Ontology-Based Description and Discovery of Business Processes

Khalid Belhajjame[1], Marco Brambilla[2]

[1] School of Computer Science, University of Manchester,
Oxford Road, M13 9PL - United Kingdom
[2] Politecnico di Milano, Dipartimento di Elettronica e Informazione
P.za L. Da Vinci, 32. I-20133 Milano - Italy
`Khalid.Belhajjame@manchester.ac.uk, Marco.Brambilla@polimi.it`

Abstract. Just like web services, business processes can be stored in public repositories to be shared and used by third parties, e.g., as building blocks for constructing new business processes. The success of such a paradigm depends partly on the availability of effective search tools to locate business processes that are relevant to the user purposes. A handful of researchers have investigated the problem of business process discovery using as input syntactical and structural information that describes business processes. In this work, we explore an additional source of information encoded in the form of annotations that semantically describe business processes. Specifically, we show how business processes can be semantically described using the so called *abstract business processes.* These are designated by concepts from an ontology which additionally captures their relationships. We show how this ontology can be built in an automatic fashion from a collection of (concrete) business processes, and we illustrate how it can be refined by domain experts and used in the discovery of business processes, with the purpose of reuse and increase in design productivity.

1 Introduction

The last two decades showed that *business process modeling (BPM)* is the solution of choice of multiple companies and government institutions for describing and enacting their internal and external work procedures. Generally speaking, a business process is modelled as a series of activities connected together using data and control dependencies. Once modelled, business processes can be made available either publicly or accessible to a specific community to share the know-how between institutions and promote the reuse of existing business processes, e.g., as building blocks for constructing new business processes. The success of such a paradigm depends partly on the availability of a means by which users can locate business processes that are relevant for their purposes.

A handful of researchers have investigated the problem of business process reuse based on similarity and repository management. Eyal *et al.* proposed a visual query language for discovering business processes modelled using BPEL [3].

Goderis *et al.* developed a framework for discovering workflows using similarity metrics that consider the activities composing the workflows and their relationships [6]. Corrales *et al.* developed a tool for comparing the controlflow of business processes [5].

The above solutions to business process discovery use as input the workflows that model the activities that constitute the business processes and their dependencies in term of controlflow. Yet, a workflow is not a complete description of the business processes. In this paper, we argue that a more effective discovery of business processes can be achieved if they are semantically described. Specifically, we show how such information can be encoded within an ontology that can be used for:

– *Abstracting discovery queries:* The user is able to formulate his/her queries in terms of the tasks (semantics) fulfilled by the desired business processes.
– *Exploiting relationships between business processes:* Business processes are inter-dependent. These dependencies can be explicitly described in the ontology in the form of binary relationships that can be used, amongst other things, for increasing the recall of discovery queries.

The paper is structured as follows. We introduce business processes and formally define the concept of abstract business process in Section 2. We present the ontology used for describing business processes in Section 3, and then show how it can be created and populated automatically in Section 4, starting from a set of concrete business process models. We show how the business process ontology can be used for discovering business processes in Section 5, and present a simple case study in order to exemplify and assess the effectiveness of our solution in Section 7. We compare our solution with existing related works in Section 8, and close the paper in Section 9.

2 Preliminaries

2.1 Business Process

A business process is a collection of interrelated tasks, which aim at solving a particular issue. It can be decomposed into several sub-processes, which have their own peculiarities, but also contribute to achieving the goal of the super-process. Execution of tasks is typically constrained by dependency rules among tasks, that consist of sequence constraints, branching and merging rules, pre- and post- conditions, event management points, and so on.

A business process can be specified by means of a workflow model, i.e., a visual representation of the correct sequence of tasks that leads to the achievement of the goal. The notations for workflow modelling provide the proper primitives for defining processes, tasks, actors, control flow and data flow between tasks. In our work, we will adopt a particular business process notation, namely BPMN (Business Process Management Notation) [4] and the terminology defined by the Workflow Management Coalition and the Business Process Management

Initiative and the concepts specified by BPDM (Business Process Definition Metamodel) [11], a platform- and notation- independent metamodel for defining business processes. However, we propose a general purpose approach, which is valid regardless of the adopted notation. The workflow model is based on the concepts of Process, Case (a process instance), Activity (the unit of work composing a process), Activity instance (an instantiation of an activity within a case), Actor (a user role intervening in the process), and Constraint (logical precedence and enabling rules for activities). Processes can be structured using a variety of control constructs: sequence, gateways implementing AND-splits (a single thread of control splits into two or more independent threads), AND-joins (blocking convergence point of two or more parallel activities), OR-splits (point in which one among multiple alternative branches is taken), OR-joins (non-blocking convergence point), iterations, pre- and post-conditions, events (happenings categorized by type). The flow of the process is described by means of arrows, that can represent either the control flow, the exchanged messages flow, or the data flow between the tasks. Activities are grouped into pools based on the participating organization that is in charge of the activity. Pool lanes are usually used to distinguish different user types within the organizations.

