eChallenges e-2014 Conference Proceedings
Paul Cunningham and Miriam Cunningham (Eds)
IIMC International Information Management Corporation, 2014
ISBN: 978-1-905824-46-5

BPAL: A Tool for Managing Semantically Enriched Conceptual Process Models

Fabrizio SMITH, Maurizio PROIETTI
National Research Council, IASI "Antonio Ruberti",
Via dei Taurini 19, 00185, Rome, Italy
Email:{fabrizio.smith, maurizio.proietti}@iasi.cnr.it

Abstract: In this paper we will provide an overview of the Business Process Abstract Language (BPAL) Platform, which implements a Business Process (BP) modelling and reasoning environment where the procedural knowledge of a BP can be enriched through ontology-based annotations. The BPAL Platform provides a graphical user interface to ease the definition of a Business Process Knowledge Base that collects the various facets of process knowledge. It also provides a reasoner implementing services for the enactment, verification, retrieval, and composition of processes in the knowledge base. After discussing the functionalities and the architecture of the tool, we report on an experimental evaluation of the whole system, whose results are encouraging and show the viability of the approach.

1. Introduction

The penetration of Business Process (BP) Management solutions into production realities is constantly growing, due to its potential for an effective support to enterprise actors and business stakeholders along the entire BP life-cycle. During the *design* and *reengineering* of a BP, modelling languages such as BPMN[1], are employed to develop *conceptual process models* to be used by the stakeholders (designers, analysts, business men, engineers) to discuss requirements, validate design choices, and support decision making. In the *implementation* and *enactment* phases, different actors (developers, engineers) use models to generate, manually or semi-automatically, the runtime support for the process execution.

With such a central role of BPs in today's business realities, hundreds of BP models developed by different designers are available within an organization, constituting a relevant amount of knowledge regarding how the business is conducted. Nevertheless, the management of such knowledge is strongly hampered because standard BP modelling languages are an insufficient means for capturing the complex process-related knowledge of organizations and making it available in a machine-accessible form [1]. While their focus is on the procedural representation of a BP as a workflow graph that specifies the planned order of operations, the domain knowledge regarding the entities involved in such a process, i.e., the business environment in which processes are carried, is mainly left implicit. As a result, the automation of many tasks, such as process analysis, verification, retrieval and composition, is severely hampered and still require great manual efforts. In this scenario, the application of well-established techniques stemming from the area of Knowledge Representation has been shown as a promising approach for the enhancement of BP [1, 2] management systems, providing the means for the semantic lifting of BP models and enabling powerful automatic reasoning techniques.

Many of the tools available today on the market, open source or free of charge are able to provide, besides a graphical editor, additional services, such as some forms of verification, simulation of the designed processes, execution or (semi-)automatic generation

[1] www.bpmn.org

of executable code (e.g., in the form of BPEL code). Nevertheless, to the best of our knowledge, no commercial tool enables the semantic annotation of business process models, nor semantics-based reasoning services. Although several approaches have been proposed in literature to enable the exploitation of semantic facilities (see, e.g., the seminal work in [2, 3], and recent proposals [4,5]), very few implemented tools (e.g., [6,7]) give a (limited) support to the integrated management of the structural definition of a flow model, the formal definition of its behaviour, and the domain knowledge related to the business scenario where it operates.

In this paper we will overview the BPAL Platform, which implements a BP modelling and reasoning environment where the procedural knowledge of a BP can be enriched through ontology-based annotations. The theoretical basis of the tool is the Business Process Abstract Language (BPAL) [8], a language grounded in Logic Programming (LP) for representing and reasoning on various facets of process knowledge: (i) the meta-model of a BP model, which covers a core of the BPMN notation, (ii) the BP execution semantics, specified in a specialized version of the Fluent Calculus, a well-known LP-based action language, (iii) the behavioural properties of process executions, expressed by means of the CTL temporal logic, and (iv) the domain specific semantics of individual activities occurring in a BP, defined via OWL annotations (falling within the OWL 2 RL fragment) along the line of Semantic Web Services proposals.

The BPAL Platform provides a graphical user interface to ease the definition of a BP Knowledge Base (BPKB) that collects the various pieces of process knowledge. BPAL also provides a reasoner implementing services for the enactment, verification, retrieval, and composition of processes in the BPKB. Complex queries combining different aspects of process knowledge can be expressed in QuBPAL [9], a query language based on the SELECT-WHERE paradigm. QuBPAL queries are translated into clausal form and answered through an efficient, sound and complete LP query evaluation mechanism.

