted in Listing. 4.6 at lines 3-4 and 9-11.

[1]Due to the lack of space, we have described how to extract connection flows from C-net at: (`http://www-inf.it-sudparis.eu/SIMBAD/tools/semantic_cnet_fragment/connection_flow/`)

Listing 4.6: A neighborhood context fragment from Fig. 4.6(a)

```
1 :c_net_fragment1 a ncf:NeighborhoodContextFragment ,
2   ncf:hasRoot :d ,
3   ncf:hasConnectionFlow :cf1 ,
4   ncf:hasConnectionFlow :cf2 ,
5   ...
6 :d a cn:Activity ;
7   cn:hasFrequency "7"^^xsd:integer ,
8   ...
9 :cf2 a ncf:ConnectionFlow ;
10  ncf:hasSource :b ,
11  ncf:hasTarget :d ,
12  ncf:ConnectionSequence cs2 ;
13 :cs2 a bpmo:ControlflowConnector ;
14  ncf:hasNext :cs2_1
15 :cs2_1 a bpmo:ParallelSplit
16 ...
```

- **ncf:ConnectionSequence** concept represents a sequence of connection elements from one activity to another. Connection element is a logical connector which can be described by BPMO, such as bpmo:ExclusiveChoice, bpmo:ParallelSplit, etc. Thus, we represent the sequence of connection elements by using the ncf:hasNext object property. An example of a connection sequence is shown in Listing. 4.6 at lines 12-15.

4.5 Neighborhood Context Graph Matching

Using neighborhood context graphs associated to C-nets, we propose an approach to recommend similar BP fragments to assist the design of BP variants. This recommendation is based on similarity factors between the different connection flows in BP fragments. The similarity computation is similar to Section 3.6, the main difference is that we take into account the frequency of executed activities. In the following we present our recommendation approach using our formulas for connection flow matching, and neighberhood context matching in Section 4.5.1 and 4.5.2, respectively.

4.5.1 Connection flow matching

Two connection flows can be matched, only if they have identical activities at their beginning and end. In our work, we consider that activities are semantically annotated by concepts from a common domain-specific ontology through an object property *bpmo:hasBusinessDomain*. We identify whether two activities are identical or not based on these concepts (Definition 3.6.1).

For two connection flows, once activities at their beginning and end are domain

equivalent, the matching is done as follows. Each connection flow represents a sequence of connection elements, which can be considered as a sequence of characters by their BPMO concept. For example, a connection sequence cs_2 (Listing. 4.6 at line 12-15) has a sequence of two connection elements *bpmo:ControlflowConnector* (line 13) and *bpmo:ParallelSplit* (line 15), and will be mapped into "*ControlflowConnectorParallelSplit*". To compute the similarity between two connection flows, we use the *Levenshtein distance* [90]. The Levenshtein distance measures the minimum number of edits (i.e. insertions, deletions or substitutions) required to change one character sequence to the other. Concretely, given two character sequences s_{cf_1} and s_{cf_2} associated to two connection flows cf_1 and cf_2, the similarity factor between them is computed by Equation (4.1) where w is a function that returns the frequency of an activity binding and $size$ is a function that returns the number of connection elements in a connection sequence.

$$\begin{aligned} SF_{cf}(cf_1, cf_2) =& (1 - \frac{LevenshteinDistance(s_{cf_1}, s_{cf_2})}{max(size(cf_1), size(cf_2))}) \\ & \times \frac{w(cf_1) + w(cf_2)}{2 \times max(w(cf_1), w(cf_2))} \end{aligned} \tag{4.1}$$

For example, the similarity factor between cs_1 and cs_9 (Fig. 4.6) is computed as follows:

$$\begin{aligned} SF_{cf}(cs_1, cs_9) &= SF_{cf}(\text{“}ExclusiveChoiceControlflowConnector\text{”}, \\ &\quad \text{“}ExclusiveChoiceControlflowConnector\text{”}) \\ &= (1 - 0) \times (3 + 2)/(2 \times 3) = 0.83 \end{aligned}$$

4.5.2 Neighborhood context matching

To compute the neighborhood context matching between two BP fragments, we match the connection flows that connect identical activities belonging to the same zone number (zoneth). Such connection flows are called common connection flows (Definition 4.5.1)

Definition 4.5.1 (Common connection flows $\equiv_{cf}$)**.** *Let s_1, s_2 and t_1, t_2 be the source and target activities of the connection flows cf_1, cf_2 from neighborhood context graphs nc_1, nc_2, respectively. Let r_1 and r_2 be the root activities of nc_1 and nc_2, respectively. cf_1 and cf_2 are common connection flows, denoted as $cf_1 \equiv_{cf} cf_2$ if $s_1 \equiv_d s_2$, $t_1 \equiv_d t_2$ and cf_1 is at the same zoneth number as cf_2.*

In order to compute similarity between two neighborhood context fragments, we sum all connection flow matching values then divide them by the maximum number of connection flows in the considered zones. Concretely, the similarity factor between two neighborhood context fragments nc_1 and nc_2 within k zones, denoted by $SF^k(nc_1, nc_2)$, is computed using Equation (4.2) where cf_i are cf_j belong to nc_1 and

nc_2, respectively, $cf_i \equiv_{cf} cf_j$, n and m are the numbers of connection flows within the k zones of nc_1 and nc_2, respectively.

$$SF^k(nc_1, nc_2) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} SF_{cf}(cf_i, cf_j)}{max(n, m)} \tag{4.2}$$

For example, the neighborhood context matching between the fragments of activity d and j (i.e., nc_d and nc_j) within the first zone from Fig. 4.6(a) and Fig. 4.6(b), respectively, giving that $e \equiv_d g$, computed by Equation 4.2 is:

$$\begin{aligned} SF^1(nc_d, nc_j) =& \frac{SF_{cf}(cs_1, cs_9) + SF_{cf}(cs_2, cs_7) + SF_{cf}(cs_4, cs_8)}{max(4, 3)} \\ =& \frac{0.83 + 0.43 + 0.88}{4} = 0.54 \end{aligned}$$

4.6 Conclusion

In this chapter, we answered the question raised in the thesis problematic which are: *R1: How to identify BP fragments that are close to process designer interests from heterogeneous process data?* and *R3: How to model heterogeneous process event logs to be shared in a common knowledge base?*.

We model heterogeneous process event logs in a shared knowledge base by using our proposed ontology, namely *Linked-CN*. *Link-CN* is constructed by extending ontologies developed from European SUPER project (i.e., *BPMO*, *UPO*, and *EVO*). Using *Linked-CN* allows to model discovered BP models from heterogeneous process event logs which enables the interoperability of process event logs within an organization.

We reuse and extend our ontology *NCFO* to represent fragments of discovered BP models from process event logs. A BP fragment modeled with *NCFO* represents a root task and relations to its closest neighbors. We retrieved BP fragments that are similar to the selected one in an under-design PB model. Thereafter, such BP fragments can be used as a data-source to design BP variants.

Our principles presented in Section 1.4.1 are respected:

- **Heterogeneous data modeling**: We use our proposed *Linked-CR* ontology to model heterogeneous process event logs.
- **Automation**: We propose an automated approach to construct a knowledge base of BP models discovered from heterogeneous process event logs. Moreover, similar BP fragments can be automatically retrieved from our knowledge base to facilitate the design of BP variants.
- **Implicit knowledge exploitation**: We utilize existing process event logs shared within an organization.

- **Focused results**: We exploit the relation between activities to measure the similarity matrices between BP fragments. Such relation, so called neighborhood context, is implicit knowledge hidden in BP models to represent our BP fragments.

We demonstrate our approach by developing a proof of concept to support the BP variant design by recommending appropriate BP fragments in Section 7.2.1. Furthermore, we present the experimentations to validate our approach using real use-cases to prove that our approach is feasible in Section 7.3.2.

Chapter 5

Supporting Cloud Resource Descriptions Interoperability

Contents

5.1 Introduction

This chapter addresses the research question *R4: How to model heterogeneous cloud resources to be shared in a common knowledge base?* We start the chapter by introducing an illustrating example (Section 5.2). In section 5.3, we propose a semantic framework where we develop a knowledge base that enables the interoperability between heterogeneous cloud resource descriptions and provide means for customization. Finally, we conclude the chapter in Section 5.4.

The work in this chapter was published in conference proceedings [164].

5.2 Illustrating example

We present in the following a scenario (see Fig. 5.1) to illustrate and motivate our approach. Let us consider that an organization's branches $Branch_1$ and $Branch_2$ possess

their own cloud environments C_1 and C_2 offering different cloud resources: c_a, n_a, and s_a in C_1, and c_b and n_b in C_2. These resources consist of three type: *Compute*, *Network*, and *Storage*. *Compute* represents a compute resource that encapsulates both CPU and memory. *Network* represents a networking entity with the purpose of forwarding data traffic between end points. *Storage* is a resource that records information to a data storage device. *Link* and *interlink* represents a connection between internal and external resources, respectively. We consider only cloud resources at the infrastructure level in this scenario.

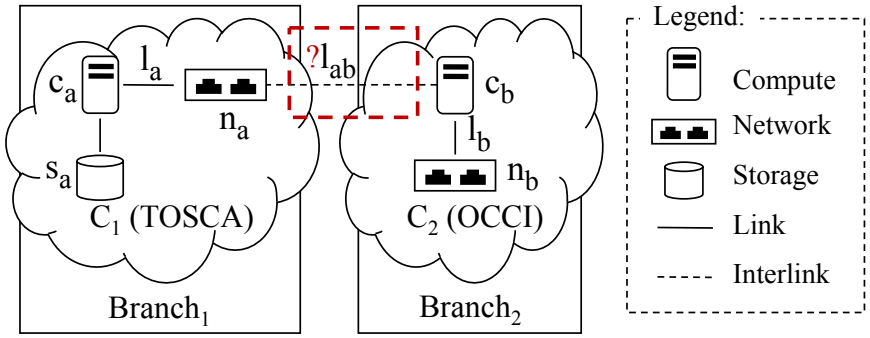


Figure 5.1: Scenario of cloud resource interoperability

In this scenario, we suppose that $Branch_1$ and $Branch_2$ are willing to enable interoperability between their cloud resources C_1 and C_2 in order to share the storage s_a. Since s_a and C_1 are connected (represented in the figure by link), and C_1 is connected to network resource n_a in C_1, thus C_2 in C_2 should be connected to n_a in order to access s_a. However, C_1 and C_2 are described using different cloud resource description standards. C_1 resources described are in TOSCA (see Listing 5.1) and C_2 described in OCCI (see Listing 5.2) using their respective standard specifications [131, 165]. Due to these heterogeneous descriptions, creating a connection between n_a and C_2 is not a straightforward task. It is required that cloud providers translate the description of n_a from TOSCA to OCCI, and then create a link l_{ab} from C_2 to n_a.

Manually translating cloud resources descriptions is a tedious and error-prone task because of the lack of a common schema and vocabulary. For example, the representation of the compute resource c_a in C_1 described in TOSCA in Listing 5.1 (line 2-8) is completely different from the compute resource c_b in C_2 described in OCCI in Listing 5.2 (line 1-5). In order to overcome this issue, we define a set of ontologies that represent cloud resources described by different standards (i.e., *standard-specific ontologies*) and an ontology (i.e., *Linked-CR*) that abstracts high-level representation of these standards. Based on these ontologies, we propose a semantic framework that populates a shared knowledge base of cloud resources described by different standards. Furthermore, Semantic Web technologies allows to query the knowledge base to look for available cloud resources regardless of their actual representation and

Listing 5.1: Excerpt from resources C_1 TOSCA description

```
node_templates:
  ca:
    type: Compute
      capabilities:
        host:
          properties:
            num_cpus: 1
            ...
  na:
    type: Network
    ...
  porta:
    type: Port
    requirements:
      - binding: ca
         relationship:
           type: LinksTo
      - link: na
         relationship:
           type: BindsTo
...
```

Listing 5.2: Excerpt from cloud resources C_2 OCCI description

```
Category: compute;
  scheme="http://.../occi/infrastructure#";
  class="kind";
X-OCCI-Attribute: occi.compute.hostname="cb"
X-OCCI-Attribute: occi.compute.cores=2
Link: <lb>;
  rel="http://.../occi/infrastructure#network";
  self="cb";
  category=".../infrastructure#networkinterface";
  ...
...
```

customize these resources in order to enable interoperability between organizations in a collaborative environment.

5.3 Semantic Framework of Cloud Resources

To enable the interoperability of cloud resources described by different standards and to provide means to customize cloud resources regardless of their representation, we propose a semantic framework with a knowledge base of heterogeneous cloud resource descriptions. Fig. 5.2 provides an overview of the proposed framework.

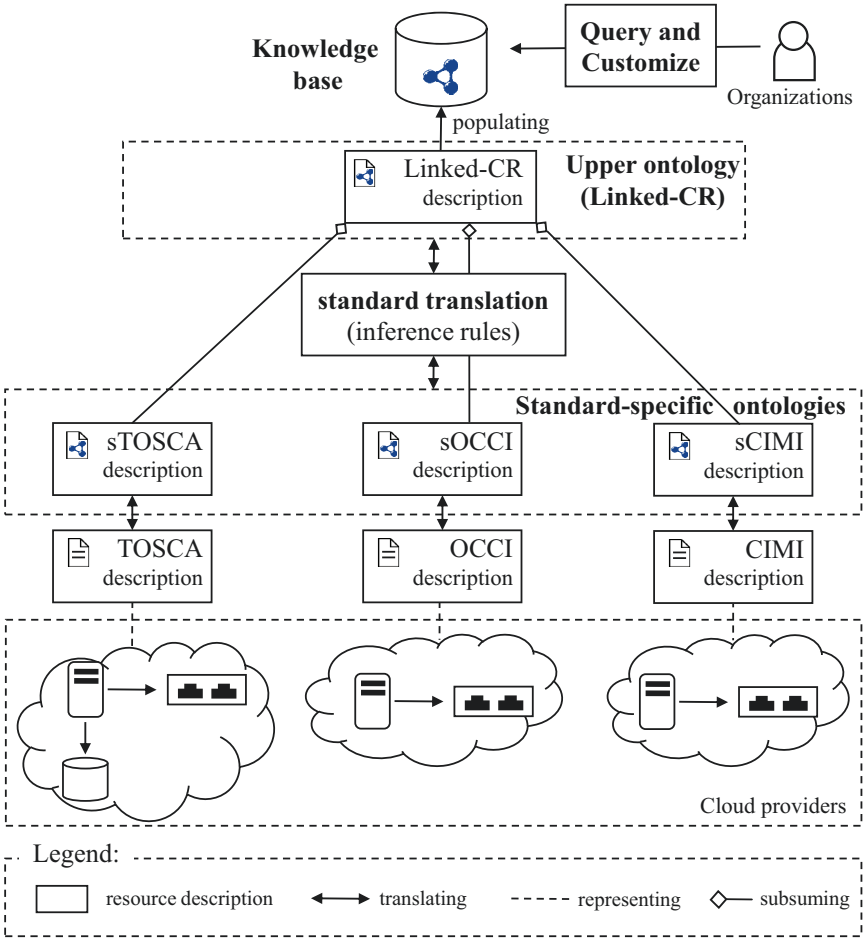


Figure 5.2: Semantic framework overview

This framework offers two core layers. The **standard-specific ontologies** layer offers ontologies defining concepts to describe cloud resources modeled with a specific standard. These ontologies include **sTOSCA**, **sOCCI** and **sCIMI** which allow

describing cloud resrouces modeled with TOSCA, OCCI, and CIMI standards, respectively (see section 5.3.1). The **upper level ontology** layer offers an ontology (**Linked-CR**) which defines high level concepts to describe cloud resources modeled with different standards (see section 5.3.2). We use semantic inference rules to enable on automated **standard translation** from standard-specific ontologies to Linked-CR and vise-versa (see section 5.3.3). Cloud resource descriptions modeled with Linked-CR are stored in a shared **knowledge base** which allows organizations to **query and customize** (see section 5.3.4) cloud resources regardless of their representation used by cloud providers (see section 5.3.4). In the following, we detail these two core layers of our framework as well as the **standard translation** and **query**.