Figure 1 exemplifies a BPMN workflow diagram of online purchase, payment, and delivery of goods. The customer can choose the products to purchase, then submits his payment information. Then, two parallel tasks are executed by the seller employees: the warehouse manager registers the shipping of the order, and a secretary prepares the bill.

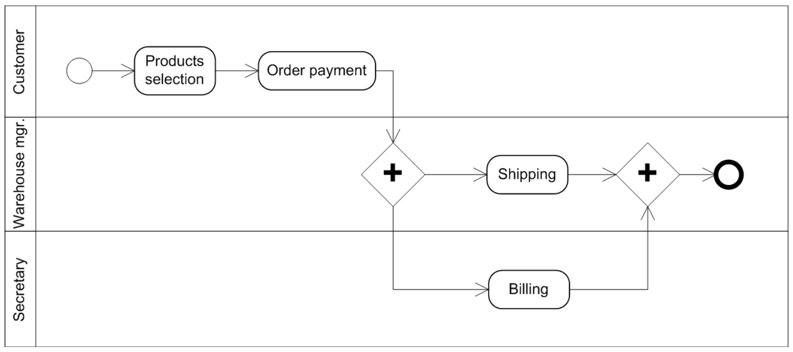

Fig. 1. Example of business process model expressed in BPMN

For the purpose of this paper, we define a business process bp by the tuple: $\langle nameBP, A, CF \rangle$, where:

- $nameBP$ is the name identifying the business process.
- A is the set of activities composing bp. An activity $a \in A$ is defined as $\langle nameA, roleA \rangle$, where $nameA$ is the activity identifier, and $roleA$ is a string determining its role within the business process.

– $CF \subseteq (A \times OP) \cup (OP \times A)$ is the controlflow. OP is the set of operators used for defining controlflow dependencies between the activities in A. Specifically: $OP = \{Sequence, ANDssplit, ANDjoin, ORsplit, ORjoin\}$.

We say that a business process *bp1* is a sub-process of a business process of *bp2* if the activities of *bp1* are activities of *bp2*, i.e., $bp1.A \subseteq bp2.A$, the control dependencies of *bp1* are also controlflow dependencies of *bp2*, i.e., $bp1.CF \subseteq bp2.CF$, and the controlflow of *bp1* forms a connected directed graph.

2.2 Abstract Business Process

An *abstract business process (ABP)* is a representative of a class of *equivalent* business processes, sharing the same set of activities and flow structure. In ABPs the activities are generic task descriptions, associated with semantic labels that provide information about the capabilities of the processing units able to perform the activities and descriptions of the data to be consumed and produced. An ABP can be implemented by several concrete business processes, which define the exact behaviour of the tasks and the names of the actors of the process and the association of the activities with the actors in charge of their execution. ABP descriptions are encoded in the form of annotations that map to concepts from ontologies that specify the semantics of these elements in the real world.

An ontology is commonly defined as an explicit specification of a conceptualisation [7]. Formally, an ontology θ can be defined as a set of concepts, $\theta = \{c1,\ldots,cn\}$. The concepts are related to each other using the sub-concept relationship, which links general concepts to more specific ones. For example, *CreditCardPayment* is a sub-concept of *OrderPayment*, for which we write $CreditCardPayment \sqsubseteq OrderPayment$. The concepts can also be connected by other kinds of binary relationships.

To semantically annotate the activities of a business process, we use the task ontology, θ_{task}. This ontology captures information about the action carried out by the activities within a domain of interest. In bioinformatics, for instance, an activity can be annotated using a term that describes the *in silico* analysis it performs. Example of bioinformatics analyses include *sequence alignment* and *protein identification*. Another example of a task ontology can be defined in the electronic commerce context. In this case, activities are annotated in terms of business transactions they implement. For instance, business transactions may include *quotation request*, *order confirmation*, and *credit card payment*.

To retrieve the task annotation of service operations we consider the function *task()* defined as $task\colon ACTIVITY \rightarrow \theta_{task}$, where $ACTIVITY$ denotes the domain of business process activities. We can now formally define an ABP.