2. Tool Functionalities

The end-user tools provided by the BPAL Platform allow the semantic enrichment of existing BP models as well as the creation of new BPs from scratch. Furthermore, reasoning capabilities based on a Logic Programming (LP) engine are made available through the query mechanism. Figure 1 presents the identified use cases for the tool, and the main addressed functionalities are described in the following.

2.1 Managing BP Repositories

The platform provides functionalities for managing BP repositories, which include: (1) creation of a new BP model; (2) importing an existing BP, possibly designed through an external BP management system, encoded in a XML linear form (currently XPDL and *.bpmn* files are supported); (3) editing of a BP model through a graphical editor based on BPMN.

2.2 Semantic Annotation.

The support provided for the semantic annotation of BP elements in terms of a *reference ontology* includes: (1) *terminological annotations,* defining correspondences between elements of a BP and ontological concepts, in order to provide a formal and unambiguous definition of the former in terms of a suitable conceptualization of the underlying business domain provided by the latter (e.g., the participant *shipper* can be associated to the concept *Carrier,* which can be either an internal *Department* or a *Business Partner*); (2) *functional annotations,* providing additional information regarding how the world changes under the execution of the activities during a process enactment, in terms of *conditions* under which a task can be executed and *effects* on the state of the world upon its execution (e.g., the task

bill_client requires an *ApprovedPO* to be enacted and results in the issuing of an *Invoice*). To this end, OWL ontologies can be imported into the workspace and browsed through a graphical representation. Finally, the meta-data produced during the semantic annotation can be exported and imported as OWL/RDF documents too, in order to ease the sharing and re-use of semantic information.

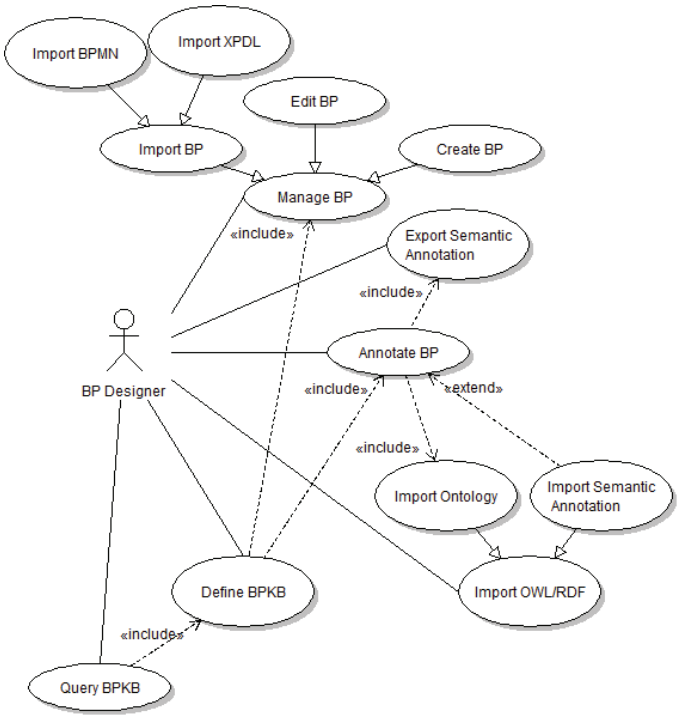

Figure 1. Use cases implemented by the BPAL Platform

2.3 BPKB Definition.

Once that the BP models are available and, possibly, they have been enriched with the domain knowledge provided by some reference ontology, a BPAL Business Process Knowledge Base (BPKB**)** can be built in order to enable the reasoning capabilities that the framework offers. To define a BPKB, the user selects the resources (i.e., processes, ontologies and annotations) to be included. A LP formalization of the BPKB is then produced, and used to feed the underlining reasoning engine.

2.4 Querying and Reasoning.

Every reasoning task supported by the framework is performed by posing LP queries on a BPKB. The user can interact with the reasoning engines through a query language (see next section) intended to ease the query formulation hiding the underlying formalism, and results can be consulted through the graphical user interface.