5.3.1 Standard-specific ontologies

In order to represent cloud resource descriptions described using different standards, we propose *Standard-specific ontologies*. We define Standard-specific ontologies as ontologies describing cloud resource description standards. In our work we consider TOSCA, OCCI, and CIMI. Concretely, we identify concepts and relationships defined in the standard specifications of these description models and represent them as ontologies. We define three *Standard-specific ontologies* as follows:

5.3.1.1 sTOSCA

sTOSCA ontology represents cloud resources described in TOSCA. Fig. 5.3 provides an excerpt from sTOSCA concepts and their properties. Note that we do not include all concepts in this thesis due the limited space. The prefix "st" represents the sTOSCA namespace. The core sTOSCA concepts are:

- **st:Node** is the root concept that all complex sTOSCA concepts derive from, such as st:Compute, st:BlockStorage, st:Network, and st:Port. st:Node can describe its st:Capability through object property st:hasCapability and its Requirement through st:hasRequirement which will be described later on.
- **st:Compute** represents a real or virtual processors of software applications or services.
- **st:Network** represents a simple, logical network service.
- **st:Port** represents a logical entity that associates between st:Compute and st:Network.
- **st:BlockStorage** represents a storage device.
- **st:Capability** represents a set of data that associated with st:Node, for example, the data property st:hasCPU which describes the number of CPU of the st:Compute.

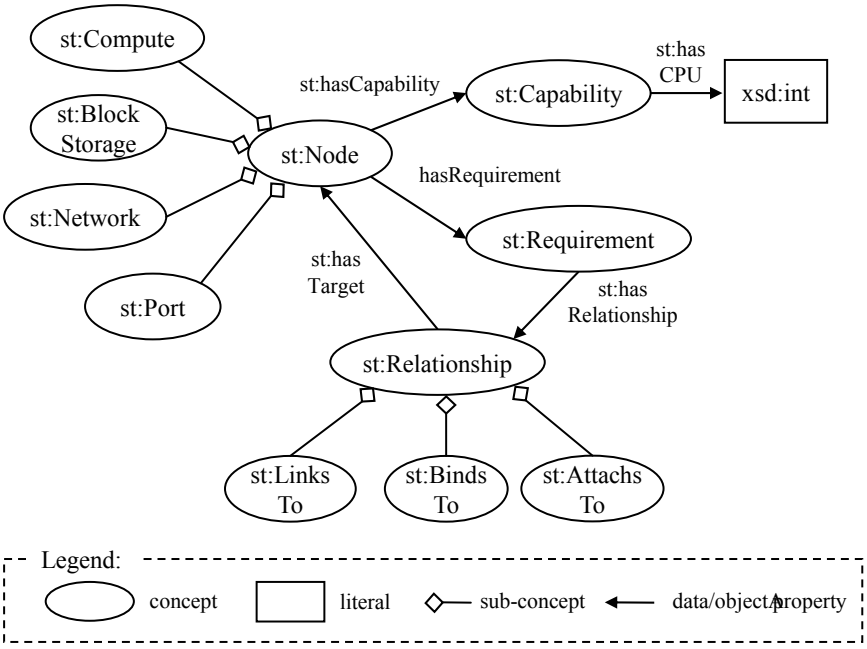


Figure 5.3: Excerpt from sTOSCA ontology

- **st:Requirement** represents a kind of requirement that a st:Node can expose. It can also represent a relationship to another st:Node through object property st:hasRelationship.
- **st:Relationship** represents a connection from one Node to another through object properties st:hasSource and st:hasTarget. It has multiple sub-concepts, for example, st:LinksTo, st:BindsTo and st:AttachesTo.
- **st:LinksTo** represents a network connection from st:Port to st:Network.
- **st:BindsTo** represents a network connection from st:Port to st:Compute.
- **st:AttachesTo** represents a connection relationship between st:Storage and st:Compute.

Cloud resources C_2 from Fig. 5.1 modeled with sTOSCA is shown in Listing 5.3. Such representation populates to the shared knowledge base.

5.3.1.2 sOCCI

sOCCI ontology semantically represents cloud resources described in OCCI. Fig. 5.4 provides an excerpt from sOCCI concepts and their properties. The prefix "so" represents the sOCCI namespace. The core sOCCI concepts are:

- **so:Category** is the root concept that all sOCCI concepts derive from.

Listing 5.3: Cloud resources C_2 modeled with sTOSCA

```
cb rdf:type st:Compute ;
  hasCapability capability1 ;
capability1 rdf:type st:Capability ;
  st:hasCPU "2"^^xsd:int ;
nb rdf:type st:Network
port rdf:type st:Port ;
  st:hasRequirement linkable1 ;
linkable1 rdf:type st:Requirement ;
  st:hasRelationship lb ;
lb rdf:type st:LinksTo ;
  st:hasTarget nb ;
...
```

- **so:Kind** represents the type identification so:Entity types.
- **so:Mixin** represents an extension mechanism, which allows defining new resource types with additional data/object properties.
- **so:Entity** represents an abstract type, which both so:Resource and so:Link inherit. so:Entity type identification is described through object property so:isKindOf to so:Kind. It can extend more data/object properties through object property so:hasMixin.
- **so:Resource** represents a cloud resource modeled with OCCI. It has three sub-concepts: so:Compute, so:Network, and so:Storage.
- **so:Link** represents a connection between cloud resources modeled with OCCI. It has two sub-concepts: so:NetworkInterface and so:StorageLink. A so:Link between two so:Resource instances can be represented through the object properties so:hasSource and so:hasTarget.
- **so:Compute** represents an information processing resource, e.g. a virtual machine. One of the data properties of so:Compute is so:hasCores which describes the number of CPU.
- **so:Network** represents a networking device (e.g., switch).
- **so:Storage** represents a data storage device (e.g., disk).
- **so:StorageLink** inherits from so:Link and represents a link from a so:Compute to a target so:Storage instance (e.g. Linking a Virtual machine to a disk).
- **so:NetworkInterface** inherits from so:Link and represents a link from a so:Compute to a target so:Network instance (e.g. network adapter).

Listing 5.4 depicts cloud resources C_2 from Fig. 5.1 modeled with sOCCI.

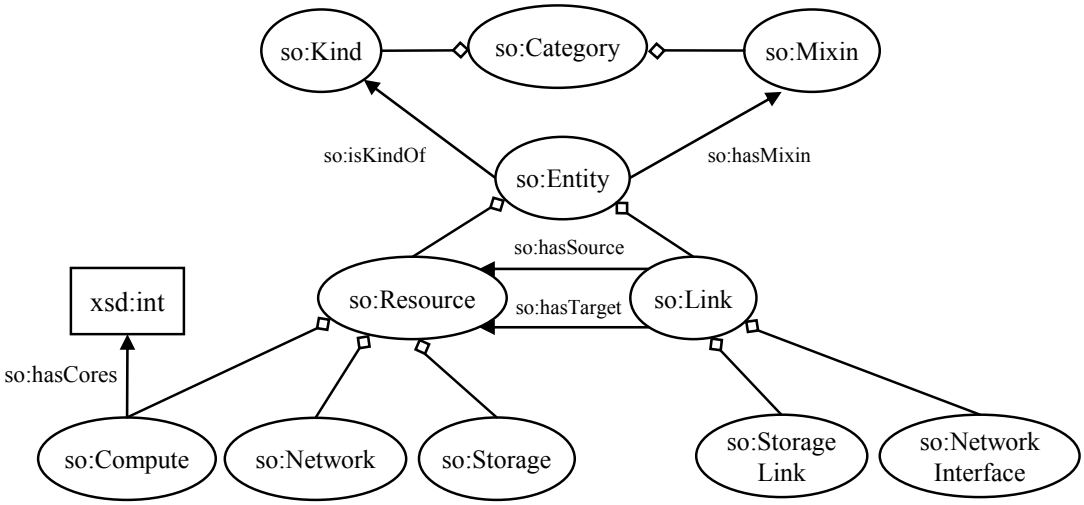


Figure 5.4: Excerpt from sOCCI ontology

Listing 5.4: Cloud resources C_2 modeled with sOCCI

```
1 cb rdf:type so:Compute ;
2   so:hasCores "2"^^xsd:int ;
3 nb rdf:type so:Network
4 lb rdf:type so:NetworkInterface
5   so:hasSource cb ;
6   so:hasTarget nb ;
7 ...
```

5.3.1.3 sCIMI

sCIMI ontology represents cloud resources described in CIMI. Fig. 5.4 provides an excerpt from sCIMI concepts and their properties. The prefix "sc" represents the sCIMI namespace. The core sCIMI concepts are:

- **sc:CloudEntryPoint** represents the cloud interface defined by CIMI.
- **sc:Resource** represents a cloud resource modeled with CIMI. It has three subconcepts: sc:Machine, sc:Network, and sc:Volume.
- **sc:Machine** represents a compute resource that encapsulates both CPU and Memory. We define different data properties to define an sc:Machine characteristics. For instance, the number of sc:Machine CPU can be described using the data property sc:hasCPU.
- **sc:Network** represents a logical network service with the purpose of forwarding data traffic between resources (e.g., switch).

- **sc:Volume** represents a storage at either the block or the file-system level. An sc:Volume can be connected to an sc:Machine. Once connected, sc:Volume can be accessed by processes on that sc:Machine.
- **sc:NetworkPort** represents a connection point between an sc:Network and an sc:MachineNetworkInterface through the object property sc:hasNetworkPort.
- **sc:MachineNetworkInterface** represents an association between an sc:Machine and an sc:Network. We define sc:Machine connects to sc:MachineNetworkInterface through object property sc:hasMachineNetworkInterface and sc:MachineNetworkInterface connects to sc:Network through object property sc:hasNetwork.
- **sc:MachineVolume** represents an association between an sc:Machine and an sc:Volume through the object property sc:hasMachineVolume.

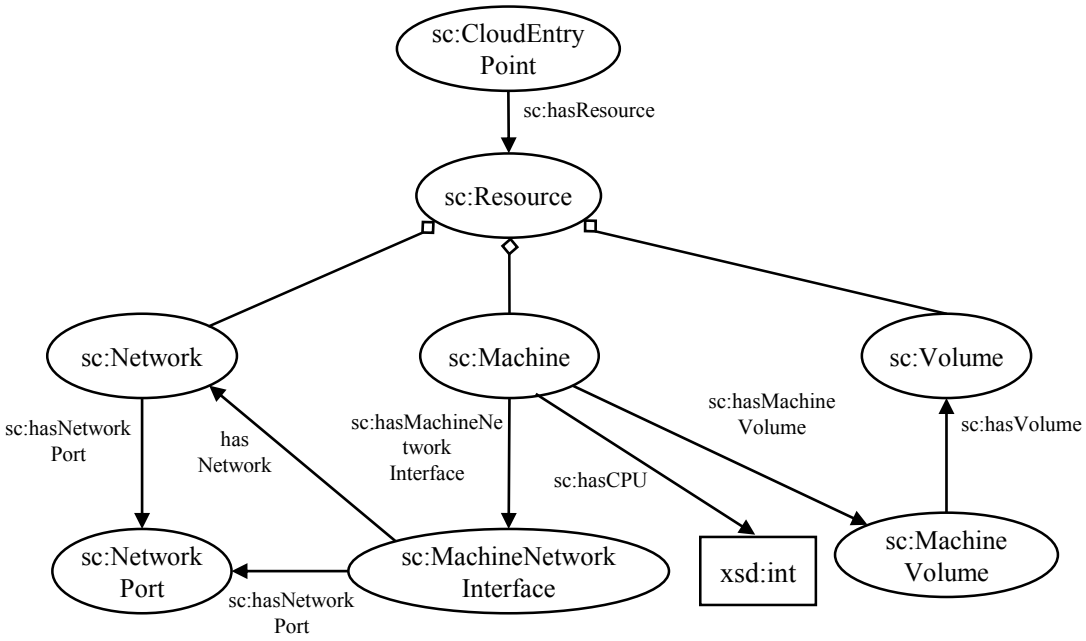


Figure 5.5: Excerpt from sCIMI ontology

Cloud resources C_2 from Fig. 5.1 modeled with sCIMI is shown in Listing 5.5.

5.3.2 Upper-level ontology

We define Linked-CR as an upper level representation of our *standard-specific ontologies*. Thus, Linked-CR enables interoperability between different cloud resource description standards and facilities the sharing of cloud resources in a collaborative environment independently from used standards. Concretely, we identify the common concepts

Listing 5.5: Cloud resources C_2 modeled with sCIMI

```
cb rdf:type sc:Machine ;
  sc:hasCPU "2"^^xsd:int ;
  sc:hasMachineNetworkInterface lb ;
nb rdf:type sc:Network
lb rdf:type sc:MachineNetworkInterface
  sc:hasNetwork nb ;
...
```

of standard-specific ontologies. Fig. 5.6 depicts Linked-CR concepts and their properties. The prefix "cr" represents the Linked-CR namespace. The core Linked-CR concepts are as follows:

- **cr:CloudResource** subsumes all concepts describing cloud resources. It has three sub-concepts: cr:Compute, cr:Network, and cr:Storage.
- **cr:Compute** represents common concepts: st:Compute, so:Compute, and sc:Compute.
- **cr:Network** represents common concepts: st:Network, so:Network, and sc:Network.
- **cr:Storage** represents common concepts: st:BlockStorage, so:Storage, and sc:Volume.
- **cr:NetworkInterface** represents common concepts: both st:LinksTo and st:BindsTo so:NetworkInterface, and sc:MachineNetworkInterface. It describes the link between so:Compute and so:Network through object properties so:hasNetworkInterface and so:hasNetwork.
- **cr:NetworkPort** represents common concepts: st:Port and sc:NetworkPort. It represents the link between cr:NetworkInterface and cr:Network through object property cr:hasNetworkPort.
- **cr:StorageLink** represents common concepts: st:AttachesTo, so:StorageLink, and sc:MachineVolume. It describes the link between sc:Compute and sc:Storage through object properties so:hasStorageLink and so:hasStorage.

We store cloud resource descriptions modeled with Linked-CR in a knowledge base. The example of cloud resources C_2 from Fig. 5.1 modeled with Linked-CR in our knowledge base is shown in Listing 5.6.

5.3.3 Standard translation

Thanks to Semantic technologies, we can define a set of inference rules for an automated translation of cloud resource descriptions represented with one standard-specific ontology to another. For example, we can define a set of SWRL rules to

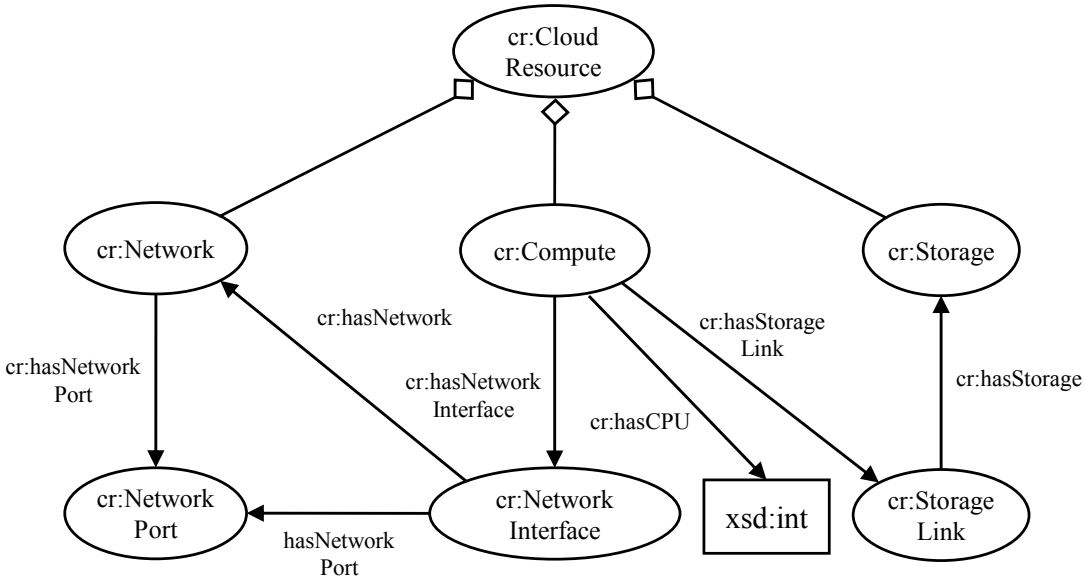


Figure 5.6: Excerpt from Linked-CR ontology

Listing 5.6: Cloud resources C_2 modeled with Linked-CR

```
1 cb rdf:type cr:Compute ;
2   cr:hasCPU "2"^^xsd:int ;
3   cr:hasNetworkInterface lb ;
4 nb rdf:type cr:Network
5 lb rdf:type cr:NetworkInterface
6   cr:hasNetwork nb ;
7 ...
```

translate a cloud resource description modeled with sTOSCA to Linked-CR (in Listing 5.7), and from Linked-CR to sOCCI (in Listing 5.8). Such rules allows an automatic translation of cloud resources semantically represented in one upper level ontology to another.