Abstract business process. An abstract business process *abp* is defined as the pair: $\langle T, CF \rangle$, where

– T is the set of tasks that constitute *abp*: $T \subseteq \theta_{task}$.
– $CF \subseteq (T \times OP) \cup (OP \times T)$ is the control flow relating the tasks in T.

To map the tasks of two abstract business processes, we consider two classes of functions the domain of which are denoted by *MapEquiv* and *MapSpec*. The functions that belong to *MapEquiv* are used to map the tasks of a given abstract business process to the tasks of another abstract business process that perform *the same or equivalent tasks*. Let *abp1* and *abp2* be two abstract business processes and let f_{map}*: abp1.T* $\rightarrow$ *abp2.T* a function that maps the tasks of *abp1* to those of *abp2*. $f_{map} \in$ *MapEquiv* iff:

$$\forall\ t\ \in abp1.T,\ task(f_{map}(t))\ \equiv\ task(t)$$

The functions in *MapSpec* are used to map the tasks of a given abstract business process to the tasks of another abstract business process that perform equivalent or more specific tasks. Let *abp1* and *abp2* be two abstract business processes and let f_{map}*: abp1.T* $\rightarrow$ *abp2.T* a function that maps the tasks of *abp1* to those of *abp2*. $f_{map} \in$ *MapSpec* iff:

$$\forall\ t\ \in abp1.T,\ task(f_{map}(t))\ \sqsubseteq\ task(t)$$

To construct the abstract business process *abp* corresponding to a (concrete) business process *bp*, we use the function *abstractBP()* with the following signature: *abstractBP: BP* $\rightarrow$ *ABP*, where *BP* denotes the domain of business processes and *ABP* the domain of abstract business processes.

3 Ontology for Business Processes and their Relationships

To describe business processes, we define the *business process ontology,* θ_{BP}. The concepts of this ontology designate abstract business processes. Given a concept *c* from business process ontology θ_{BP}, we use the function *getABP(abp)*: $\theta_{BP} \rightarrow$ *ABP* to retrieve the abstract business process designated by *c*. The concepts in the ontology θ_{BP} are related using binary properties that encode relationships between abstract business processes. Specifically, we identify four binary properties to encode process relationships, namely, *equivalence*, *specialisation*, *overlap*, and *partOf*.

Process equivalence. Two abstract processes are equivalent iff their respective constituent tasks are equivalent tasks and are connected using the same controlflow. Formally, let *c1* and *c2* two concepts from the business process ontology θ_{BP} that designate the abstract business processes *abp1* and *abp2*, respectively. That is *abp1 = getABP(c1)* and *abp2 = getABP(c2)*. The two concepts *c1* and *c2* are equivalent, for which we write *c1* $\equiv$ *c2*, iff there exists a mapping function f_{equiv}*: abp1.T* $\rightarrow$ *abp2.T* in *MapEquiv* such that:

$$abp2.CF = \{(f_{equiv}(t), op),(t, op) \in abp1.CF\} \cup \{(op, f_{equiv}(t)),(op, t) \in abp1.CF\}$$

Process Specialisation. Let *c1* and *c2* be two concepts from the business process ontology θ_{BP} that designate the abstract business processes *abp1* and *abp2*, respectively. *c1* specialises *c2* iff the tasks of *abp1* are equivalent to or more specific than the tasks of *abp2*, and that they have the same controlflow. Formally, *c1* specialises *c2*, for which we write *c1* $\sqsubseteq$ *c2*, iff there exists a mapping function f_{spec}*: abp1.T* $\rightarrow$ *abp2.T* in *MapSpec* such that:

$$abp2.CF = \{(f_{spec}(t), op), (t, op) \in abp1.CF\} \cup \{(op, f_{spec}(t)), (op, t) \in abp1.CF\}$$

Part-of relationship. Let *c1* and *c2* be two concepts from the business process ontology θ_{BP} that designate the abstract business processes *abp1* and *abp2*, respectively. We say that *c1* is *part-of c2* iff there exists a concept *c3* that is equivalent to *c1* and that designates abstract business process *abp3 = getABP(c3)* that is sub-process of *abp2*.

Process overlap. Two concepts *c1* and *c2* overlap iff their respective ABPs have one or more tasks in common. Let *abp1* and *abp2* the abstract business processes designated by *c1* and *c2*, respectively. *c1* and *c2* overlap iff: $abp1.T \cap abp2 \neq \emptyset$

As mentioned earlier, the concepts in the business process ontology designate abstract business processes. Given a concept *c* from the business process ontology, the function *getAbstractBP(c)*: $\theta_{BP} \rightarrow ABP$ returns the abstract business process designated by *c*.