3. Reasoning Services

Our rule-based framework supports several reasoning services that can combine complex knowledge about BPs from different perspectives, such as the workflow structure, the behavioural semantics, and the ontology-based domain description. The provided services are all made available through a query mechanism supported by the query language QuBPAL, designed for interrogating a repository of semantically enriched BPs and based on the framework presented in [9]. It does not require the user to understand the technicalities of the underlying LP engine, since QuBPAL queries are SELECT-WHERE statements intended to be automatically translated into LP queries, and then evaluated by using the underlying reasoner.

The SELECT statement defines the output of the query evaluation, which can be a boolean answer, or values for variables occurring in the WHERE statement. The WHERE statement specifies an expression that restricts the set of data returned by the query, built from the set of the predicates defined in the BPKB and the connectives AND, OR, NOT, with the standard logic semantics. In the queries we use question mark to denote variables (e.g., *?x*), and we use the notation *?x*::*C* to indicate the terminological annotation of a variable *x* with respect to the concept *C*. It is worth noting that OWL/RDF resources are represented by means of the ternary predicate *t*(*s*, *p*, *o*) representing an RDF statement with subject *s*, predicate *p* and object *o*. For instance, the assertion *t*(*a*, *rdfs*:*subClassOf*, *b*) represents an inclusion axiom between *a* and *b*. The encoding of ontologies as set of triples allows us to pose queries over the ontology in a form very close to the SPARQL[2] standard, defined by the World Wide Web Consortium and widely accepted in the Semantic Web community. SPARQL is in fact designed to query RDF resources that essentially are organized as directed and labelled graphs, by matching graph pattern over RDF graphs. Graph patterns are in turn specified as triples where variables can occur in every position, along with their conjunctions and disjunctions. In this sense, while providing additional primitives to be used specifically for querying BPs, the ontology-related reasoning is specified in a QuBPAL query accordingly to consolidated Semantic Web standards.

To provide some insights about the language, we report in the following some examples of reasoning tasks expressed through QuBPAL queries, which are related to the verification of correctness criteria, compliance rules, retrieval and composition.

3.1 Verification

BPAL enables the verification of properties that depend on the interaction between the operational behaviour of the process and the ontology-based semantic annotation. Thus, besides well-known correctness criteria typically addressed in the workflow community (e.g., soundness), the tool is also able to verify that, during a BP enactment, no semantics-related constraint is violated (e.g., the fact that an order cannot be marked at the same time as *approved* and *rejected*). Some examples are reported in the following. Q1 expresses the *option to complete* property, i.e., from any reachable state of given BP *p*, it is possible to complete the process reaching the final state. Q2 verifies that no reachable state contains a contradiction, i.e., *false* is never implied by the assertions inferred in any state. Q3 verifies the existence of a state where an activity reached by the control flow is unable to execute due to some unsatisfied enabling condition. In the examples, *all_reachable*(*prop*, *bp*) and *reachable*(*prop*, *bp*) are predicates defined in terms of (CTL) temporal operators. The former means that from any state of *bp*, it is possible to reach a state where property *prop* holds, while the latter means that from the initial state of *bp* it is possible to reach a state where *prop* holds.

```
Q1. SELECT <>
WHERE all_reachable(final, p)

Q2. SELECT <>
WHERE NOT reachable(false, p)

Q3. SELECT <>
WHERE reachable(flow(?x, ?a) AND NOT enabled(?a), p)
```

[2] http://www.w3.org/TR/sparql11-overview/

3.2 Compliance

QuBPAL queries can also be used for verifying the compliance with business rules, i.e., directives expressing internal policies and regulations of an enterprise. In an eProcurement scenario, one such compliance rule may be that every order is eventually closed. In order to verify this property, we can define the query Q4 that holds if it is possible to reach the final state of the process *p* where, for some *o*, it can be inferred that *o* is an *order* which is not *closed*.

```
Q4. SELECT <>
WHERE reachable(final AND t(?o, rdf:type, PurchaseOrder) AND NOT
          t(?o, rdf:type, ClosedPO), p)
```

3.3 Retrieval

The LP inference mechanism can be used for computing boolean answers to ground queries, but also for computing, via unification, substitutions for variables occurring in non-ground queries. By exploiting this query answering mechanism we can easily provide, besides the verification services described above, also reasoning services for the retrieval of process fragments. For instance, if we want to retrieve all activities that must precede a *delivery* and require an *authorization* by the *sales manager*, then we may issue the query Q5.

```
Q5. SELECT ?a
WHERE precedes(?a, Delivering, p) AND requiresSalesMgrAuth(?a)
```

where (*i*) *precedes*(*a*, *b*, *bp*) is a predicate, defined by using the CTL temporal operators, which means that in any enactment of process *bp*, activity *a* precedes activity *b*, and (*ii*) *requiresSalesMgrAuth*(*a*) holds if the (terminological) annotation of *a* is a concept subsuming the OWL assertion (*requiresAuth* some *SalesMgr*).