5.3.4 Querying

We populate a shared knowledge base of cloud resources semantically represented in Linked-CR. The knowledge base allows organizations to retrieve and manipulate these cloud resources regardless of their actual description models used by cloud providers. To query the knowledge base, we use the SPARQL query language. Listing 5.9 shows a SPARQL query that retrieves a list of compute resources linked to network resources. Given cloud resources C_1 and C_2 from Fig. 5.1 semantically represented in *Linked-CR*, we can execute this query on the knowledge base and the result are C_1 links to n_a and

Listing 5.7: Excerpt from sTOSCA → Linked-CR SWRL rules

```
1: st:Compute(?x) ⟶ cr:Compute(?x)
2: st:Compute(?x) ∧ st:hasCapability(?x, ?c) ∧
   st:hasCPU(?c, ?cpu) ⟶ cr:hasCPU(?c, ?cpu)
3: st:Network(?x) ⟶ cr:Network(?x)
4: st:LinksTo(?x) ⟶ cr:NetworkInterface(?x)
5: st:LinksTo(?x) ∧ st:hasSource(?x, ?y)
   ⟶ cr:hasNetworkInterface(?y, ?x)
6: st:LinksTo(?x) ∧ st:hasTarget(?x, ?y)
   ⟶ cr:hasNetwork(?x, ?y)
```

Listing 5.8: Excerpt from Linked-CR → sOCCI SWRL rules

```
1: cr:Compute(?x) ⟶ so:Compute(?x)
2: cr:Compute(?x) ∧ cr:hasCPU(?c, ?cpu)
   ⟶ so:hasCores(?x, ?cpu)
3: cr:Network(?x) ⟶ so:Network(?x)
4: cr:NetworkInterface(?x) ⟶ so:NetworkInterface(?x)
5: cr:NetworkInterface(?x)
   ∧ cr:hasNetworkInterface(?x, ?y)
   ⟶ so:hasSource(?x, ?y)
6 cr:NetworkInterface(?x) ∧ cr:hasNetwork(?x, ?y)
   ⟶ so:hasTarget(?x, ?y)
```

C_2 links to n_b. Furthermore, a user can execute the SPARQL query in Listing 5.10 to create a link l_{ab} between C_2 and n_a which enables interoperability between cloud resources C_1 and C_2.

5.4 Conclusion

In this chapter, we answered questions raised in the thesis problematic which is *R4: How to model heterogeneous cloud resources to be shared in a common knowledge*

Listing 5.9: SPARQL query for retrieving links between compute and network resources

```
SELECT ?compute ?network
WHERE
{
  ?compute rdf:type cr:Compute .
  ?compute cr:hasNetworkInterface ?networkInterface .
  ?networkInterface cr:hasNetwork ?network .
}
```

Listing 5.10: SPARQL query for creating a link between compute and network resources

```
INSERT DATA
{
  :l2 rdf:type ?networkInterface .
  :c2 cr:hasNetworkInterface :l2 .
  :l2 cr:hasNetwork :n1 .
}
```

base?.

To model heterogeneous cloud resources described using different standards, we propose a semantic framework built based upon our proposed ontologies. Concretely, we have defined ontologies to represent cloud resources and thus store in a shared the knowledge base. These cloud resources are described using a common ontology *Linked-CR*. Our framework also offers three *Standard-specific ontologies*, enriched with translation rules, allowing an automatic translation between resource descriptions in TOSCA, OCCI, and CIMI.

Our principles presented in Section 1.4.1 are respected:

- **Heterogeneous data modeling**: We propose our ontologies to model heterogeneous cloud resource described using different standards..
- **Automation**: We propose an automated approach to construct a knowledge base from heterogeneous cloud resources. Moreover, we define translation rules to automatically translate cloud resources between different description standards (i.e., TOSCA, OCCI, CIMI).
- **Implicit knowledge exploitation**: We utilize existing cloud resources shared within an organization.

To validate our approach, we develop a proof of concept prototype to populate a knowledge base from cloud resources described using different standards in Section 7.2.2. We also evaluate our proposed ontologies in qualitative and quantitative aspect in Section 7.3.4.

Chapter 6

Assisting Cloud Resource Allocation to Business Processes using Genetic Algorithm

Contents

6.1 Introduction

This chapter addresses the research question: *R5: How to solve optimal allocation of cloud resources to business processes?* To do so, we present a novel approach using a genetic algorithm to select the optimal cloud resource allocations to BPs in term of quality of service (QoS). We start by an illustrating example (Section 6.2). Thereafter, we describe how to model cloud resource allocation to BPs (Section 6.3). In Section 6.4, we present our approach to find the most optimal resource allocation variant using a genetic algorithm. Finally, we conclude the chapter in Section 6.5.

The work in this chapter was published in conference proceedings [166].

6.2 Illustrating example

We present in the following a scenario (Fig. 6.1) to illustrate our approach. We consider an organization that is going to deploy their BP on their own cloud environment. Such cloud environment offers different cloud resources: c_1, c_2, n_1, n_2, n_3,

s_1, and s_2. These resources types are: *Compute* (c_i), *Network* (n_i), and *Storage* (s_i). *Compute* represents a computing resource that encapsulates both CPU and memory. *Network* represents a networking entity that forwards data traffic between end points. *Storage* is a resource that stores information. Note that we consider only cloud resources at the infrastructure level. We suppose that these cloud resources are modeled with Linked-CR ontology (Section 5.3.2) in a knowledge base shared within an organization.

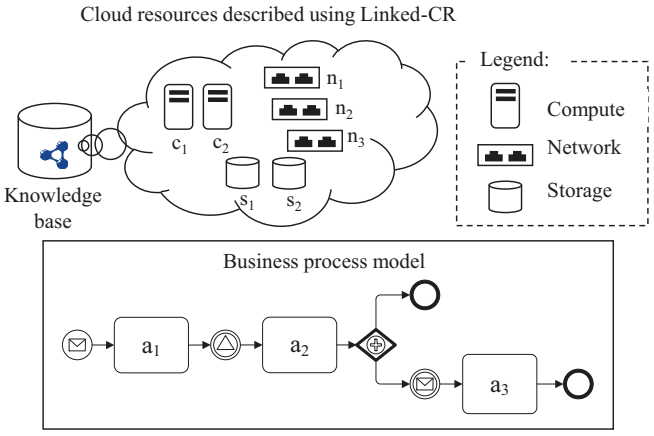


Figure 6.1: Illustrating example

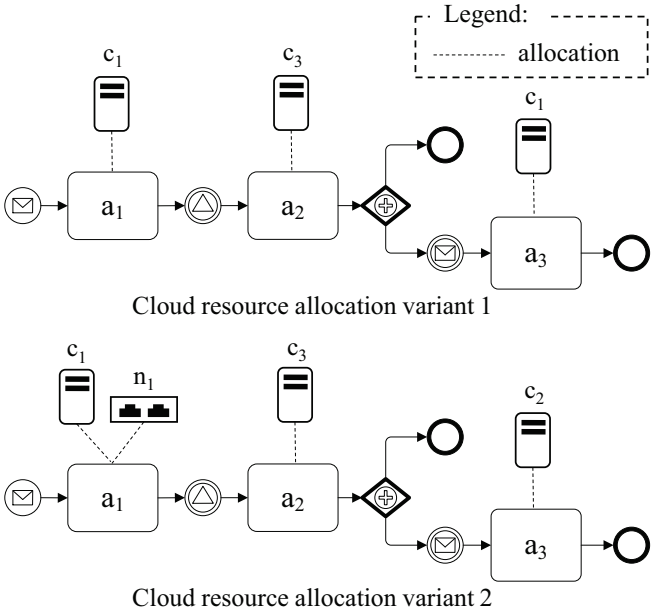


Figure 6.2: Example: cloud resource allocation variants of the same BP

In this chapter, we take into account variants of resource allocation to BPs instead of variants of BP control flow. Suppose that an organization needs to allocate cloud resources to BP activities a_1, a_2, and a_3. However, ensuring an optimal resource allocation in term of quality of service (QoS) might be combinatorial problem due to the amount of possible allocation variants. For example, the number of possible cloud resource allocation variants of a_1 is $2^7 = 128$ because there are 7 available cloud resources. Thereafter, the number of possible cloud resource allocations variants for a_1, a_2, and a_3 is 128^3. Figure 6.2 depicts two example of such variants. Solving this allocation problem is tedious and complex. Thus, we overcome this issue by proposing an operator to model cloud resource allocation which allows to specify only desirable allocation variants (Section 6.3). Then, we formalize our problem using a genetic algorithm in order to obtain an optimal allocation variant with respect to particular QoS properties.

6.3 Cloud Resource Allocation Operator

In this section, we describe how to model resource allocation to BPs inspired from our previous work [167]. Each BP activity has different requirements/needs of cloud resources which could be modeled with variables. Therefore, we propose a cloud resource allocation operator describes cloud resources allocated to a specific activity. To do so, we define two parameters for the operation: (*i*) a connector type and (*ii*) a range. We formally define our operator as follow:

Definition 6.3.1 (Cloud resource allocation operator)**.** *A cloud resource allocation operator is denoted as* $O = \{CR_O, A_O, ct, R\}$ *where:*

- $CR_O = \{cr_1, cr_2, ..., cr_n\}$ *is a set of possible cloud resource allocation;*
- $A_O = \{a_1, a_2, ...a_n\}$ *is a set of assigned BP activity;*
- $ct \in \{XOR, OR, AND\}$ *is a connector type (i.e., gateway);*
- $R_{rt} = \{(min, max)\}$ *is an order pair of range (i.e., allocation guideline) for connector type* OR, $rt \in \{C, N, S\}$ *is a cloud resource type (*C *is compute,* N *is network, and* S *is storage),* min *and* max *is the minimum and maximum number of allocated resources, respectively;*
- $R = \{R_C, R_N, R_S\}$ *is a set of ranges for each cloud resource type.*

Figure 6.3 shows examples of our operator with its parameters. A connector type ct parameter can be either a exclusive choice (XOR), a inclusive choice (OR), or a parallel flow (AND). These connectors have similar behavior as the BP control flow connectors. Concretely, XOR allows only one cloud resource to be allocated, AND allocates all resources, and OR allocates one or more cloud resources to be

allocated. A range $Range_x$ parameter represents the minimal and maximum number of cloud resource needed where $x \in \{C, N, S\}$ is a required resource type. For example, a_1 is assigned to an operator o_1 which has type OR, requires from 1 to 2 compute resources ($R_C = \{1, 2\}$), and only 1 network resource ($R_N = \{1, 2\}$). Therefore, cloud resources allocation variants of o_1 are $\{c_1, n_1\}$, $\{c_1, n_2\}$, $\{c_2, n_1\}$, $\{c_2, n_2\}$, $\{c_1, c_2, n_1\}$, and $\{c_1, c_2, n_2\}$.

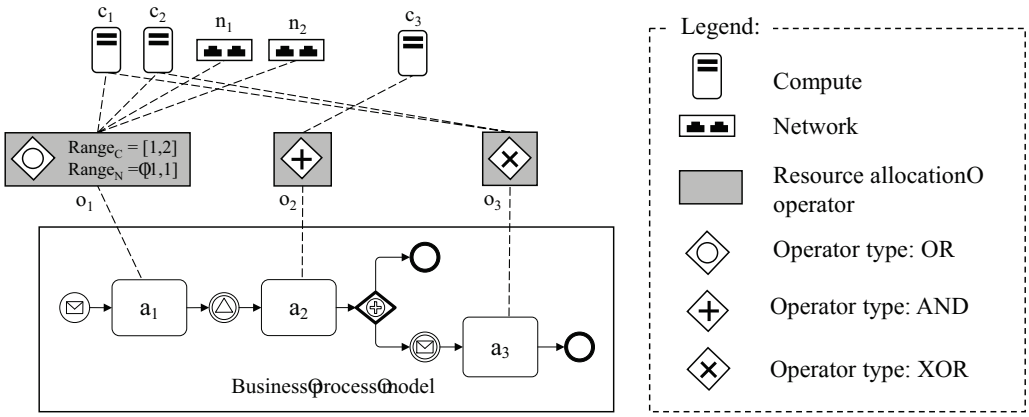


Figure 6.3: Example: configurable resource operators

6.4 Problem Formalization using Genetic Algorithm

In this section, we formalize our problem that aims at ensuring an optimal cloud resource allocation to BPs using genetic algorithm. Genetic algorithm is a powerful tool to deal with combinatorial problems. In large-scale complexity problem, genetic algorithm has been preferred than other approaches such as Linear Integer Programming (LIP) [168]. Furthermore, it has been successfully applied in many other research domains. To apply genetic algorithm, we encode the problem in the form of genomes (Section 6.4.1). Thereafter, we show our genetic algorithm with the assessment of particular quality of service (QoS) properties in Section 6.4.2.

6.4.1 Genome Encoding

We use a genetic algorithm to find the most optimal cloud resource allocation variant in BPs. Genetic algorithm [169] is an optimal solution search in which the problem is encoded in the form of an array which represent individuals. The set of these individuals, which constitute a population, is randomly created to represent different points in the search space. Each individual is evaluated by a fitness function that represent the degree of goodness. Operators (e.g., crossover, and mutation) are applied to generate new individuals. This generation continues until a maximum number of generations

is achieved or specific conditions are satisfied, then the output is the individual with the best fitness value as an optimal solution [170].

Our BP model contains a set of activities A and a set of cloud resources CR; and using our resource allocation operator, each activity a_i is associated with multiple cloud resource allocation variants RAV_{a_i} (Definition 6.4.1). For example, an activity a_1 (Fig. 6.3) has the following resource allocation variants $RAV_{a_1} = \{\{c_1, n_1\}, \{c_1, n_2\}, \{c_2, n_1\}, \{c_2, n_2\}, \{c_1, c_2, n_1\}, \{c_1, c_2, n_2\}\}$.

Definition 6.4.1 (Resource allocation variant)**.** *A set of resource allocation variants for a BP is denoted as* $RAV_{BP} = \{RAV_{a_1}, RAV_{a_1}, ..., RAV_{a_k}\}$ *where:*

- $A = \{a_1, a_2, ..., a_m\}$ *is a set of activities;*
- $CR = \{cr_1, cr_2, ..., cr_l\}$ *is a set of cloud resources;*
- $RAV_{a_i} = \{RAV_{a_{i1}}, RAV_{a_{i2}}, ..., RAV_{a_{in}}\}$ *is a set of resource allocation variant for activity* a_i*. It contains multiple combinations of cloud resources from* CR *based on cloud resource allocation operators assigned to* a_i *where* n *is the number of possible variants.*

We encode multiple cloud resource allocation variants to a genome. The genome represents an array where the array size is equal to the number of activities. Each element refers to an activity a_i in which its array value is an index into the set of cloud resource allocation variant RAV_{a_i}. We apply the standard two-point crossover operator to generate the populations of genome array. Such operator use previous populations to produce a new population. We use mutation operator to randomly select an activity a_i (i.e., an index in the genome array) and replaces the array value by the index to possible resource allocation variant in RAV_{a_i}. In our example (Fig. 6.3), the maximum number of resource allocation variants for one activity is 6. Thus, three bits are enough to express every cloud resource allocation variants (Table 6.1). One of the possible generated population is 010 0 01 which represents respectively cloud resource allocation variants $RAV_{a_{13}} = \{c_2, n_1\}$ for the activity a_1, $RAV_{a_{21}} = \{s_1\}$ for a_2, and $RAV_{a_{32}} = \{c_2\}$ for a_3.