To manipulate the business process ontology, we assume the existence of the following operations:

defineConcept: $ABP \rightarrow \theta_{BP}$

defineProperty: $PROPERTY \times \theta_{BP} \times \theta_{BP} \rightarrow Boolean$

addInstance: $BP \times \theta_{BP} \rightarrow Boolean$

To define a new concept *c* that represents an abstract business process *abp* in the business process ontology, we use the operation *defineConcept(abp)*. The operation returns the concept defined. The operation *defineProperty(p,c1,c2)* defines a property $p \in$ *Property* between the concepts *c1* and *c2*. *Property* denotes the domain of binary properties, i.e., *Property*= {*equivalence, specialisation, part-of, overlap*}.

Business processes are defined as instances of the concepts in the business process ontology. Specifically, a business processes *bp* can be defined as an instance of a concept *c* iff *c* designate the abstract business process *abp* corresponding to *bp*: i.e., *abp = abstractBP(bp)*. To define *bp* as an instance of the concept *c*, we use the operation *addInstance(bp,c)*. The operation returns true if it is executed successfully and false, otherwise.

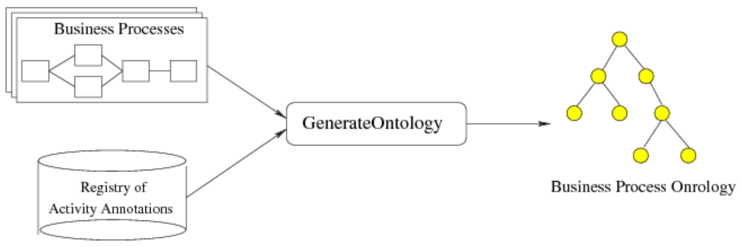

Fig. 2. Generation of the business process ontology

4 Creating and Populating the Ontology

The business process ontology is created and populated in an automatic fashion. Figure 2 illustrates the generation process: given a set of business processes together with semantic annotations describing the tasks of their constituent activities, the concepts of the business process ontology are defined. The binary properties that relate the concepts in the business process ontology, as seen in the previous section, are also automatically inferred. Furthermore, (concrete) business processes are defined as instances of the ontology concepts, thereby allowing the business process ontology to be used for business process discovery. The business process ontology is created according to the following algorithm:

```
Algorithm GenerateOntology
input BP
output $\theta_{BP}$
begin
1    for each bp $\in$ BP do
2        abp = abstract(bp)
3        if ($\exists$ c $\in$ $\theta_{BP}$, abp = getAbstractBP(c))
4        then
5            addInstance(bp,c)
6        else
7            c := defineConcept(abp)
8            addInstance(bp,c)
9            deriveAndAssertProperties(c)
end
```

For each business process *bp*, the corresponding abstract process *abp* is built *(line 2)*. If the business process ontology contains a concept *c* that designates the abstract business process *abp (line 3)*, then *bp* is defined as an instance of *c (line 5)*. If not, then a new concept is defined within the business process ontology to represent the abstract business process *abp (line 7)*, and *bp* is defined as an instance of the concept defined *(line 8)*. Furthermore, the binary properties that relate the newly defined concept *c* to other concepts in the business process ontology are derived and asserted using the *deriveAndAssertProperties(c)* subroutine *(line 9)*, operating as follows. The ABP designated by the concept *c* is compared to the ABPs designated by other concepts in the business process ontology. If the two abstract processes are found to be equivalent (see Section 3) then an *equivalence* property is defined for the respective concepts in the business process ontology. The *specialisation*, *part-of*, and *overlap* are defined in a similar fashion.

5 Discovering Business Processes

Most of existing proposals to business process discovery adopt the following paradigm. The user first formulates a query specifying the business process of interests by describing the activities that compose the business processes (e.g., specifying the actors in charge) and the controlflow that connects them. Then, a matching operation extracts the business processes that match the user query.