3.4 Composition

The tool allows the user to design a new process by specifying a process skeleton, which constitutes a high level definition of a new BP to be composed, and retrieving sub-processes from a given BP repository [10] to implement the skeleton. Tasks appearing in the skeleton are associated with local constraints, which express requirements for the selection of the corresponding sub-processes to be retrieved, and global constraints, specifying the requirements on the composed BP as a whole. Local and global constraints are expressed as QuBPAL queries and evaluated over the BPKB in order to compute possible compositions.

4. Technology Description

The BPAL Platform[3] is implemented as an Eclipse Plug-in, whose main components are depicted in the functional view in Figure 2. It provides the BPKB Editor to assist the user through a graphical interface in the definition of a BPKB, and the BPAL Reasoner, based on an LP engine, able to operate on the BPKB through the query language QuBPAL.

[3] A short video demonstration is available at http://www.youtube.com/watch?v=xQkapzjhO7g

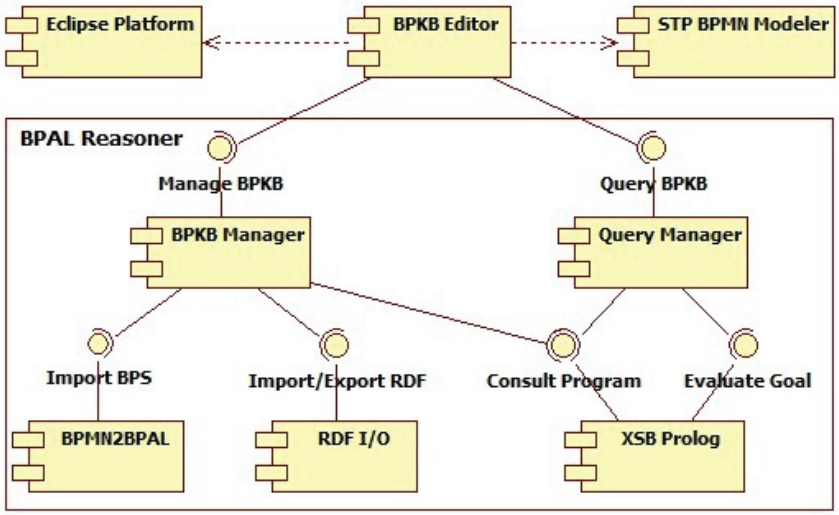

Figure 2. Functional view of the BPAL Platform

4.1 BPKB Editor

This component provides a graphical user interface to define a BPKB and to interact with the BPAL Reasoner. A screen-shot of the main components of the GUI is depicted in Figure 3.

- The left panel (Figure 3.a) is the Package Explorer, providing a tree view of the resources available in the workspace, including BP models and ontologies.
- The central panel (Figure 3.b) is the BP Modelling View, based on the Eclipse STP BPMN Modeller[4], comprising an editor and a set of tools to model BP diagrams using the BPMN notation.
- On the bottom left (Figure 3.c), the Ontology View allows for the visualization of OWL ontologies, published on the Internet or locally stored.
- The bottom panel (Figure 3.d) is the Annotation View, an editor for the annotation of process elements with respect to the reference ontology.
- The top-central panel (Figure 3.e) is the QuBPAL View, which provides a query prompt to access the BPAL reasoner through the query mechanism. Results can be consulted in the result panel (Figure 3.f).

4.2 BPKB Reasoner

This component implements the reasoning methods described in Section 3 by using the XSB Prolog system[5], which is a LP system based on the tabling resolution inference strategy. Tabling resolution has profound consequences in our setting since, as discussed in [8], it guarantees a sound and complete terminating evaluation of QuBPAL queries, with polynomial time (in the size of the state space) complexity.