6.4.2 Fitness Function

We can now model the problem by means of a fitness function. To do so, we associate our fitness function to the important quality of service (QoS) properties: cost (CT), response time (RT) and availability (AT). The two first constraints denote that the CT and RT of a given BP should be less than maximal thresholds $cmax$ and $rmax$. While the last constraint means that the AT should be greater than minimal threshold referred to as $amax$.

Activity	Cloud resource allocation variant	Bit representation
a_1	$RAV_{a_{11}} = \{c_1, n_1\}$	000
	$RAV_{a_{12}} = \{c_1, n_2\}$	001
	$RAV_{a_{13}} = \{c_2, n_1\}$	010
	$RAV_{a_{14}} = \{c_2, n_2\}$	011
	$RAV_{a_{15}} = \{c_1, c_2, n_1\}$	100
	$RAV_{a_{16}} = \{c_1, c_2, n_2\}$	101
a_2	$RAV_{a_{21}} = \{c_3\}$	0
a_3	$RAV_{a_{21}} = \{c_1\}$	00
	$RAV_{a_{22}} = \{c_2\}$	01

Table 6.1: Genome encoding of cloud resource allocation variants encoded to genome

$$\begin{aligned} CT(BP) &\geq cmax \\ RT(BP) &\geq rmax \\ AT(BP) &\leq amax \end{aligned} \tag{6.1}$$

Thereafter, we define a fitness function (FF) to enable the assessment of these QoS properties for a given BP described as follows.

$$\begin{aligned} FF(BP) = \alpha_1 * (cmax - CT(BP)) + \alpha_2 * (rmax - RT(BP)) \\ + \alpha_3 * (AT(BP) - amax) \end{aligned} \tag{6.2}$$

Where α_1, α_2 and α_3 are defined as the weighting factors that belong to the interval [0,1]. They represent the user preferences w.r.t the QoS properties ($\alpha_1 + \alpha_2 + \alpha_3 = 1$).

We model our problem by means of a fitness function. The fitness function needs to maximize the function FF (see Equation 6.2). It is necessary to define a stop condition for genetic algorithm. Hence, we consider the following condition:

$$FF(g) \geq \delta \tag{6.3}$$

where δ is the weighting quality factor (positive real value). The value of δ is determined by experimentations. A solution represented by a genome is feasible only if it respects this stop condition. Algorithm 5 shows the steps of our genetic-based algorithm. Firstly, our algorithm input is a set of cloud resource allocation variants RAV_{BP} and a δ value. Firstly, we generate an initial bit representation *bit* based on the maximum number of cloud resources in RAV (line 1). For example, an initial

bit representation for cloud resource allocation variants in Table 6.1 is 6 bits (000 0 00). We start our genetic algorithm with an initial bit in line 2 and return a bit represented optimal allocation in line 3. In our genetic algorithm (line 4-11), we compute the fitness function FF from *bit* (line 5). If the result is less then the δ value, we apply the crossover operator for obtaining next bit representation by taking into account previous populations to produce a new population (line 6-7). The mutation operator randomly selects an activity (i.e., a position in the genome) and randomly replaces the bit representation with another possible cloud resource allocation variant. Thus, we recursively call the genetic algorithm function for the next population (line 8). If the result is satisfy with the δ value, the algorithm will return the resource allocation variant of the bit representation *bit* (line 9).

Algorithm 5 Energy efficient resource allocation variant construction's algorithm

Input: RAV_{BP}, δ
Output: *bit*

```
1: bit ← GenerateAnInitialBit(RAV_BP)   ▷ Generate an initial bit representation
2: bit ← GENOME(bit, δ)                 ▷ Start genetic algorithm.
3: return bit                           ▷ Return an optimal cloud resource allocation variant.
4: procedure GENOME(bit, δ)             ▷ Check if the stop condition is respected
5:     while FF(bit) < δ do
6:         bit ← Crossover()            ▷ Apply the crossover operation to the bit
7:         bit ← Mutation()             ▷ Apply the mutation operation to the bit
8:         GENOME(bit, δ)               ▷ Recuresive call for the next population.
9:     end while
10:    return bit                       ▷ Return an optimal cloud resource allocation variant.
11: end procedure
```

6.5 Conclusion

In this chapter, we answered the research questions raised in the thesis problematic which is: *R5: How to solve optimal allocation of cloud resources to business processes?*.

We proposed an approach that aims at ensuring an optimal cloud resource allocation to BPs. We reuse our previous work [167] to model cloud resource allocation to BP models. Thereafter, we apply our genetic-based approach to guarantee, on the one hand, the selection of the process variant which best fits to the organization's branch requirements, and optimize on the other hand the QoS impact.

The approach we present in this section respects our principles introduced in Section 1.4.1:

- **Automation**: We propose an automated approach find the optimal cloud resource allocation to BPs using genetic algorithm.

We validate our approach by performing experiment (Section 7.3.3) to compare our genetic-based approach with LIP which is a broadly used mathematical optimization program [171].

Chapter 7

Evaluation and Validation

Contents

7.1 Introduction

In this chapter, we present (*i*) a prototype as a proof of concept to realize our automated approaches to assist the design of BP variants and populate a knowledge base of heterogeneous process data (i.e., BP models and process event logs) and cloud resources, and (*ii*) experiments that we have conducted to evaluate the efficiency of our solutions. Our objective is to prove that our approach is feasible and accurate in real use-cases.

Basically, we implemented two proof of concepts:

- A BP variant modeling application by extending Signavio Process Editor[1]. It assists the design of BP variants by recommending BP fragments. Its implementation is based on the ontologies and algorithms developed in Chapter 3 and 4.
- An application that populates a knowledge base of cloud resources. Its implementation is based on the ontologies and algorithms developed in Chapter 5.

Regarding the validation of our approach, we have performed experiments on large public datasets and real use cases.

- For the automated support of BP variant design by exploiting heterogeneous BP models, we used two large public dataset of real BPs: (*i*) SAP reference models [75], (*ii*) and BP models [76] shared by the IBM Business Integration Technologies (BIT) team. We showed that our approach is feasible in realistic situations by providing the recommendation results. We also proved that our approach is accurate by evaluating its precision and recall metrics.
- For the automated support of BP variant design by exploiting heterogeneous process event logs, firstly we evaluated our proposed ontologies by comparing process event representation to their standards. Secondly, we evaluated our recommendation approach by using simulated logs generated from the SAP reference models. We showed that our approach is feasible and accurate by evaluating its precision and recall metrics.
- For the optimal cloud resource allocation to BP variants, we compare our approach with another technique, Linear Integer Programming (LIP) [168]. We showed that our approach performs better than LIP when a large number of cloud resources are considered.
- For modeling heterogeneous cloud resources, firstly we evaluated our ontologies by comparing ontology concepts to classes from cloud resource description standards. Thereafter, we showed that our approach is feasible by applying it on use cases identified in different standard specifications.

We present in Section 7.2 our two implemented proof of concepts. The experiments and related discussion are presented in Section 7.3. Finally, we conclude the chapter in Section 7.4.

7.2 Proof of Concept

In this section, we present the proof of concepts that we have developed to realize our approach. Our approach aims to enable interoperability of heterogeneous process data

[1] https://code.google.com/p/signavio-core-components/

(i.e., BP models and process event logs) and cloud resources between an organization's branches. Thereafter, we exploit such data to design BP variants. To do so, we have developed prototypes to populate a common knowledge base using heterogeneous process data and cloud resources as input (Figure 7.1). We have extended Signavio application for existing BP variant design by populating and mining process data in

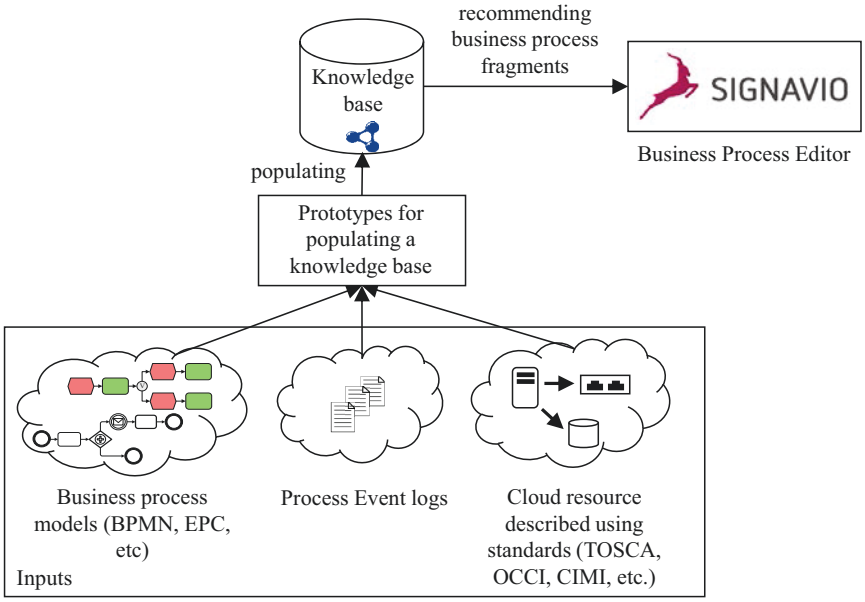


Figure 7.1: Our prototypes

7.2.1 BP variant design support application

Our approach provides means for BP variant design by recommending BP fragments that are relevant to selected positions of an under-design BP. Such recommended BP fragments are retrieved from a knowledge base of heterogeneous process data. In order to validate our approach, we extend the Signavio Process Editor as a proof of concept to support BP variant design by recommending appropriate BP fragments. Signavio is an open-source Web-based application for editing BPs in BPMN. As Signavio is Web-based, it is prone to be a cloud-based application. Our developed application can be found at `http://www-inf.it-sudparis.eu/SIMBAD/tools/BPVariant-modeling/`. We have added new functionalities to the Signavio which are highlighted in the screen-shot of Fig. 7.2.

Originally, only BPMN is supported by Signavio (area 6). Hence, we have enhanced it to also support EPC (area 1). To populate a knowledge base of heterogeneous process models (Chapter 3), we provide three properties for each BP element

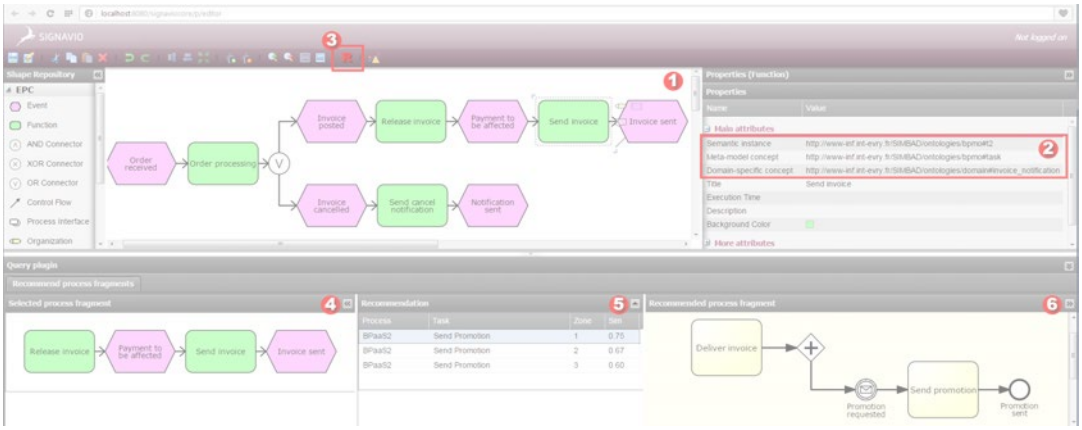


Figure 7.2: A screen-shot of our BP variant design support application

for semantic annotation as follows: (*i*) semantic instance representing the instance identifier in a knowledge base, (*ii*) meta-model annotation representing the BPMO concept, and (*iii*) domain-specific annotation representing the domain-specific concept. Users may semantically annotate a BP element by specifying these properties.

Furthermore, we provide a functionality to recommend process fragment based on neighborhood context matching based on ontologies and algorithms in Chapter 3 and 4 as a plugin in Signavio. Users may select a task from an under design BP in area 1, then click on the recommendation plugin (through the button highlighted in area 3). Area 4 illustrates the fragment of the business process model. Thereafter, the recommended process fragments and their matching degrees will be showed in area 5 and 6. Area 5 shows a list of BP fragments recommendations retrieved from the knowledge base. Area 6 shows the recommended BP fragment selected from the recommendation list.

After the construction of a semantically annotated BP model in Signavio, it is automatically stored as a set of RDF triples [160] in a knowledge base (represented as a triplestore database). *OpenLink Virtuoso*[2], a well-known opensource database engine for triplestores, is used as a triplestore. We extended Signavio with a functionality allowing it to access OpenLink Virtuoso triplestore using the Virtuoso Jena Provider API [172]. Hence, we can retrieve information from our knowledge base using a SPARQL endpoint provided by OpenLink Virtuoso. To recommend BP fragments, we assume that a knowledge base has already been populated by different BP models which can be done using this application.

[2]http://virtuoso.openlinksw.com/

7.2.2 Cloud resource knowledge base populating application

We develop a prototype as a proof of concept to populate an ontology-based knowledge base from cloud resources described using different standards. Concretely, we use OWL[3] ontology language to develop Standard-specific ontologies and Linked-CR via Protege[4], an open source ontology editor. Thus, our prototype stores the knowledge base as RDF triples modeled with these ontologies. Our prototype accesses the triplestore via Virtuoso Jena Provider API [172]. This API facilitates retrieving and manipulating the data in our knowledge base using SPARQL.

Cloud providers may use our application to import their cloud resource descriptions and thus populate the knowledge base using Standard-specific ontologies and Linked-CR. Once cloud resource descriptions have been stored in the knowledge base, cloud users (e.g., organizations) may retrieve and manipulate these resources through SPARQL queries. Users can also define links between cloud resource descriptions at this point in order to share existing resources and thus enable interoperability between organizations. Developed prototype can be found at `http://www-inf.it-sudparis.eu/SIMBAD/tools/linked-cr/`

7.3 Experimentation

In this section, we present the experiments that we performed to evaluate our two dedicated approaches for: (*i*) assisting BP variant design by exploiting heterogeneous process data (i.e., process models and process event logs) and (*ii*) optimal allocation of heterogeneous cloud resource to BPs. We firstly present the experiments for assisting the BP variant design by modeling and mining BP models (Section 7.3.1) and process event logs (Section 7.3.2) in a shared knowledge base. We evaluate our approach to optimally allocate cloud resources to BPs in Section 7.3.3. Thereafter, we show the evaluation on modeling heterogeneous cloud resources in our knowledge base (Section 7.3.4).

7.3.1 Modeling and mining business process models experiments

In order to evaluate the approach presented in Chapter 3, we conducted several experiments to prove that our approach is feasible in real use cases and assesses its accuracy in retrieving similar BP fragments from the knowledge base having similar neighborhood contexts from a large collection of heterogeneous BP models. Concretely, we populated a shared knowledge base with BP models annotated with BPMO and domain-specific ontology, namely Suggested Upper Merged Ontology (SUMO)[5]. SUMO is a formal public domain ontology mainly used for research. We implemented

[3]OWL is a knowledge representation language [173]

[4] `http://protege.stanford.edu/`

[5]http://www.adampease.org/OP/

a script to extract BP fragments from the knowledge base. Thereafter, BP Fragments are represented in the knowledge base with our NCFO ontology. Finally, we developed a tool to compute the matching between BP fragments for recommendations. To do so, we considered two datasets having different modeling languages (EPC and BPMN):

1. **Dataset 1** containing 205 EPC processes from the SAP reference models [75] represented in EPML (EPC Markup Language) format. In average, there are 4.2 functions, 4.2 start events, 6 end events, 14.2 events, 7.3 logical connectors and 25.9 flows per one process.