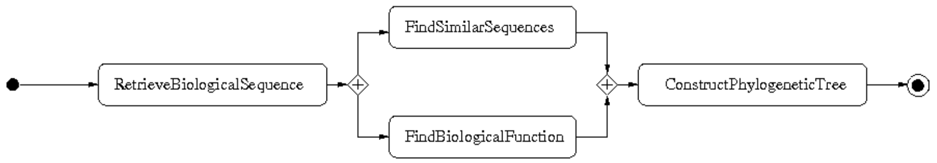

Fig. 3. Example of abstract business process specified by the user

We adopt a different approach that exploits information about business processes and their relationships encoded within the business process ontology. A discovery query takes the form of an abstract business process abp_{user} designed by the user by selecting concepts from the task ontology and connecting them using a controlflow graph. As a running example, consider that the user specifies the abstract business process illustrated in Figure 3. This is a simple process taken from the domain of bioinformatics and is composed of four tasks. First, the *RetrieveBiologicalSequence* fetches a biological sequence from accessible biosources. Then, the gene annotations associated with the sequence retrieved are fetched using the *FindBiologicalFunction* task, and its homologous sequences are fetched using the *FindSimilarSequences* task: these two tasks are concurrently performed. Finally, the phylogenetic tree of the sequence retrieved and its homologues is constructed using the *ConstructPhylogeneticTree* task. We distinguish the following cases for discovering the business processes that implement the abstract business process specified by the user.

- There exists in the business process ontology a concept *abp* that is equivalent to the abstract business process specified by the user, i.e., $abp_{user} = abp$. The result of the user query, in this case, is the set of business processes that are instances of *abp*: *instances(abp)*.
- Suppose now that there does not exist any concept in the business process ontology that is equivalent to abp_{user}. In this case, the concepts ABP in the business process ontology that are subconcepts of *abp* are retrieved: $ABP = \{abp \in \theta_{BP} \ s.t. \ abp \sqsubset abp_{user}\}$.
  As an example, the abstract business process illustrated in Figure 4 is subsumed by the abstract business process illustrated in Figure 3. Indeed, the task *RetrieveProteinSequence* is a subconcept of *RetrieveBiologicalSequence*, *FindSimilarBiologicalSequences* is a subconcept of *FindSimilarProteinSequences*, and the remaining two tasks, *FindBiologicalFunction* and *ConstructPhylogeneticTree* are subconcepts of themselves.
  The query result in this case is the set of business processes that are instances of at least one abstract process in ABP. That is:
  $$\bigcup_{abp_i \, \in \, ABP} instancesOf(abp_i)$$
- The business process ontology may not contain any concept that is equivalent or subconcept of the abstract business process specified by the user, abp_{user}.

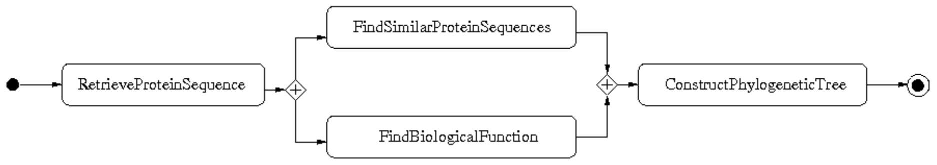

Fig. 4. Example of ABP that is subconcept of that illustrated in Figure 3

Instead of returning a null result to the user request, we attempt to create business processes that match the user request by aggregating other business processes.

The algorithm for building new aggregated business processes as further responses to the user queries is the following:

1. The set ABP of concepts in the business process ontology designating abstract business processes that are part of abp_{user} are retrieved. That is: $ABP = \{abp_i \in \theta_{BP}\ s.t.\ abp_i\ partOf\ abp_{user}\}$
2. Of the set ABP we extract a subset ABP' of abstract business processes, of which the union of tasks is a set that contains all the tasks required for building abp_{user}. Specifically:

$$\bigcup_{abp_i\ \in\ ABP'} abp_i.T\ =\ abp_{user}.T$$

For example, the abstract business processes illustrated in Figure 5 are parts of the abstract business process illustrated in in Figure 3. Moreover, the union of the tasks that compose the abstract business processes in Figure 5 covers all the tasks that compose the abstract business process illustrated in Figure 3.

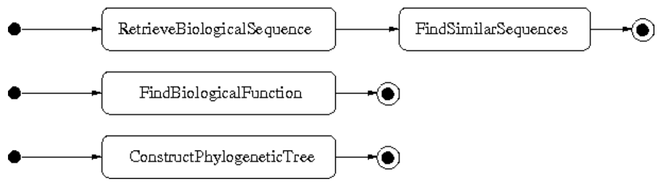

Fig. 5. Example of ABP that are parts of that illustrated in Figure 3

3. For each abstract process abp_i in ABP', we retrieve its business process instances, i.e., $instancesOf(abp_i)$
4. The result of the user query are business processes that are obtained by substituting the abstract business processes abp_i that are part of abp_{user}, with business processes that are instances of abp_i. As an example, consider that abp_{user} is composed of two abstract business processes abp and abp' that are connected using a sequence operator. And suppose that:

- *instancesOf(abp)* $= \{bp_1, bp_2\}$, i.e., there are two business processes bp_1 and bp_2 that are instances of *abp*.
- *instancesOf(abp′)* $= \{bp'_1, bp'_2\}$, i.e., there are two business processes bp'_1 and bp'_2 that are instances of abp'.