[4] http://www.eclipse.org/soa/
[5] http://xsb.sourceforge.net/

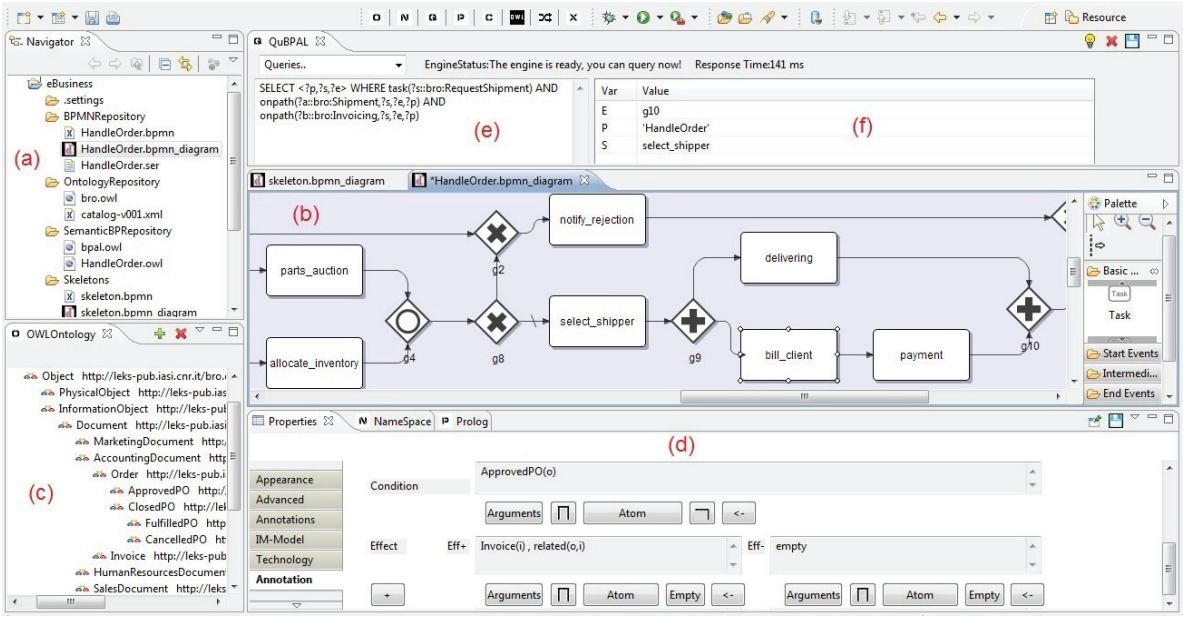

Figure 3 GUI of the BPAL Platform

Process models are imported into the BPKB from BPMN files via the BPMN2BPAL interface. In order to ease the sharing and re-use of semantic meta-data, semantic information used and produced during the annotation process (i.e., reference ontologies and semantic annotations) can be exported and imported from OWL/RDF files by means of the RDF I/O module. The underlying XSB-Prolog implementation of the rule-based reasoner can deal indifferently with RDF, RDFS and OWL 2 RL ontologies. The BPKB Manager handles the set-up of the reasoning engine by initializing and updating a BPKB. After populating the BPKB, inference is essentially performed by posing queries to XSB, connected through a Java/Prolog interface[6]. To this end, the Query Manager exposes functionalities to translate QuBPAL queries into LP queries, evaluate them, and collect the results in a textual form or export them in an XML serialization.

5. Experimental Evaluation

The approach has been applied to real-world scenarios coming from end-users involved in the European Project BIVEE[7] and from the pilot conducted within a bilateral collaboration between the Italian CNR and SOGEI (ICT Company of the Italian Ministry of Finance). The former is related to the modelling of production processes in manufacturing oriented networked enterprises, while the latter regards the procedural modelling of legislative decrees in the tax domain. The experiments we have conducted are encouraging and revealed the practical usability of the tool and its acceptance by business experts. On a more technical side, the LP reasoner based on the XSB system shown a significant effciency, since very sophisticated reasoning tasks have been performed on BPs of small-to-medium size (about one hundred of activities and several thousands of reachable states) in an acceptable amount of time and memory resources.