2. **Dataset 2** containing 850 BPMN process models [76] shared by the IBM Business Integration Technologies (BIT) team. In average, there are 11.3 activities, 2.6 start events, 3.4 end events, 19 gateways and 46.8 flows per one process.

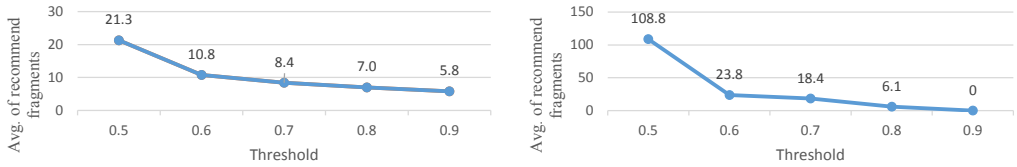
We conduct two experiments to evaluate the feasibility (Section 7.3.1.1) and the recommendation accuracy (Section 7.3.1.2) of our approach.

7.3.1.1 Approach feasibility

BP fragments are extracted from the datasets. We consider matching degrees between BP fragments within the same dataset only because the two datasets do not have the same type of deliverable services. We consider the size of fragments as $zone = 1$, and compute the matching degrees between fragments in the same dataset. For dataset 1, 814/870 fragments from the knowledge base (93.6%) have matching values with others greater than 0, in which 76.3% have matching values greater than 0.5 and 32.6% have matching values belonging to [0.9,1.0]. For dataset 2, 4373/4466 fragments from the knowledge base (97.9%) have matching values with fragments greater than 0, in which 1889 (43.2%) have matching values greater than 0.5 and 168 fragments (3.8%) have matching values belonging to [0.9,1.0].

In another experiment, for each BP fragment, within the fragment's size ($zone \leq 3$), we calculate the average number of recommended BP fragments that have matching values greater than a given threshold. With 0.8 as threshold, for each fragment, our approach recommends on average 21.3 fragments for the first dataset (see Fig. 7.3a), while for the second dataset (see Fig. 7.3b) the recommendation average is 6.05 fragments. We also notice that the average number of recommended fragments decreases when the threshold increases.

Our experiments, using a knowledge base populated from real and heterogeneous BP models, show that our approach is feasible in realistic situations. Therefore, similar BP fragments can be retrieved. Hence, BP variant designer can use this recommendation approach to facilitate the design of new BP variants by retrieving similar process fragments within an organization.



(a) Dataset 1 (b) Dataset 2

Figure 7.3: Average number of recommendation

7.3.1.2 Approach accuracy

We use the domain-specific ontology annotation as the ground-truth data for Precision (i.e., the rate of relevant recommended BP fragments) and Recall (i.e., the fraction of relevant BP fragments that are recommended) computation. To do so, for each fragment in each semantic BP model, we consider its root task as unknown. We then consider its recommended fragments. These fragments should have root tasks annotated to the same domain-specific concept.

Concretely, let's consider a selected fragment having a root task r in a process P. r appears in n processes. The recommendation results for this selected fragment, consist of l fragments from the knowledge base. Among them, we consider $t \leq l$ as the number of fragments having root task domain equivalent to r. Precision and Recall are computed as follows (Equation 7.1 and 7.2 respectively).

$$Precision = \frac{t}{l} \tag{7.1}$$

$$Recall = \frac{t}{n} \tag{7.2}$$

We conduct the precision/recall evaluation on the first dataset by retrieving recommendation for each fragment in each BP model. We vary the number of recommended fragments 5 to 1. Furthermore, we consider the matching in the first zone to ignore the noise of the irrelevant tasks in further zones. We compute Precision and Recall for only fragments having a root task that appear in at least 10 semantic BP models from the knowledge base. Consequently, 29 tasks selected as root tasks and 267 processes are considered for our experiments.

The average Precision and Recall values are shown in Fig. 7.4. The Precision value increases when the number of recommended fragments decreases which means that the relevant fragments are mostly part of the recommended fragments. However, the Recall value is not very high due to the fact that the number of recommended fragments is much more smaller than the relevant fragments.

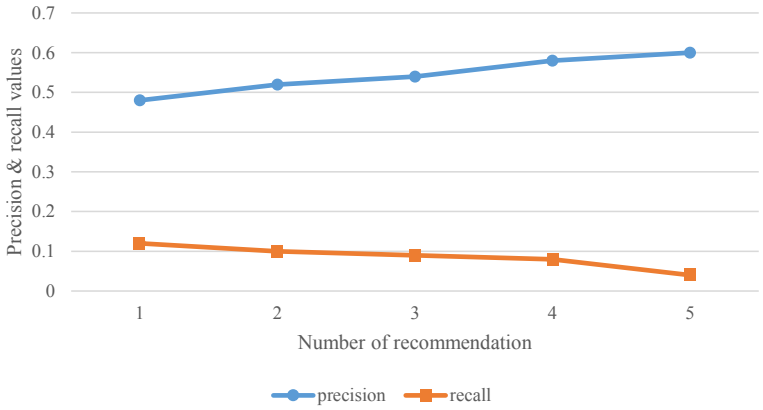


Figure 7.4: Precision and recall values

7.3.2 Modeling and mining process event logs experiments

In the following, we present the experimentation efforts made to validate our approach in Chapter 4. We first provide in Section 7.3.2.1 a testbed for our recommendation evaluation and an evaluation of our proposed ontologies. Thereafter, we use this testbed to experiment the feasibility of our recommendation approach in Section 7.3.2.2.

7.3.2.1 Setting the testbed

As a testbed for our approach evaluation, we populate a knowledge base of heterogeneous process event logs represented by our ontologies. We used simulated logs from realistic business processes to develop a shared knowledge base between organization's branches. The advantage of using simulated logs is that it is possible to vary external factors (e.g., user, timestamp) and thus to ensure a better diversity of examples and a more accurate validation.

We performed experiments on a large public dataset [174] which contains 186 different SAP reference models in EPC. We developed a script to transform SAP BP models to Colored Petri Nets (CPN) format and used a high-level Petri net tool, known as CPN Tools[6], to simulate the execution of these BP models and generate XES process event logs[7]. Table 7.1 provides more details about the generated logs.

On top of the generated logs, we use a heuristic mining technique to discover C-net BP models and then extract BP fragments. Thereafter, we represent discovered C-nets and C-net fragments in our knowledge base using *Linked-CN* and *NCFO*, respectively. Afterwards, we deploy our knowledge base on an open-source triple-store database, namely OpenLink Virtuoso.

[6] `http://wiki.daimi.au.dk/cpntools/cpntools.wiki`

[7] Our dataset in EPC, CPN and XES format are published at: `http://www-inf.it-sudparis.eu/SIMBAD/tools/semantic_cnet_fragment/dataset/`

	Min.	Max.	Average
No. of start events in an instance	1	9	1.68
No. of end events in an instance	1	11	2.60
No. of events in a process	2	28	5.32
No. of executions for each process model	52	299	178.9
No. of occurring events in a process	40	2910	425.23

Table 7.1: Dataset details

One of the most common approaches for evaluating ontologies is comparing them to a gold standard [175]. This can be done by checking the coverage of concepts defined in ontologies with classes defined in standards. In our case, the fundamental goal of this work is to semantically represent and share process event log fragments among organization's branches. Therefore, we build *Linked-CN* and *NCFO* based upon two standards: XES and Causal Net Modeling language. Hence, we evaluate our ontologies by analyzing their coverage to these two standards. We determine the coverage by checking if a standard class can be represented by our ontology concept or not. The results summarized in Table 7.2 show that we cover all classes of these two standards.

Standard	Classes	Covered	Ratio
XES	Log, Trace, Event, Attribute Nested Attribute, Global Attribute Event Classifiers	7	100%
Causal Net	Causal net, Activity, Start activity End activity, Input binding Output binding, Dependency relation	7	100%

Table 7.2: *Linked-CN* and *NCFO* coverage to standards

7.3.2.2 Recommendation evaluation

Using our knowledge base, we evaluate the number of recommended BP fragments for different activity fragments with a fragment size (*zone*) varying from 1 to 5. Fig. 7.5 shows the percentage of fragments that has at least 1 recommended fragment with a similarity value greater than 0, 0.5, and 0.8. Concretely, our experiments show that 56.49% fragments have at least one similar fragment (i.e. similarity value greater than 0) with *zone* number between 1 and 5. For $zone = 1$, 36.64% fragments with a similarity value greater than 0.5 and 30.92% fragments with a similarity value greater than 0.8 were recommended. Moreover, the percentage of recommended BP fragments decreases when the zone number increases because BP fragments contain more and more different neighbor activities. The obtained results indicate that our approach provides recommendations for the majority of fragments as we can retrieve similar fragments for more than 50% of fragments in average. This shows that our approach

is *feasible* and can be applied to assist BP variant design.

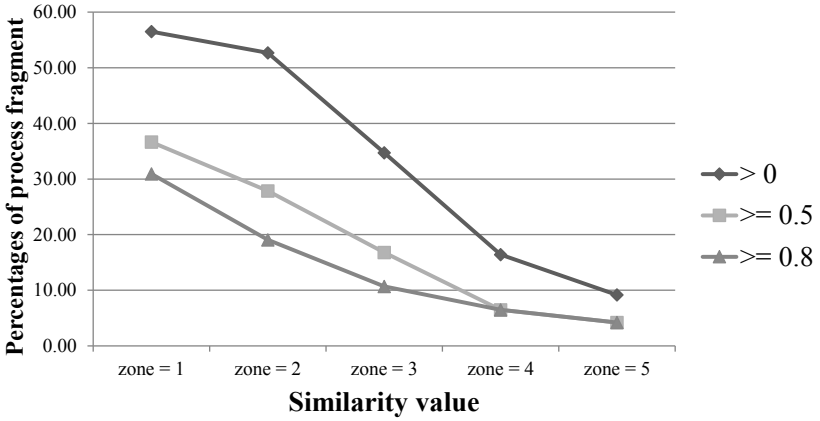


Figure 7.5: Percentage of recommended BP fragments

We also evaluate the *accuracy* of our approach based on precision and recall metrics. The primary objective is to retrieve fragments that are likely relevant (i.e., high precision) and to retrieve potentially relevant fragments (i.e., moderate recall). We measure the precision based on the root activity, i.e., a recommendation is precise if recommended fragment and selected fragments have the same root activity. Concretely, consider a fragment having root activity a as a selected fragment P. The recommendation for P consists of l fragments, in which t fragments having root activity a. Hence, $Precision = \frac{t}{l}$. Assume that n fragments have a as a root activity. Therefore, $Recall = \frac{t}{n}$. We compute the precision and recall values on top-N recommended fragments for each selected fragment. We performed the experiment with *zone* equals to 1 and on activities that appear in at least 10 different BPs. The results (see Fig. 7.6) show that we obtain good precision values (from 0.55 to 0.54). We also obtain moderate recall values (0.34 at top-10 fragments recommendation). Recall values increase when we increase the number of recommended fragments N which means that we can retrieve more relevant fragments when the number of recommended fragments increase. These results show that our approach is *accurate* enough to be applied in real use-cases.

7.3.3 Solving optimal cloud resource allocation to business process variants

To validate our cloud resource allocation approach, we present a comparison with LIP which is a broadly used mathematical optimization program [171]. For this end, we study the growth of our algorithm computation time comparing to the number of consumed resources per process variant. Concretely, we generate 5 variants from our configurable process model having number of assigned cloud resources per activity from 4 to 20. In addition, we maximize the inclusive choice (OR) resource operators

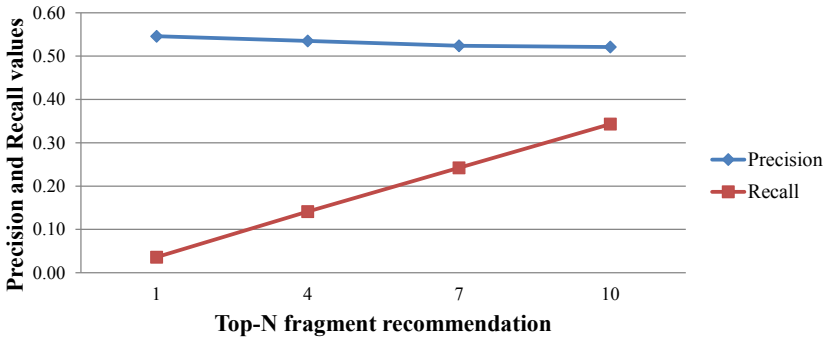


Figure 7.6: Precision and recall values

to increase the complexity. For example, having 8 cloud resources allocated to an activity can imply 255 possible resource allocation variants.

Fig. 7.7 depicts the results of our comparison. It shows that when the number of assigned cloud resources per activity is small (4-12), LIP marginally outperforms our genetic-based approach. However, when the number of cloud resources becomes higher (16-20), our genetic-based approach is able to keep its computation time almost constant while LIP performs an important exponential growth. For instance, if the activity allocates 20 resources, the computation time of our genetic algorithm is 700ms, whereas the computation time related to LIP algorithm reaches 14000ms.

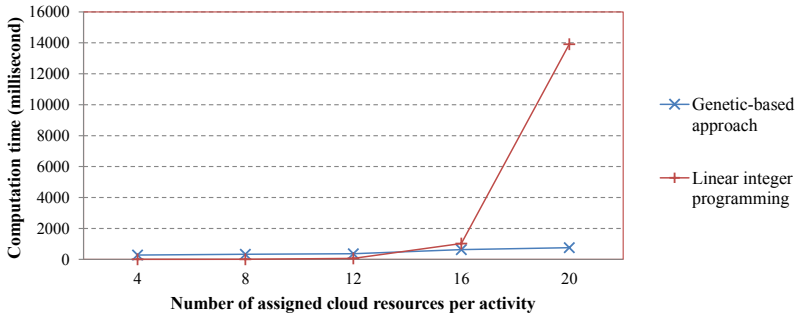


Figure 7.7: Comparing Genetic-based approach with Linear integer programming

Therefore, the LIP algorithm can be efficient only when the problem is not complex. We can conclude that our genetic-based approach should be preferred than LIP in case where have a large number of cloud resources that can be assigned to process activities. We refer herein to the case of large-scale service-based business processes.

7.3.4 Modeling cloud resources

In this section, we present the experiments to validate our approach in Chapter 5. We evaluate our proposed ontologies in qualitative aspect (Section 7.3.4.1) and quantitative aspect (Section 7.3.4.2).

7.3.4.1 Qualitative evaluation

We compare concepts in an ontology to classes in a standard to indicate the overall coverage and deviations to standards. We define four ontologies, namely sTOSCA, sOCCI, sCIMI, and Linked-CR based upon three standards: TOSCA, OCCI and CIMI. Hence, we evaluate our ontologies by analyzing their coverage to these standards. The results summarized in Table 7.3 show that we have a complete coverage of standard classes related to cloud resource at the infrastructure level.

Standard	Concepts	Covered	Ratio
TOSCA	11	11	100%
OCCI	11	11	100%
CIMI	8	8	100%
		Average	100%

Table 7.3: Ontology concept coverage to standards

We also evaluate the deviations, i.e., the difference of concepts between an ontology and a standard, of *Linked-CR* concepts comparing to these three standards. In fact, we have identified common concepts between Standard-specific ontologies when developing Linked-CR since it is an upper level representation of these ontologies. We found that Linked-CR cannot describe all the concepts defined in Standard-specific ontologies. For example, cr:NetworkPort represents common concepts st:Port and sc:NetworkPort. However, this resource type cannot be represented in sOCCI unless using so:Mixin to create a new resource instance for network port (cr, st, and so namespaces associate to Linked-CR, sTOSCA, and sOCCI, respectively).

Table 7.4 shows the deviations of concepts in Linked-CR comparing to concepts in standard-specific ontologies. The results show that we have a high coverage (on average 80.7%) of our common Linked-CR over standard-specific ontologies concepts. This good coverage indicates that Linked-CR although it drastically reduces the number of concepts (from 30 standard-specific ontologies concepts to 7 Linked-CR concepts), ensures minimum semantic loss.