The business processes returned to the user are those obtained by substituting *abp* and abp' with thier instances. The business processes obtained using any possible combination of the instances of the business processes of *abp* and *abp'* are returned: in total the following combinations are used to build the business processes that are instances of abp_{user}: abp_1 and abp'_1; abp_1 and abp'_2; abp_2 and abp'_1; abp_2 and abp'_2.

In addition to the query paradigm just described , we developed an additional method in which business processes are *discovered by example*. In this case, instead of specifying an abstract business process, the user specifies an actual business process bp_{user} that is composed of activities (instead of tasks). The processing of this kind of queries is implemented in two phases:

- In the first phase, we construct an abstract business process abp_{user} that corresponds to the business process specified by the user bp_{user}.
- Then, we use the method for discovering business process presented above using as input abp_{user}.

This paradigm for querying business processes is suitable for users who are not familiar with the task ontology and, therefore, may not be able to specify an abstract business process that reflects their true needs. Also, it can be useful for designers who already have specified a business process and are interested in finding similar business processes developed by other designers.

6 Validation: Application to Bioinformatics Business Processes

The solution reported in this paper raises the following questions:

- Can we automatically create an ontology that describe the relationships between business processes using the algorithm presented?
- Does the business process ontology created provide useful information, in other words, does it help end users identify the business processes that meet their requirement. Typically, if the ontology created is composed of concepts that are independent from each other, by which we mean that they are not related using the properties we advocated in this paper, then this implies that the approach we describe is not applicable in practice. On the other hand, if the concepts in the ontology created are related by dependencies, then this mean that the ontology created can be beneficial for end users when exploring and querying business processes.

To answer the above questions, we conducted an experiment in which we used business processes from the domain of bioinformatics. Workflows are currently widely used in the bioiformatics field as a means for specifying and enacting em in silico experiments. As a result, a wide range of popular scientific workflow workbenches have been developed and are now used by the life science community. Among these workflow systems, we mention Taverna, Kepler and Triana. Figure illustrates a bioinfromatics workflow that is specified using the Taverna workbench.

For the purposes of our experiment, we used 23 workflows that were created in the context of eScience projects, namely, ISPIDER, myGrid and Planet. Some of the web services that embody the activities that constitue those workflows were annotated using the myGrid task ontology. Figure illustrate a fragement of the task ontology.

For the purposes of our experiement, we annotated the activities associated with non annotated web services. To this, we used the task ontology illustrates in Figure. Using the algorithm presented in section 4, we then automatically generated the business process ontology illustrated in Figure 6. Notice that the number of concepts in this ontology is 11 instead of 23. This is because, some of the concepts, i.e., abstract business processes, in the businesss ontology created are associated with multiple workflows.

Notice that the concepts in the created business process ontologfy are linked to each opther using specialisation, part-of and overlap relationships. This is a positive results, since it means that the ontology automatiucally created can be used by end users for exploring scientific workflows based on the dependencies between their associated abstract business processes.

To further assess our solution, we used the business process ontology created for evaluating the following queries:

Q1 : returns the business processes instances of the concept *abp1*. Figure illusstrates the abstract business process *abp1*.
Q2 : returns the business processes instances of *abp1* and its subconcepts.
Q3 : returns the business processes instances of the concept *abp1*, its subconcepts, and the concepts *abp1* is part-of.
Q4 : returns the business processes instances of the concept *abp1*, its subconcepts, and the concepts that are part-of *abp1*.
Q5 : returns the business processes instances of the concept *abp1* and the concepts that overlap with *abp1*.

The objectives intended by evaluating the above queries is two-fold. Firstly, to see whether the use of the business process ontology for guiding the query process allows increasing the recall of the results by using the notion of abstract business process and the specialisationproperty. Secondely, to see whether the ontology can be used for exploring the dependencies between business processes by using the specialisation, part-of and overlap relationships.

Table 2 shows the number of business processes returned for each of the above queries. It shows an increase in recall compared to the case where the queries

	Q1	Q2	Q3	Q4	Q5
Number of returned business processes	2	3	6	3	10

Table 1. Number of results obtained using the business process ontology

are evaluated relying only on structural information of business processes. For example, it shows that the number of business processes that are equivalent to or specialises the abstract business process abp_1 is *3*. Also, the ontology can be used for exploring the dependencies between business process tahnks to the specialisation, part-of and overlap properties. For instance, by considering the abstract business processes of which abp_1 is part-of, the number of business processes returned is 6.