Some empirical results are reported in the following, related to a dataset described in Table 1. We started by adapting a real word process, dealing with eProcurement, obtaining the BP *P*, for which we report: the size, in terms of the number of flow elements; the number of reachable states; the number of exclusive, parallel, and inclusive gateways. As summarized in the table, the considered BP does not contain logical errors (e.g., deadlocks) and is characterized by a considerable number of gateways, that is, branching/merging

[6] http://www.declarativa.com/InterProlog/
[7] BIVEE: Business Innovation and Virtual Enterprise Environment (FoF-ICT-2011.7.3-285746)

points (about 40% of the total number of elements). We then annotated in three different ways the process, obtaining P_1, P_2, P_3. For each one, in Table 1 we report: the number of reachable states; the coverage of the annotation, in terms of the percentage of the annotated flow elements; the average size of each state, in terms of the number of ontological assertions (derived by the annotations) occurring in each state; the average size of the annotation, in terms of the number of assertions associated to the annotated flow elements; the errors exhibited by the BP. In particular, P_1 has been annotated without preventing logical errors induced by the annotation, P_2 presents a revised version of P_1 annotation, further extended in P_3.

For the annotation of the BP, we adapted an ontology covering documents and production related activities in the context of eProcurement and eBusiness, comprising about 100 concepts.

Table 1. Annotated processes used in the evaluation

	Size	*States*	*XOR*	*PAR*	*OR*	*Errors*
P	87	821	14	14	6	NO

	States	*Annotation Coverage*	*Average State Size*	*Average Annotation Size*	*Errors*
$\mathbf{P_1}$	944	35 %	7	3	2 non executable activities 150 inconsistent states 2 deadlocks
$\mathbf{P_2}$	2172	70 %	11	5	NO
$\mathbf{P_3}$	3866	100 %	16	8	NO

The experiments have been performed on an Intel laptop, with a 3 GHz Core 4 CPU, 8 GB RAM and Windows operating system. For each BPS we first tested the set-up of the reasoner, which include the translation of the BPKB into LP rules, their loading into the XSB reasoner, and the computation of the state space. Timing (measured in milliseconds) and memory occupation (measured in megabytes) are reported in Table 2. We then run the queries Q1-Q5 presented in Section 3. For each query, the average timing obtained in 10 runs is reported.

To better understand the performed tests, additional considerations are needed. Firstly, the above queries have been executed after the computation of the state space, which, due to the resolution strategy implemented by XSB, causes the population of the *tables* storing the intermediate results. The tables are then available in the subsequent queries, speeding up the computation. Secondly, to stress the engine, the evaluation of the performed queries requires the verification of ontology-based properties for each reachable state. Finally, the amount of required memory depends on the strategy adopted by the engine for the management of the tables. In the above experiments the default behaviour has been adopted and, according to that, every intermediate result is materialized. This explains the large memory consumption, which, if needed, can be strongly reduced by introducing specific configurations to limit the use of tables, trading space for time. It is also worth noting that no code optimization has been performed. Another remark regards the overhead introduced by the Java/Prolog bridge, which does not introduce a relevant performance degradation. Indeed, by running the same tests directly on XSB, without the Java infrastructure, the timings differ (up to a 10%) only in the presence of a large amount of results, mainly due to the inter-process data exchange.

Table 2. Run-time phase evaluation

	State Space		***Query Evaluation***				
	Time	*Memory*	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*
P	265	35	60	100	60	-	-
P_1	1030	210	110	2710	110	50	30
P_2	3300	670	530	4320	240	90	50
P_3	9720	1200	970	9250	405	105	60

6. Conclusions

In this paper we presented the BPAL Platform, a proof of concept implementation of a semantic BP modelling tool. The BPAL Platform provides a graphical user interface to assist the user in the definition and interrogation of a BPKB. We discussed how functionalities for modelling, semantically annotating and querying BPs are made available by the tool. Then, a practical evaluation of the reasoning engine has been conducted to show the viability of the approach.

The main design choices have been oriented at guaranteeing both a solid formal foundation and a high level of practical usability. The formal foundation is rooted in the logic-based approach of the BPAL framework. The practical usability derives from the efforts to support widely used and accepted standards and technologies. We adopted BPMN as graphical modelling notation, and its XML linear form to import and manipulate BP models, possibly designed through external BP management systems. For what concerns the ontology representation, we commit to OWL/RDF, the current de-facto standard for ontology modelling and meta-data exchange. The whole platform is finally packaged as an Eclipse plug-in, which extends the Eclipse STP BPMN Modeller, an established open-source BPMN editor.

For what concerns the reasoning engine, we built on top of a standard LP engine (XSB) an environment to manipulate and query semantically annotated BPs. The results have been quite encouraging, since without any particular query optimization strategy, the rule based implementation of the OWL reasoner and the effective tabled evaluation mechanism of the XSB engine shown good response time and scalability.