7.3.4.2 Quantitative evaluation

In order to quantitatively evaluate our approach, we use ten use cases identified in the different standard specifications [9, 131, 135]: 4 use cases from TOSCA, 4

	Linked-CR		
Standard	Common concepts	Deviating concepts	Coverage ratio
TOSCA	9	2	81.8%
OCCI	8	3	72.7%
CIMI	7	1	87.5%
		Average	80.7%

Table 7.4: Linked-CR concept deviations from standards

use cases from OCCI and 2 use cases for CIMI. We use our prototype to construct RDF triples upon these use cases using the Standard-specific ontologies. Thereafter, we translate these RDF triples to other standards using predefined inference SWRL rules (Section 5.3.3). The number of RDF triples after translation and the coverage percentage of RDF triples by the translation rules are shown in Table 7.5. The results show that most of the triples are covered by the translation rules apart from the translation between sTOSCA and sOCCI because of their limitation. We also found that all use cases are completely translated to Linked-CR which means the deviating concepts (Table 7.4) are not often used (e.g., cr:NetworkPort and st:Port). In conclusion, our evaluation shows that practically Linked-CR is semantically rich enough to model heterogeneous cloud resources. We publish use cases and their RDF triples at `http://www-inf.it-sudparis.eu/SIMBAD/tools/linked-cr/`.

	TOSCA use cases			
	sTOSCA	sOCCI	sCIMI	Linked-CR
No. of RDF triples	88	64	58	58
Coverage by the translation rules	100.0%	93.2%	100.0%	100.0%

	OCCI use cases			
	sOCCI	sTOSCA	sCIMI	Linked-CR
No. of RDF triples	35	27	29	29
Coverage by the translation rules	100.0%	85.2%	100.0%	100.0%

	CIMI use cases			
	sCIMI	sTOSCA	sOCCI	Linked-CR
No. of RDF triples	18	23	38	18
Coverage by the translation rules	100.0%	100.0%	100.0%	100.0%

Table 7.5: Numbers of RDF triples after translation

7.4 Conclusion

In this chapter, we presented the implementations and experiments conducted to validate our contribution. Two applications have been implemented proving that our approach is feasible.

Regarding the modeling and mining of process data (i.e., BP models and process event logs) in a knowledge base, we performed experiments on large datasets. Experimental results showed that our approach recommends comprehensible BP fragments of low complexity. They also showed that the recommended BP fragments are accurate.

Regarding our genetic-based approach to solve optimal cloud resource allocation to BP variants, we compare our proposed approach with a broadly used mathematical optimization program, i.e., LIP. The results showed that our approach should be preferred than LIP in when having a large number of cloud resources that can be assigned to BP activities.

For modeling heterogeneous cloud resources in our knowledge base, we firstly compared our proposed ontologies to standards. The results showed that we have a complete coverage. Thereafter, we applied our approach on use cases identified in different cloud resource description standards. Experimental results showed that our approach is adequate for describing and translating descriptions of heterogeneous cloud resources.

Chapter 8

Conclusion and Future Works

The research problem of this thesis is expressed by this question: *How to support business process variant design in cloud environments?* Previous chapters presented our solutions to answer the raised interrogation. In this chapter, we summarize our work (Section 8.1) and present the future work (Section 8.2).

8.1 Contributions

More and more organizations are adopting PAIS on cloud environments, especially large organizations having multiple branches situated in different regions. Such organizations need to support many BP variants and provision large amount of cloud resources. Thus, BP models need to be flexibly adjustable to derive individual BP variants that suit particular requirements. In addition, cloud resources consumed by these BP variants need to be effectively allocated and shared within an organization to optimize costs. These challenges are considered in many researches at both academic and industry level.

Contemporary approaches tackle these challenges in different ways. They have proposed to support BP design by using process templates or finding similar BPs using graph matching. However, theses approaches mainly study conceptual BP models and take into account the entire BP topology which is labor-intensive and often the cause for error-prone and time consuming BP variant design. Moreover, they do not consider the allocation of cloud resources consumed by BP variants. To address these issues, we proposed in this thesis an approach to support BP variant design by exploiting heterogeneous process data (i.e., BP models and process event logs). Furthermore, we proposed an approach to optimally allocate cloud resources to BP variants using genetic algorithm.

In this thesis, we addressed the problems of modeling heterogeneous process data and cloud resources by using ontologies. We reused and extended ontologies from the European project SUPER to describe heterogeneous process data. These heterogeneous process data are represented using ontologies and stored in a shared knowledge base to offer means for reusing these data.

We proposed to use our knowledge base as a data-source for assisting BP variant design. To do so, we recommend relevant BP fragments (i.e., a set of activities and their relations) for selected positions on an under design BP model.

We proposed to compute the similarity between BP fragments by matching their neighborhood contexts. The recommendation results are based on these computed similarities. Our approach extracts fragments from BP models and process event logs. Thereafter, our approach assists finding suitable BP fragments that can be plugged-in into selected positions of BP models.

We have also developed ontologies to describe cloud resources based on different cloud resource description standards. These heterogeneous cloud resources are stored in a shared knowledge base to offer means for reusing.

We proposed to adopt a genetic algorithm to ensure the optimal cloud resource allocation to BP variants. We proposed a modeling operator to describe possible cloud resource allocations to BP activities. We derived possible variants of cloud resource allocation from this operator. Thereafter, we adopt a genetic algorithm by encoding such variants to a genome in the form of array which represent individuals. We apply genetic operators (i.e., crossover, and mutation) to randomly generate new individuals (i.e., cloud resource allocation variant). Each individual is evaluated by a fitness function that represents the degree of goodness. Finally, we identify the individual (i.e., cloud resource allocation variant) with the best fitness value as an optimal solution.

In order to validate our approach, we have developed two applications: (*i*) BP variant design support application, and (*ii*) cloud resource knowledge base populating application. The first application demonstrates modeling heterogeneous process data using ontologies and recommendation technique for BP variant design. The second application demonstrates modeling heterogeneous cloud resources using ontologies. We performed different experiments on a real public dataset and use cases identified in different cloud resource description standards. Experimental results showed that our approach is feasible and accurate (according to obtained precision matrices).

Hence, the principles that we have set in Section 1.4.1 are respected:

- **Heterogeneous data modeling**: We proposed to adopt and extend existing ontologies to model heterogeneous process data. Moreover, we developed our ontologies based on different standards to describe heterogeneous cloud resources.

- **Automation**: We proposed an automated approach to construct a knowledge base from heterogeneous process data. Similar BP fragments can be automatically retrieved from our knowledge base to facilitate the design of BP variants. We also proposed a genetic-based approach to optimally allocate cloud resources to BP variants. Furthermore, we proposed an automated approach to construct a knowledge base of heterogeneous cloud resources using our ontologies.

- **Implicit knowledge exploitation**: We utilized existing and accessible process data shared within an organization as a data-source to design BP variants. Furthermore, we solved the optimal cloud resource allocation to BP variants using cloud resources and BP models shared within an organization.
- **Focused results**: We exploited the relation between activities to measure the similarity matrices between BP fragments. Therefore, we assist the design of BP variants by recommending BP fragments based on this similarity computation.

8.2 Future work

In the future work, we aim at improving the quality of automated support (Section 8.2.1) and at supporting cloud resource allocation for BPs (Section 8.2.2).

8.2.1 Improving the quality of automated support

Currently, our work takes into account the control-flow of BPs, execution paths of process event logs, and task functionality by its annotated specific-domain ontology. In our future work, we intend to extend our approach by taking into account other important perspectives such as the resource and data perspectives. We intend to associate our approach to existing works related to different perspectives, for example, resource modeling in BP [176] and BP role-based approaches [177, 178].

Moreover, we aim to improve our BP fragment matching by using multiple criteria. So far we consider that two tasks are similar in term of the functionality if they are semantically annotated with the same concept from a domain-specific ontology. In the future, we will construct a vector of task characteristics, including name, description, location, input, output, etc. Thereafter, the matching between two characteristic vectors will precisely improve our BP fragment similarity computation.

Currently, we match only connection flows having similar begin and end tasks in the same fragment layer (i.e., the number of tasks from the shortest path relation to the root task). We do not take into account the connection flow in different layers. This may miss some meaningful information that would improve the quality of our matching algorithm. Thus, we will extend our matching algorithm by matching connection flows from different layers.

8.2.2 Modeling for cloud resource allocation to business process variants

We have exploited the elasticity of cloud resources by effectively allocating heterogeneous cloud resources to BPs. Concretely, we have proposed a genetic-based approach to identify an optimal cloud resource allocations for BP variants. We have also proposed a semantic framework that populates a common knowledge base of heterogeneous BPs and cloud resources. However, the link (e.g., allocation and consumption)

between BPs and cloud resources have not been yet defined. Therefore, we aim to propose an ontology-based approach to model cloud resource allocations to BPs. Existing works have provided models for cloud resource perspective of BPs [179, 180]. Hence, we can integrate such models into our work for modeling BPs taking into account their cloud resources allocation. Thereafter, BPs and their cloud resource allocation could be shared and interoperated between an organization's branches.

Appendices

Appendix A

Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN), formerly known as Business Process Modeling Notation, has been introduced by Business Process Management [124]. BPMN provide a standard specification to create and document BP models. It is widely used in industries as the de facto BP modeling notation. BPMN in latest versions has been semantically enhanced to enable the execution of BP models.

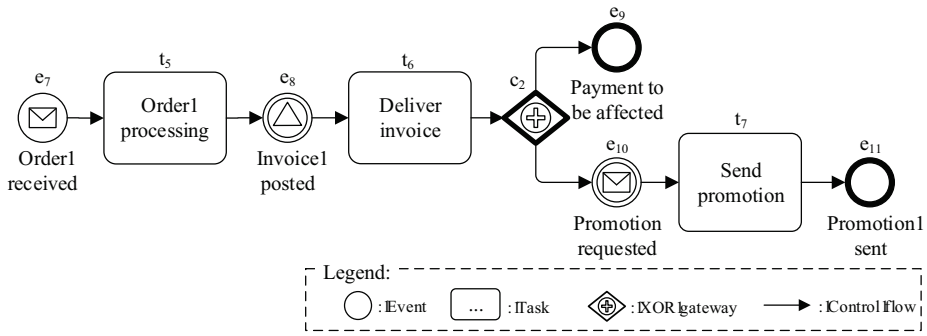


Figure A.1: Example: BPMN model

BPMN provides a rich set of elements to capture different perspectives of the BP. The BPMN elements can be categorized into four categories: (*i*) *flow objects*, (*ii*) *connecting objects*, (*iii*) *swimlanes*, and (*iv*) *artifacts*. *Flow Objects* represents the control flow perspective of a BP in terms of *activities*, *events*, *gateways*, and *sequence flow*. Figure A.1 depicts an example of BP at the control flow perspective. An *activity* is the main element of a BP model describing the task that must be done. It is graphically represented as a rectangle. An *event* represents something that happens during the execution of a BP which consists of three main types: *Start*, *Intermediate*, and *End event*. An *event* can also be specialized to other types, such as, *Message event* that can either send a message to a communication partner or react on the arrival of message, and *Error event* that reacts on a canceled transaction. An *event* is graphically represented with a circle. A *gateway* represents the split (from one control flow to two or more flows) and join (from two or more control flows to

one flow) paths in the BP model which consists of three main types: *AND* (parallel forking and synchronization), *XOR* (exclusive choice and merging) and *OR* (inclusive choice and merging). A gateway can also be specialized into other types, such as, *event-based gateway*, and *complex gateways*. However, they can still mapped to *AND*, *XOR*, and *OR*. The *flow objects* elements are connected through the *sequence flow* elements that determine the order of activities that will be performed in a BP model. Lastly, the *Artifacts*, *Swimlanes* and other elements represent the resource and data perspectives in the BP.

Appendix B

Event-driven Process Chain (EPC)

Event-driven Process Chain (EPC) is a widespread process modeling technique adopted by many Process-Aware Information Systems (PAISs), such as, Enterprise Resource Planning (ERP) systems, and Workflow Management (WFM) systems [125]. Originally, EPC has been introduced in the early 1990s in the framework of Architecture of Integrated Information Systems (ARIS) [181].

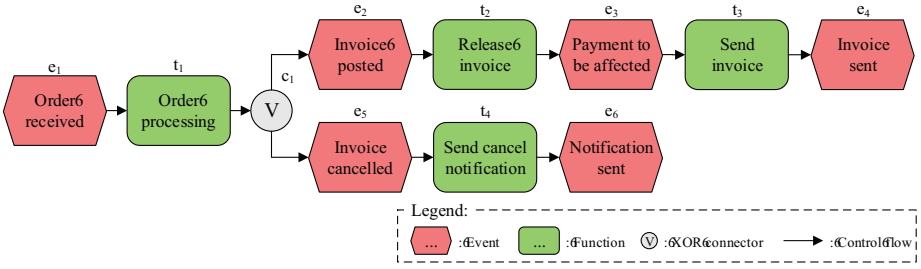


Figure B.1: Example: EPC model

The EPC elements can be categorized into many categories. However, the main control flow elements are categorized into four types: (*i*) *Function*, (*ii*) *Event*, (*iii*) *Logical connector*, and (*iv*) *Control flow*. An *event* represents a circumstance/state of a function/process. It is graphically represented with a hexagon. Typically, an EPC model starts with an event and ends with an event. A *function* represents the task or activity and describes a transformation from an initial state to a resulting state. It is graphically represented with a rounded rectangle. A logical connector represents a split from one control flow to two or more flows, and to synchronize from two or more control flows to one flow. It consists of three main types: *AND* (Fork/Join), *XOR* (Branch/Merge) and *OR*. Lastly, a control flow connects events with functions, or logical connectors creating sequence between them.

Appendix C

List of Publications

Journal Article

1. Karn Yongsiriwit, Nour Assy and Walid Gaaloul, *A Semantic Framework for Multi-tenant Business Process Design*, Journal of Networking and Computer Applications published by Elsevier 59, 168-184 (2016).

Conference Proceeding

1. Karn Yongsiriwit, Mohamed Sellami, Walid Gaaloul, *A Semantic Framework Supporting Business Process Variability Using Event Logs*, 13^{th} International Conference on Services Computing, IEEE SCC 2016, San Francisco, USA, June 27 - July 2, 2016.

2. Karn Yongsiriwit, Mohamed Sellami, Walid Gaaloul, *A Semantic Framework Supporting Cloud Resource Descriptions Interoperability*, 9^{th} International Conference on Cloud Computing, IEEE CLOUD 2016, San Francisco, USA, June 27 - July 2, 2016.

3. Emna Hachicha, Karn Yongsiriwit, Walid Gaaloul, *Energy Efficient Configurable Resource Allocation in Cloud-Based Business Processes*, 13^{th} International Conference on Cooperative Information Systems, CoopIS 2016, Rhodes, Greece, October 24 - 28, 2016.

4. Karn Yongsiriwit, Mohamed Sellami, Walid Gaaloul, *Semantic Process Fragments Matching to Assist the Development of Process Variants*, 12^{th} International Conference on Services Computing, IEEE SCC 2015, NY, USA, June 27 - July 2, 2015.

5. Karn Yongsiriwit, Nguyen Ngoc Chan, Walid Gaaloul, *Log-Based Process Fragment Querying to Support Process Design*, 48^{th} Hawaii International Conference on System Sciences, HICSS 2015, Kauai, Hawaii, USA, January 5-8, 2015.

6. Nguyen Ngoc Chan, Karn Yongsiriwit, Walid Gaaloul, Jan Mendling, *Mining Event Logs to Assist the Development of Executable Process Variants*, 26th International Conference on Advanced Information Systems Engineering, CAiSE 2014, Greece, June 16-20, 2014.

7. Nour Assy, Karn Yongsiriwit, Walid Gaaloul, Imen Grida Ben Yahia, *A Framework for Semantic Telco Process Management-An Industrial Case Study*, 14^{th} International Conference on Intelligent Systems Design and Applications, IEEE ISDA 2014, Okinawa, Japan, November 27-29, 2014.

Bibliography

[1] F. Gottschalk. *Configurable Process Models*. Phd thesis, Eindhoven University of Technology, December 2009.