To summarise, the validation we conducted using business process from the life sciences domain shows the affectiveness of the solution we proposed for describing and discoverying business processes. Specifically, the validation shows that:

- The ontology used for capturing semantic information about business processes and their relationships can be created in an automatic fashion.
- The concepts in the business process ontology created are highly dependent in the sense that they are related using the specialiosation, overlap and part-of properties advocated in this paper.
- The use of the ontology for evaluating business process discovery queries increases the recall with respect to the structural information only.
- Business process ontolopgy can be used as a mean for exploring the dependencies between business processes by relying on the properties between the abstract business processes that constitute the ontology.

7 Case study

To exemplify and give a flavour of the effectiveness of our method, we describe a simple case study aiming at: (1) showing that the business process ontology can be created automatically; and (2) showing that the recall of business process discovery queries increases when using the ontology.

Let's consider 10 business processes bp_i, $1 \leq i \leq 10$, covering all possible controlflow dependency types and with activities randomly annotated using a task ontology that we created for the sake of our evaluation. Using the algorithm presented in section 4, we automatically generated the business process ontology illustrated in Figure 6. Notice that the number of concepts in this ontology is 7 instead of 10, because some of the business processes were instances of the same ABP (e.g., both bp_1 and bp_4 were instances of the abstract business process denoted by the concept abp_1).

The business processes $\{bp_1, \ldots, bp_{10}\}$ are pair-wise different. That is:

$$\forall\, i, j \in \{1, \ldots, 10\},\; i \neq j \rightarrow bp_i \neq bp_j$$

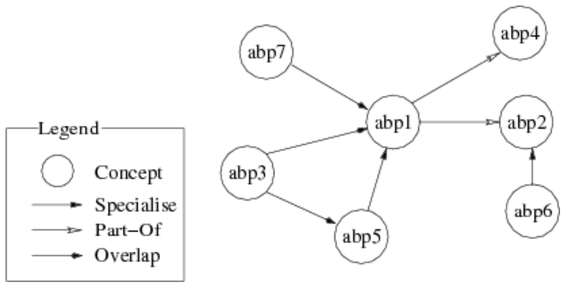

Fig. 6. The business process ontology automatically created using example BPs

Therefore, discovery queries that rely solely on the structural properties of business processes always return as a result *1* business process at most. To see whether the use of the ontology for answering business process discovery queries increase the recall, we posed the following queries over the ontology illustrated in Figure 6.

Q1 : returns the business processes instances of the concept *abp1*.
Q2 : returns the business processes instances of *abp1* and its subconcepts.
Q3 : returns the business processes instances of the concept *abp1*, its subconcepts, and the concepts *abp1* is part-of.
Q4 : returns the business processes instances of the concept *abp1*, its subconcepts, and the concepts that are part-of *abp1*.
Q5 : returns the business processes instances of the concept *abp1* and the concepts that overlap with *abp1*.

	Q1	**Q2**	**Q3**	**Q4**	**Q5**
Number of returned business processes	2	3	6	3	10

Table 2. Number of results obtained using the business process ontology

Table 2 illustrates the number of business processes returned for each of the above queries. It shows an increase in recall compared to the case where the queries are evaluated relying only on structural information of business processes. For example, it shows that the number of business processes that are equivalent to or specialises the abstract business process abp_1 is *3*. Also, relaxing the discovery query conditions implies an increase in the recall. For example, by considering the abstract business processes of which abp_1 is part-of, the number of business processes returned is 6. In summary, this shows that:

– The ontology used for capturing semantic information about business processes and their relationships can be created in an automatic fashion.
– The use of the ontology for evaluating business process discovery queries increases the recall with respect to the structural information only.
– Discovery queries can be relaxed to increase the recall by considering relationships such as *part-of* and *overlap*.

8 Related Work

Recently, many proposals have attempted to facilitate the discovery of business processes. Most of the approaches only apply graph-based comparison or XML-based querying on the business process specifications, disregarding ontology-based similarity discovery. In early works, Van der Aalst *et al.* [12] posed the basis of the concepts of inheritance between business processes, that we exploit in the relationships described in our ontology. Other works [13], defined the formal foundations and the semantics of business processes and similarity, on which we base our definitions.

Eyal *et al.* [3] proposed BP-QL, a visual query language for querying and discovering business processes modelled using BPEL. Lu and Sadiq [9] proposes a way for comparing and retrieving business process variants. Corrales *et al.* [5] developed a tool for comparing the controlflow of business processes in the scenario of service matchmaking, by reducing the problem of behavioral matching to a graph matching problem (i.e., receiving as input two BPEL models and evaluating the graph-based distance between them). These proposal offer a query mechanism on the process structure and topology only.