In the literature, several approaches have been proposed from different areas for dealing with the three main perspectives of process knowledge: graph-matching for *structural querying* [11,12], model checking for *behavioural properties verification* [13,14], description logics for *domain knowledge representation* [15,16,17]. Each of them proposes techniques that have been proven effective with respect to the specific aspect they address, but no one of them is able to model and reason about BPs by combining the knowledge related to the different perspectives considered above. As a result, many services with a practical relevance, e.g., the identification of a process fragment which enforces some behavioural properties where also domain knowledge is considered, would require, whenever possible, an ad-hoc integration of heterogeneous formalisms and tools. Our final goal is to manage a BPKB which organizes and stores the conceptual knowledge about the three aforementioned perspectives, and allows inference over this structure in a uniform and formal framework.

Finally, we do not aim at providing an alternative to existing solutions but, conversely, an “add-on” facility to be associated to the available BP modelling tools enhancing their functionalities. Indeed, most of the BP management systems adopted nowadays in business realities provide additional support for the definition of organizational, data and object models. While these conceptual models are designed for different purposes, they actually provide the very same kind of knowledge we expect formalized in business ontologies. Lifting existing models into formal theories suitable for automatic reasoning is the core of our proposal, and the prototype here discussed constitutes a step in that direction.

References

[1] Hepp, M., et al. (2005). Semantic Business Process Management: A Vision Towards Using Semantic Web Services for BPM. In Proc. of Int. Conf. on e-Business Engineering, IEEE.

[2] Fensel, D., et al. (2006). Enabling Semantic Web Services: The Web Service Modeling Ontology, Springer.

[3] Burstein, M., et al. (2004). OWL-S: Semantic Markup for Web Services. W3C Member Submission, http://www.w3.org/Submission/OWL-S/.

[4] Baryannis, G., Plexousakis, D. (2013). WSSL: A Fluent Calculus-Based Language for Web Service Specifications. In Proc. of the 25th CAiSE Conference, LNCS 7908, Springer.
[5] Calvanese, D., et al. (2012). Ontology-Based Governance of Data-Aware Processes. In Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems, LNCS 7497, Springer.
[6] Dimitrov, M., et al. (2007). WSMO Studio: A Semantic Web Services Modelling Environment for WSMO. In Proc. of the 4th European Conf. on the Semantic Web. LNCS 4519, Springer.
[7] Born, M., et al. (2009). Supporting Execution-level Business Process Modeling with Semantic Technologies. In Proc. of the 14th DASFAA Conference. LNCS 5463, Springer.
[8] Smith, F., Proietti, M. (2013). Rule-based Behavioral Reasoning on Semantic Business Processes. In Proc. of the 5th Int. Conf. on Agents and Artificial Intelligence, SciTePress.
[9] Smith, F., Missikoff, M., Proietti, M. (2012). Ontology-Based Querying of Composite Services. In Business System Management and Engineering, BSME 2010, LNCS 7350, Springer.
[10] Smith, F., Bianchini, D. (2012). Semi-Automatic Process Composition via Semantics Enabled Sub-Process Selection and Ranking. Enterprise Interoperability V, I-ESA'12, Springer.
[11] Sakr, S., Awad, A. (2010). A Framework for Querying Graph-based Business Process Models. In Proc. of the 19th Int. Conf. on World Wide Web, ACM.
[12] Beeri, C., et al. (2008). Querying Business Processes with BP-QL. *Information Systems*, 33:477–507.
[13] Fu, X. Bultan, T., Su, J. (2004). Analysis of Interacting BPEL Web Services. In Proc. of the 13th Int. Conf. on World Wide Web, ACM.
[14] Liu, Y., Muller, S., Xu, K. (2007). A Static Compliance-checking Framework for Business Process Models. *IBM System Journal*, 46:335–361.
[15] Di Francescomarino, C., Tonella, P. (2009). Crosscutting Concern Documentation by Visual Query of Business Processes. In Business Process Management Workshops, LNBIP 17, Springer.
[16] Lin, Y. (2008). Semantic Annotation for Process Models: Facilitating Process Knowledge Management via Semantic Interoperability. PhD Thesis, Norwegian University of Science and Technology.
[17] Markovic, I. (2011). Semantic Business Process Modeling, KIT Scientific Publishing.