[2] Remco Dijkman, Marlon Dumas, and Luciano Garcia-Banuelos. Graph matching algorithms for business process model similarity search. In *BPM '09*.

[3] Craig Schlenoff, Michael Gruninger, Taddle Creek, Mihai Ciocoiu, and Jintae Lee. The essence of the process specification language. *Transactions of the Society for Computer Simulation*, 16:204–16, 1999.

[4] Jintae Lee, Michael Gruninger, Yan Jin, Thomas W. Malone, Austin Tate, and Gregg Yost. The process interchange format and framework. *Knowledge Eng. Review*, 13(1):91–120, 1998.

[5] Owl-s: Semantic markup for web services. `https://www.w3.org/Submission/OWL-S/`.

[6] Yun Lin and John Krogstie. Semantic annotation of process models for facilitating process knowledge management. *IJISMD*, 1(3):45–67, 2010.

[7] Arne Sølvberg. *Data and What They Refer to*. Springer Berlin Heidelberg, 1999.

[8] Topology and orchestration specification for cloud applications version 1.0. `http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html`, January 2013.

[9] Open cloud computing interface - infrastructure (june 2011). `http://ogf.org/documents/GFD.184.pdf`.

[10] Antonucci Y.L. Using workflow technologies to improve organizational competitiveness. *International journal of management*, 14(1):117–126, 1997.

[11] Richard Lenz and Manfred Reichert. It support for healthcare processes - premises, challenges, perspectives. *Data Knowl. Eng.*, 61(1):39–58, April 2007.

[12] Marlon Dumas, Wil M. van der Aalst, and Arthur H. ter Hofstede. *Process-aware Information Systems: Bridging People and Software Through Process Technology*. John Wiley & Sons, Inc., New York, NY, USA, 2005.

[13] Wil M. P. van der Aalst. Process-aware information systems: Lessons to be learned from process mining. *Trans. Petri Nets and Other Models of Concurrency*, 2:1–26, 2009.

[14] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske. Business process management: A survey. In *Business Process Management, International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings*, pages 1–12, 2003.

[15] Mathias Weske. *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, 2012.

[16] Howard Smith and Peter Fingar. *Business process management: the third wave*. Springer, 2003.

[17] Wil M. P. van der Aalst and Kees M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.

[18] Dimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distrib. Parallel Databases*, 3(2):119–153, April 1995.

[19] Pernille Kræmmergaard Jensen, Harry Boer, and Charles Møller. *ERP implementation: an integrated process of radical change and continuous learning*. 2001.

[20] Becker J., Kugeler M., and Rosemann M. *Process Management: A guide for he design of business processes*. Springer, 2003.

[21] T. Panagacos. *The Ultimate Guide to Business Process Management: Everything You Need to Know and How to Apply It to Your Organization*. CreateSpace Independent Publishing Platform, 2012.

[22] W. M. P. van der Aalst and A. H. M. ter Hofstede. Yawl: Yet another workflow language. *Inf. Syst.*, 30(4):245–275, June 2005.

[23] OMG. Business process model and notation (bpmn) 2.0. `http://www.omg.org/spec/BPMN/2.0/`.

[24] v2.3 OMG. Unified modelling language. http://www.omg.org/spec/uml/2.3/.

[25] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[26] Wil M. P. van der Aalst. Challenges in business process analysis. In *9th International Conferenc on Enterprise Information Systems (Selected Papers)*, pages 27–42. Springer, 2007.

[27] Wil M. P. van der Aalst, Hajo A. Reijers, A. J. M. M. Weijters, Boudewijn F. van Dongen, Ana Karla Alves de Medeiros, Minseok Song, and H. M. W. (Eric) Verbeek. Business process mining: An industrial application. *Inf. Syst.*, 32(5):713–732, 2007.

[28] Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[29] Nguyen Ngoc Chan, Karn Yongsiriwit, Walid Gaaloul, and Jan Mendling. Mining event logs to assist the development of executable process variants. In *CAiSE 2014*, pages 548–563, 2014.

[30] Karn Yongsiriwit, Nguyen Ngoc Chan, and Walid Gaaloul. Log-based process fragment querying to support process design. In *HICSS 2015*, pages 4109–4119, 2015.

[31] W.M.P. van der Aalst. Business process configuration in the cloud: How to support and analyze multi-tenant processes? *Web Services, European Conference on*, 0:3–10, 2011.

[32] Aaron Weiss. Computing in the clouds. *netWorker*, 11(4):16–25, December 2007.

[33] Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. Technical report, 2011.

[34] Gartner. Forecast overview: Public cloud services, worldwide, 2011-2016, 2q12 update. Technical report, 2012.

[35] Jörg Becker, Martin Kugeler, and Michael Rosemann. *Process management : a guide for the design of business processes / Jörg Becker, Martin Kugeler, Michael Rosemann, editors*. Springer Berlin ; Heidelberg ; New York, 2003.

[36] Rania Khalaf. From rosettanet pips to bpel processes: a three level approach for business protocols. In *BPM '05*, 2005.

[37] Peter Fettke and Peter Loos. Classification of reference models: a methodology and its application. *Information Systems and eBusiness Management*, 2003.

[38] Remco M. Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph matching algorithms for business process model similarity search. In *BPM 2009*, pages 48–63, 2009.

[39] Boudewijn F. van Dongen, Remco M. Dijkman, and Jan Mendling. Measuring similarity between business process models. In *Seminal Contributions to Information Systems Engineering*, pages 405–419. 2013.

[40] Matthias Kunze and Mathias Weske. Metric trees for efficient similarity search in large process model repositories. In *BPM 2010*, pages 535–546, 2010.

[41] Zhiqiang Yan, Remco M. Dijkman, and Paul W. P. J. Grefen. Fast business process similarity search. *Distributed and Parallel Databases*, 30(2):105–144, 2012.

[42] Wil M. P. van der Aalst, Niels Lohmann, Marcello La Rosa, and Jingxin Xu. Correctness ensuring process configuration: An approach based on partner synthesis. In *Business Process Management - 8th International Conference, BPM 2010, Hoboken, NJ, USA, September 13-16, 2010. Proceedings*, pages 95–111, 2010.

[43] Thomas Curran, Gerhard Keller, and Andrew Ladd. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. 1998.

[44] Scott Stephens. Supply chain operations reference model version 5.0: A new tool to improve supply chain efficiency and achieve best practice. *Information Systems Frontiers*, 3(4):471–476, 2001.

[45] Boudewijn F. van Dongen, Jan Mendling, and Wil M. P. van der Aalst. Structural patterns for soundness of business process models. In *EDOC 2006*, pages 116–128, 2006.

[46] Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.

[47] Dana Petcu. Portability and interoperability between clouds: Challenges and case study - (invited paper). In *ServiceWave 2011*, pages 62–74, 2011.

[48] Ieee, standard computer dictionary - a compilation of ieee standard computer glossaries, 1990.

[49] Europeen Comission. European interoperability framework for pan-european egovernment services. *IDA working document*, 2, 2004.

[50] Line Pouchard and Agent based Systems. Ontology engineering for distributed collaboration in manufacturing.

[51] Oliver Thomas and Michael Fellmann. Semantic business process management: Ontology-based process modeling using event-driven process chains. pages 29–44, 2007.

[52] Andreas Bögl, Michael Schrefl, Gustav Pomberger, and Norbert Weber. Semantic annotation of epc models in engineering domains by employing semantic patterns. In *ICEIS (2)*, pages 106–115, 2008.

[53] Agata Filipowska, Monika Kaczmarek, and Sebastian Stein. Semantically annotated EPC within semantic business process management. In *Business Process Management Workshops, BPM 2008 International Workshops, Milano, Italy, September 1-4, 2008. Revised Papers*, pages 486–497, 2008.

[54] Yun Lin and Hao Ding. Ontology-based semantic annotation for semantic interoperability of process models. In *2005 International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA 2005), International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC 2005), 28-30 November 2005, Vienna, Austria*, pages 162–167, 2005.

[55] Witold Abramowicz, Agata Filipowska, Monika Kaczmarek, and Tomasz Kaczmarek. Semantically enhanced business process modelling notation. In *SBPM*, 2007.

[56] Liliana Cabral, Barry Norton, and John Domingue. The business process modelling ontology. In *Proceedings of the 4th International Workshop on Semantic Business Process Management*, SBPM '09, pages 9–16, 2009.

[57] Super (semantics utilised for process management within and between enterprises) integrated project. `http://projects.kmi.open.ac.uk/super/`.

[58] Oliver Thomas and Michael Fellmann. Semantic process modeling - design and implementation of an ontology-based representation of business processes. *Business & Information Systems Engineering*, 1(6):438–451, 2009.

[59] Yves Raimond and Samer Abdallah. The event ontology (2007). `http://purl.org/NET/c4dm/event.owl`.

[60] Alejandro Rodríguez, Robert E. McGrath, Yong Liu, and James D. Myers. Semantic management of streaming data. In *ISWC 2009*, pages 80–95, 2009.

[61] Carlos Pedrinaci and John Domingue. Towards an ontology for process monitoring and mining. In *SBPM 2007*, 2007.

[62] A. J. M. M. Weijters and A. K. Alves De Medeiros. Process mining with the heuristicsminer algorithm.

[63] Wil M. P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. Causal nets: A modeling language tailored towards process discovery. In *CONCUR 2011*, pages 28–42, 2011.

[64] Grace A. Lewis. The role of standards in cloud- computing interoperability. Technical report, 2012.

[65] Fotis Gonidis, Anthony J. H. Simons, Iraklis Paraskakis, and Dimitrios Kourtesis. Cloud application portability: An initial view. In *Proceedings of the 6th Balkan Conference in Informatics*, BCI '13, pages 275–282, 2013.

[66] A V Parameswaran and Asheesh Chaddha. Cloud interoperability and standardization. *19 SETLabs Briefings*, 7, 2009.

[67] Jingxin K. Wang, Jianrui Ding, and Tian Niu. Interoperability and standardization of intercloud cloud computing. *CoRR*, abs/1212.5956, 2012.

[68] Mohamed Sellami, Sami Yangui, Mohamed Mohamed, and Samir Tata. Paas-independent provisioning and management of applications in the cloud. In *2013 IEEE Cloud, Santa Clara, CA, USA, June 28 - July 3, 2013*, pages 693–700, 2013.

[69] Reza Rezaei, Thiam K. Chiew, Sai P. Lee, and Zeinab S. Aliee. A Semantic Interoperability Framework for Software as a Service Systems in Cloud Computing Environments. *Expert Systems with Applications*, 2014.

[70] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. *CoRR*, 2010.

[71] Topology and orchestration specification for cloud applications (tosca). `https://www.oasis-open.org/committees/tosca/`.

[72] Open cloud computing interface (occi). `http://occi-wg.org/`.

[73] Cloud infrastructure management interface (cimi). `https://www.dmtf.org/standards/cloud`.

[74] Jean-Philippe Martin-Flatin. Challenges in cloud management. *IEEE Cloud Computing*, 1(1):66–70, 2014.

[75] Gerhard Keller and Thomas Teufel. *Sap R/3 Process Oriented Implementation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1998.

[76] Dirk Fahland, Cédric Favre, Barbara Jobstmann, Jana Koehler, Niels Lohmann, Hagen Völzer, and Karsten Wolf. Instantaneous soundness checking of industrial business process models. In *7th BPM*, pages 278–293, 2009.

[77] GeorgeM. Giaglis. A taxonomy of business process modeling and information systems modeling techniques. *International Journal of Flexible Manufacturing Systems*, 13(2):209–228, 2001.

[78] Wil M. P. van der Aalst, Marlon Dumas, Florian Gottschalk, Arthur H. M. ter Hofstede, Marcello La Rosa, and Jan Mendling. Preserving correctness during business process model configuration. *Formal Asp. Comput.*, 22(3-4):459–482, 2010.

[79] Stefanie Rinderle, Manfred Reichert, and Peter Dadam. Flexible support of team processes by adaptive workflow systems. *Distributed and Parallel Databases*, 16(1):91–116, 2004.

[80] Gabriel Hermosillo, Lionel Seinturier, and Laurence Duchien. Using Complex Event Processing for Dynamic Business Process Adaptation. In *IEEE International Conference on Services Computing*, pages 466–473, 2010.

[81] Marcello La Rosa, Marlon Dumas, Arthur H. M. ter Hofstede, and Jan Mendling. Configurable multi-perspective business process models. *Inf. Syst.*, 36(2):313–340, 2011.

[82] Thomas Curran, Gerhard Keller, and Andrew Ladd. *SAP R/3 business blueprint: understanding the business process reference model*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[83] Scott Stephens. Supply chain operations reference model version 5.0: A new tool to improve supply chain efficiency and achieve best practice. *Information Systems Frontiers*, 3:471–476, December 2001.

[84] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recogn. Lett.*, 18(9):689–694, August 1997.

[85] Zhiqiang Yan, Remco Dijkman, and Paul Grefen. Fast business process similarity search with feature-based similarity estimation. In *Proceedings of the 2010 international conference on On the move to meaningful internet systems - Volume Part I*, OTM'10, pages 60–77, Berlin, Heidelberg, 2010. Springer-Verlag.

[86] Chen Li, Manfred Reichert, and Andreas Wombacher. On measuring process model similarity based on high-level change operations. In *Proceedings of the 27th International Conference on Conceptual Modeling*, ER '08, pages 248–264, Berlin, Heidelberg, 2008. Springer-Verlag.

[87] Marc Ehrig, Agnes Koschmider, and Andreas Oberweis. Measuring similarity between semantic business process models. In *Proceedings of the fourth Asia-Pacific conference on Comceptual modelling - Volume 67*, APCCM '07, pages 71–80, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[88] W.M.P. Aalst, A.K.Alves Medeiros, and A.J.M.M. Weijters. Process equivalence: Comparing two process models based on observed behavior. In Schahram Dustdar, JosÃ©Luiz Fiadeiro, and AmitP. Sheth, editors, *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 129–144. Springer Berlin Heidelberg, 2006.

[89] Boudewijn Dongen, Remco Dijkman, and Jan Mendling. Measuring similarity between business process models. In *Proceedings of the 20th international conference on Advanced Information Systems Engineering*, CAiSE '08, pages 450–464, Berlin, Heidelberg, 2008. Springer-Verlag.

[90] VI Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, 1966.

[91] Mohammad Abdulkader Abdulrahim. *Parallel algorithms for labeled graph matching*. PhD thesis, Golden, CO, USA, 1998. AAI0599838.

[92] Ahmed Awad. Bpmn-q: A language to query business processes. In *EMISA*, pages 115–128, 2007.

[93] Ahmed Awad, Artem Polyvyanyy, and Mathias Weske. Semantic querying of business process models. In *Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference*, pages 85–94, Washington, DC, USA, 2008. IEEE Computer Society.

[94] Ahmed Awad, Gero Decker, and Mathias Weske. Efficient compliance checking using bpmn-q and temporal logic. In *Proceedings of the 6th International Conference on Business Process Management*, BPM '08, pages 326–341, Berlin, Heidelberg, 2008. Springer-Verlag.

[95] Sherif Sakr and Ahmed Awad. A framework for querying graph-based business process models. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1297–1300, New York, NY, USA, 2010. ACM.

[96] Sherif Sakr, Emilian Pascalau, Ahmed Awad, and Mattias Weske. Partial process models to manage business process variants. *International Journal of Business Process Integration and Management (IJBPIM)*, 6(2):20, September 2011.

[97] Thomas Hornung, Agnes Koschmider, and Georg Lausen. Recommendation based process modeling support: Method and user experience. In *Proceedings of the 27th International Conference on Conceptual Modeling*, ER '08, pages 265–278, Berlin, Heidelberg, 2008. Springer-Verlag.

[98] Maya Lincoln, Mati Golani, and Avigdor Gal. Machine-assisted design of business process models using descriptor space analysis. In *Proceedings of the 8th international conference on Business process management*, BPM'10, pages 128–144, Berlin, Heidelberg, 2010. Springer-Verlag.

[99] A. J. M. M. Weijters and W. M. P. van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integr. Comput.-Aided Eng.*, 10(2):151–162, April 2003.