Goderis *et al.* [6] developed a framework for discovering workflows using similarity metrics that consider the activities composing the workflows and their relationships, implementing a ranking algorithm.

[10] proposed a framework for flexible queries on BP models, for providing better results when too few processes are extracted. [1] proposes the BPMN-Q query language for visual semantic queries over BPMN models. Kiefer *et al.* [8] proposed the use of semantic business processes to enable the integration and inter-operability of business processes across organizational boundaries. They offer an imprecise query engine based on iSPARQL to perform the process retrieval task and to find inter-organizational matching at the boundaries between partners. The work of Zhuge *et al.* [14] is instead closer to our approach, presenting an inexact matching approach based on SQL-like queries on ontology repositories. The focus is on flexible workflow process reuse, based on a multi-valued process specialization relationship. The matching degree between two workflow processes is determined by the matching degrees of their corresponding sub-processes or activities. Differently from us, the ontology cannot be automatically built from the workflow models and does not include explicit relationships between business processes, but only exploits ontological distances.

Beco *el al.* [2] specified the language OWL-WS (OWL for workflow and services) for describing ontologies of workflows and services aiming at providing grid architectures with dynamic behaviour on workflow specification and service invocation. The resulting workflow ontology did not focus on relationships between business processes and was not exploited for querying workflow similarity. Instead, the language was mainly used for specifying adaptive business processes.

9 Conclusions

In this paper we presented an approach for describing, storing, and discovering Business Processes. We extended the concept of similarity between process models by exploiting ontology definitions and the concept of abstract business process (ABP). Queries based on ABPs allow reuse and matching of business process models, thus saving time and reducing cost of implementation of enterprise workflows. Thanks to ontology-based comparison, we can evaluate the similarity between processes in a more flexible way with respect to traditional approaches, and therefore identify more potential similarities, for instance based on activity descriptions that are semantically close.

Ongoing and future works include the development of a large-scale repository of real business processes in the banking field, where some real applications are being developed for a major European bank.

References

1. Ahmed Awad, Artem Polyvyanyy, and Mathias Weske. Semantic querying of business process models. In *Enterprise Distributed Object Computing Conference (EDOC)*, pages 85 – 94, 2008.
2. Stefano Beco, Barbara Cantalupo, Ludovico Giammarino, Nikolaos Matskanis, and Mike Surridge. Owl-ws: A workflow ontology for dynamic grid service composition. In *e-Science*, pages 148–155, 2005.
3. Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying business processes. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *VLDB*, pages 343–354. ACM, 2006.
4. BPMI and OMG. Business Process Management Notation (BPMN) 1.2. http://www.bpmn.org/, 2009.
5. Juan Carlos Corrales, Daniela Grigori, and Mokrane Bouzeghoub. Bpel processes matchmaking for service discovery. In Robert Meersman and Zahir Tari, editors, *OTM Conferences*, volume 4275 of *LNCS*, pages 237–254. Springer, 2006.
6. Antoon Goderis, Peter Li, and Carole A. Goble. Workflow discovery: the problem, a case study from e-science and a graph-based solution. In *ICWS*, pages 312–319. IEEE Computer Society, 2006.
7. T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 1993.
8. Christoph Kiefer, Abraham Bernstein, Hong Joo Lee, Mark Klein, and Markus Stocker. Semantic process retrieval with isparql. In *ESWC*, pages 609–623, 2007.
9. Ruopeng Lu and Shazia Sadiq. Managing process variants as an information resource. In *Conf. on Business Process Management(BPM)*, pages 426–431, 2006.
10. Ivan Markovic, Alessandro Costa Pereira, and Nenad Stojanovic. A framework for querying in business process modelling. In *Multikonferenz Wirtschaftsinformatik*, February 2008.
11. OMG. Business Process Definition Metamodel. http://www.omg.org/cgi-bin/doc?bei/03-01-06, 2006.
12. Wil M. P. van der Aalst. Inheritance of Interorganizational Workflows to Enable B-to-B ECommerce. *Electronic Commerce Research*, 2(3):195–231, 2002.
13. Rob van Glabbeek. The linear time - branching time spectrum i; the semantics of concrete, sequential processes. In *Handbook of Process Algebra. J.A. Bergstra, A. Ponse, S.A. Smolka, eds.*, pages 3–99, 2001.
14. Hai Zhuge. A process matching approach for flexible workflow process reuse. *Information & Software Technology*, 44(8):445–450, 2002.