[100] Wil van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. on Knowl. and Data Eng.*, 16(9):1128–1142, September 2004.

[101] FabrizioM. Maggi, R.P.JagadeeshChandra Bose, and WilM.P. Aalst. Efficient discovery of understandable declarative process models from event logs. In Jolita Ralyte, Xavier Franch, Sjaak Brinkkemper, and Stanislaw Wrycza, editors, *Advanced Information Systems Engineering*, volume 7328 of *Lecture Notes in Computer Science*, pages 270–285. Springer Berlin Heidelberg, 2012.

[102] Robert Engel, Wil van der Aalst, Marco Zapletal, Christian Pichler, and Hannes Werthner. Mining inter-organizational business process models from edi messages: A case study from the automotive sector. In Jolita RalytÃ©, Xavier Franch, Sjaak Brinkkemper, and Stanislaw Wrycza, editors, *Advanced Information Systems Engineering*, volume 7328 of *Lecture Notes in Computer Science*, pages 222–237. Springer Berlin / Heidelberg, 2012.

[103] Wil M. P. van der Aalst. Process discovery: capturing the invisible. *Comp. Intell. Mag.*, 5(1):28–41, February 2010.

[104] A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95, March 2008.

[105] Wil M. P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

[106] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst. Conformance checking using cost-based fitness analysis. In *Proceedings of the 2011 IEEE 15th International Enterprise Distributed Object Computing Conference*, EDOC '11, pages 55–64, Washington, DC, USA, 2011. IEEE Computer Society.

[107] Dirk Fahland, Massimiliano De Leoni, Boudewijn F. Van Dongen, and Wil M. P. Van Der Aalst. Conformance checking of interacting processes with overlapping instances. In *Proceedings of the 9th international conference on Business process management*, BPM'11, pages 345–361, Berlin, Heidelberg, 2011. Springer-Verlag.

[108] Jan Mendling, Gustaf Neumann, and Wil Van Der Aalst. Understanding the occurrence of errors in process models based on metrics. In *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I*, OTM'07, pages 113–130, Berlin, Heidelberg, 2007. Springer-Verlag.

[109] J. Mendling, H. M. W. Verbeek, B. F. van Dongen, W. M. P. van der Aalst, and G. Neumann. Detection and prediction of errors in epcs of the sap reference model. *Data Knowl. Eng.*, 64(1):312–329, January 2008.

[110] WilM.P. Aalst and Minseok Song. Mining social networks: Uncovering interaction patterns in business processes. In Jorg Desel, Barbara Pernici, and Mathias Weske, editors, *Business Process Management*, volume 3080 of *Lecture Notes in Computer Science*, pages 244–260. Springer Berlin Heidelberg, 2004.

[111] Wil M. P. Van Der Aalst, Hajo A. Reijers, and Minseok Song. Discovering social networks from event logs. *Comput. Supported Coop. Work*, 14(6):549–593, December 2005.

[112] Nguyen Ngoc Chan, Walid Gaaloul, and Samir Tata. Assisting business process design by activity neighborhood context matching. In *Service-Oriented Computing - 10th International Conference, ICSOC 2012, Shanghai, China, November 12-15, 2012. Proceedings*, pages 541–549, 2012.

[113] W.M.P. van der Aalst. Intra- and Inter-Organizational Process Mining: Discovering Processes within and between Organizations. In *PoEM '11*.

[114] Linda G. DeMichiel. Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *IEEE Trans. Knowl. Data Eng.*, 1(4):485–493, 1989.

[115] Christine Collet, Michael N. Huhns, and Wei-Min Shen. Resource integration using a large knowledge base in carnot. *IEEE Computer*, 24(12):55–62, 1991.

[116] Vipul Kashyap and Amit P. Sheth. Semantic and schematic similarities between database objects: A context-based approach. *VLDB J.*, 5(4):276–304, 1996.

[117] Jinsoo Park and Sudha Ram. Information systems interoperability: What lies beneath? *ACM Trans. Inf. Syst.*, 22(4):595–632, 2004.

[118] Thomas W. Malone, Kevin Crowston, and George A. Herman, editors. *Organizing Business Knowledge: The MIT Process Handbook*, volume 1. The MIT Press, 1 edition, 2003.

[119] Thomas W. Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26:87–119, 1994.

[120] Michael Genesereth, Richard E. Fikes, Ronald Brachman, Thomas Gruber, Patrick Hayes, Reed Letsinger, Vladimir Lifschitz, Robert Macgregor, John Mccarthy, Peter Norvig, and Ramesh Patil. Knowledge interchange format version 3.0 reference manual, 1992.

[121] P. Mika, D. Oberle, A. Gangemi, and M. Sabou. In S. Staab and P. Patel-Schneider, editors, *Proceedings of the World Wide Web Conference (WWW2004), Semantic Web Track*.

[122] W. M. P. van der Aalst, A. P. Barros, A. H. M. ter Hofstede, and B. Kiepuszewski. *Advanced Workflow Patterns*, pages 18–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[123] Yun Lin. *Semantic Annotation for Process Models : Facilitating Process Knowledge Management via Semantic Interoperability.* PhD thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, 2008.

[124] OMG Business Process Model and Notation. `http://www.omg.org/spec/BPMN/2.0.2/`.

[125] August-Wilhelm Scheer. *ARIS - vom Geschäftsprozess zum Anwendungssystem.* Springer, 4., durchges. aufl. edition, 2002.

[126] Martin Hepp, Knut Hinkelmann, Dimitris Karagiannis, Rüdiger Klein, and Nenad Stojanovic, editors. *SBPM 2007*, volume 251 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[127] Super specification: sbpmn and sepc to bpmo translation. `http://www-inf.it-sudparis.eu/SIMBAD/SUPER/D4.5.doc`.

[128] Deborah L. Mcguinness, Richard Fikes, James Rice, and Steve Wilder. An environment for merging and testing large ontologies. pages 483–493. Morgan Kaufmann, 2000.

[129] Boudewijn F. van Dongen and Wil M. P. van der Aalst. A meta model for process mining data. In *EMOI - INTEROP'05, Porto (Portugal), 13th-14th June 2005*, 2005.

[130] Tobias Binz, Gerd Breiter, Frank Leymann, and Thomas Spatzier. Portable cloud services using TOSCA. *IEEE Internet Computing*, 16(3):80–85, 2012.

[131] Tosca simple profile in yaml version 1.0 (nov. 2013). `http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/TOSCA-Simple-Profile-YAML-v1.0.pdf`.

[132] Open cloud computing interface - core (june 2011). `http://ogf.org/documents/GFD.183.pdf`.

[133] Cloud infrastructure management interface (cimi) model and restful http-based protocol (mar 2015). `https://www.dmtf.org/sites/default/files/standards/documents/DSP0263_2.0.0c.pdf`.

[134] Cimi xml schema (feb 2014). `https://www.dmtf.org/standards/cloud`.

[135] Cloud infrastructure management interface (cimi) use cases (dec 2014). `https://www.dmtf.org/standards/cloud`.

[136] Nikolaos Loutas, Vassilios Peristeras, Thanassis Bouras, Eleni Kamateri, Dimitrios Zeginis, and Konstantinos A. Tarabanis. Towards a reference architecture for semantically interoperable clouds. In *CloudCom 2010, November 30 - December 3, 2010, Indianapolis, Indiana, USA, Proceedings*, pages 143–150, 2010.

[137] Eleni Kamateri, Nikolaos Loutas, Dimitris Zeginis, James Ahtes, Francesco D'Andria, Stefano Bocconi, Panagiotis Gouvas, Giannis Ledakis, Franco Ravagli, Oleksandr Lobunets, and Konstantinos A. Tarabanis. Cloud4soa: A semantic-interoperability paas solution for multi-cloud platform management and portability. In *ESOCC 2013, Málaga, Spain, September 11-13, 2013. Proceedings*, pages 64–78, 2013.

[138] Deepak K. Vij and David Bernstein. Draft standard for intercloud interoperability and federation (siif). In *IEEE P2303*. 2014.

[139] Beniamino Di Martino, Giuseppina Cretella, Antonio Esposito, A. Willner, A. Alloush, David Bernstein, Deepak Vij, and J. Weinman. Towards an ontology-based intercloud resource catalogue. In *IEEE IC2E 2015, Tempe, AZ, USA, March 9-13, 2015*, pages 458–464, 2015.

[140] Sami Yangui, Iain James Marshall, Jean-Pierre Laisné, and Samir Tata. Compatibleone: The open source cloud broker. *J. Grid Comput.*, 12(1):93–109, 2014.

[141] Radu Prodan and Simon Ostermann. A survey and taxonomy of infrastructure as a service and web hosting cloud providers. In *10th IEEE/ACM International Conference on Grid Computing, October 13-15, 2009, Banff, Alberta, Canada*, pages 17–25, 2009.

[142] Ralf Teckelmann, Christoph Reich, and Anthony Sulistio. Mapping of cloud standards to the taxonomy of interoperability in iaas. In *IEEE CloudCom 2011, Athens, Greece, November 29 - December 1, 2011*, pages 522–526, 2011.

[143] Open virtualization format white paper, distributed management task force, apr. 2014, dsp2017 version 2.0.0. `https://www.dmtf.org/sites/default/files/standards/documents/DSP2017_2.0.0.pdf`.

[144] Cloud data management interface (cdmi), snia technical position, june 2012, version 1.0.2. `http://www.snia.org/sites/default/files/CDMI20v1.0.2.pdf`.

[145] Thamarai Selvi Somasundaram, Kannan Govindarajan, Usha Kiruthika, and Rajkumar Buyya. Semantic-enabled CARE resource broker (secrb) for managing grid and cloud environment. *The Journal of Supercomputing*, 68(2):509–556, 2014.

[146] Balachandar R. Amarnath, Thamarai Selvi Somasundaram, Mahendran Ellappan, and Rajkumar Buyya. Ontology-based grid resource management. *Softw., Pract. Exper.*, 39(17):1419–1438, 2009.

[147] Pierfrancesco Bellini, Daniele Cenni, and Paolo Nesi. Smart cloud engine and solution based on knowledge base. In *Procedia Comput. Sci., vol. 68*, pages 3–16.

[148] Beniamino Di Martino, Giuseppina Cretella, Antonio Esposito, and Graziella Carta. An OWL ontology to support cloud portability and interoperability. *IJWGS*, 11(3):303–326, 2015.

[149] Stefan Schulte, Christian Janiesch, Srikumar Venugopal, Ingo Weber, and Philipp Hoenisch. Elastic business process management: State of the art and open challenges for BPM in the cloud. *Future Generation Comp. Syst.*, 46:36–50, 2015.

[150] Evert Ferdinand Duipmans. Business process management in the cloud: Business process as a service (bpaas). In *2012*.

[151] MingXue Wang, Kosala Yapa Bandara, and Claus Pahl. Process as a service. In *2010 IEEE International Conference on Services Computing, SCC 2010, Miami, Florida, USA, July 5-10, 2010*, pages 578–585, 2010.

[152] Cristina Cabanillas and et al. Towards process-aware cross-organizational human resource management. In *Enterprise, Business-Process and Information Systems Modeling - 15th International Conference, BPMDS 2014, 19th International Conference, EMMSAD, Held at CAiSE, Thessaloniki, Greece, June 16-17.*, pages 79–93, 2014.

[153] Cristina Cabanillas and et al. Ralph: A graphical notation for resource assignments in business processes. In *Advanced Information Systems Engineering - 27th International Conference, CAiSE, Stockholm, Sweden, June 8-12*, pages 53–68, 2015.

[154] Philipp Hoenisch, Stefan Schulte, Schahram Dustdar, and Srikumar Venugopal. Self-adaptive resource allocation for elastic process execution. In *2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, June 28 - July 3, 2013*, pages 220–227, 2013.

[155] Philipp Hoenisch and et al. Workflow scheduling and resource allocation for cloud-based execution of elastic processes. In *IEEE 6th International Conference on Service-Oriented Computing and Applications, Koloa, USA, December 16-18*, pages 1–8, 2013.

[156] Stefan Schulte, Dieter Schuller, Phillipp Hoenisch, Ulrich Lampe, Schahram Dustdar, and Ralf Steinmetz. Cost-driven optimization of cloud resource allocation for elastic processes. *International Journal of Cloud Computing*, 1(2):1–15, 2013.

[157] Karn Yongsiriwit, Mohamed Sellami, and Walid Gaaloul. Semantic process fragments matching to assist the development of process variants. In *SCC 2015*, 2015.

[158] Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco M. Dijkman. Business process model merging: An approach to business process consolidation. page 11, 2013.

[159] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[160] W3c. rdf vocabulary description language 1.0: Rdf schema. `http://www.w3.org/TR/rdf-schema/`.

[161] Semantic web rule language. `http://www.w3.org/Submission/SWRL/`.

[162] Karn Yongsiriwit, Mohamed Sellami, and Walid Gaaloul. A semantic framework supporting business process variability using event logs. In *IEEE International Conference on Services Computing, SCC 2016, San Francisco, CA, USA, June 27 - July 2, 2016*, pages 163–170, 2016.

[163] H.M.W. Verbeek, J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. XES, XESame, and ProM 6. In P. Soffer and E. Proper, editors, *Information Systems Evolution*, volume 72 of *Lecture Notes in Business Information Processing*, pages 60–75. Springer-Verlag, Berlin, 2010.

[164] Karn Yongsiriwit, Mohamed Sellami, and Walid Gaaloul. A semantic framework supporting cloud resource descriptions interoperability. In *IEEE International Conference on Cloud Computing, 2016, San Francisco, CA, USA, June 27 - July 2, 2016*, 2016.

[165] Open cloud computing interface - restful http rendering (june 2011). `http://ogf.org/documents/GFD.185.pdf`.

[166] Emna Hachicha, Karn Yongsiriwit, and Walid Gaaloul. Energy efficient configurable resource allocation in cloud-based business processes (short paper).

In *On the Move to Meaningful Internet Systems: OTM 2016 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2016, Rhodes, Greece, October 24-28, 2016, Proceedings*, pages 437–444, 2016.

[167] Emna Hachicha, Nour Assy, Walid Gaaloul, and Jan Mendling. A configurable resource allocation for multi-tenant process development in the cloud. In *Advanced Information Systems Engineering - 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings*, pages 558–574, 2016.

[168] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[169] S. N. Sivanandam and S. N. Deepa. *Introduction to genetic algorithms*. Springer, 2008.

[170] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85.

[171] Srinivas Talluri and R. C. Baker. A multi-phase mathematical programming approach for effective supply chain design. *European Journal of Operational Research*, 141(3):544–558, 2002.

[172] Virtuoso jena provider. `http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtJenaProvider`.

[173] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. Technical report, W3C, 2004.

[174] Jan Mendling and Markus Nüttgens. EPC markup language (EPML): an xml-based interchange format for event-driven process chains (epc). *Inf. Syst. E-Business Management*, 4(3):245–263, 2006.

[175] Marta Sabou and Miriam Fernández. Ontology (network) evaluation. In *Ontology Engineering in a Networked World.*, pages 193–212. 2012.

[176] Nick Russell, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and David Edmond. Workflow resource patterns: Identification, representation and tool support. In *Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005, Proceedings*, pages 216–232, 2005.

[177] David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[178] Elisa Bertino, Elena Ferrari, and Vijayalakshmi Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, 2(1):65–104, 1999.

[179] Philipp Hoenisch, Christoph Hochreiner, Dieter Schuller, Stefan Schulte, Jan Mendling, and Schahram Dustdar. Cost-efficient scheduling of elastic processes in hybrid clouds. In *8th IEEE International Conference on Cloud Computing, CLOUD 2015, New York City, NY, USA, June 27 - July 2, 2015*, pages 17–24, 2015.

[180] Emna Hachicha and Walid Gaaloul. Towards resource-aware business process development in the cloud. In *29th IEEE International Conference on Advanced Information Networking and Applications, AINA 2015, Gwangju, South Korea, March 24-27, 2015*, pages 761–768, 2015.

[181] August-Wilhelm Scheer. *ARIS - vom Geschäftsprozess zum Anwendungssystem.* Springer, Berlin [u.a.], 4., durchges. aufl. edition, 2002.