BPAL: A Platform for Managing Business Process Knowledge Bases via Logic Programming*

Fabrizio Smith, Dario De Sanctis, Maurizio Proietti

National Research Council, IASI "Antonio Ruberti" - Viale Manzoni 30, 00185 Roma, Italy
{fabrizio.smith, dario.desanctis, maurizio.proietti}@iasi.cnr.it

1 Introduction

The adoption of structured approaches for the management of the Business Processes (BPs) operating within an organization is constantly gaining popularity. Nevertheless, their further automation is severely hampered by the fact that standard approaches are an insufficient means for capturing the complex process-related knowledge and making it available in a machine-accessible form [1]. As a result, many tasks, such as process analysis, verification, retrieval and composition, 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 and Web Service [1, 2] management systems.

While several tools are today available for the modeling, verification, simulation, and execution (e.g., Intalio, Tibco, YAWL, Enhydra Shark), no commercial tool enables the semantic annotation of BP 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 behavior, and the domain knowledge related to the business scenario where it operates.

The BPAL platform implements a BP modeling 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 [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 schema (BPS), which covers a core of the BPMN notation, (ii) the BPS execution semantics, specified in a specialized version of the Fluent Calculus, a well-known LP-based action language, (iii) the behavioral 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,

* A video demonstration is available at `http://www.youtube.com/watch?v=xQkapzjhO7g`

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 An Overview of the Functionalities of BPAL

Management of BP Repositories. The platform provides functionalities for managing BP repositories, such as: (1) creating a new BPS, (2) importing an existing BPS from an XML serialization of a BPMN diagram, and (3) editing a BPS via a graphical editor.
Semantic Annotation. Two kinds of annotations enable the enrichment of a BPS with domain related knowledge defined in a given reference ontology: (1) *terminological annotations*, which associate BPS elements with concept expressions, and (2) *functional annotations*, which define the conditions under which flow elements can be executed and the effects of their execution on the state of the world.
Enactment. The execution of a BP is modeled as an *execution trace*, corresponding to a plan in the Fluent Calculus, i.e., a sequence of actions of the form $[\texttt{begin}(\texttt{e}_1), \ldots, \texttt{complete}(\texttt{e}_\texttt{n})]$ where $\texttt{e}_\texttt{i}$ represents a flow elements. Execution traces correspond to process logs, which are commonly stored by BPM systems to record the enactment of BP instances. BPAL can verify whether a trace can be generated by a BP enactment (i.e., the compliance of a trace w.r.t. a given BPS) and, by exploiting the LP inference mechanism, the rules defining the trace semantics can also be used to *generate* the traces of a BPS satisfying some given (behavioral and/or ontological) property.
Verification. BPAL enables the verification of properties that depend on the interaction between the operational behavior 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. For instance, given a BPS named p, we can define the following predicate:

$$\texttt{holds}(\texttt{not}(\texttt{ef}(\texttt{false})), \texttt{bps}(\texttt{p}))$$

meaning that no state is reachable (expressed by the temporal operator ef 'exists finally') where the false concept can be inferred from functional annotations specifying the effects of execution (e.g., o : approvedPO and o : rejectedPO) and execution-independent OWL axioms (e.g., approvedPO $\sqcap$ rejectedPO $\sqsubseteq$ false).
Compliance. Temporal queries can also be used for analyzing 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*. This rule can be expressed by the following predicate meaning that it is not possible to reach the final state of the process where some order is not closed:

$$\texttt{holds}(\texttt{not}(\texttt{ef}(\texttt{final}(\texttt{p}) \texttt{ and } \texttt{nonclosedPO})), \texttt{bps}(\texttt{p}))$$

Here nonclosedPO holds in a given state if, for some O, the OWL assertion O : order holds and the OWL assertion O : closedPO does not hold.
Retrieval. The LP inference mechanism based on resolution can be also used for computing, via unification, substitutions for variables occurring in queries. BPAL exploits this query answering mechanism and provides a reasoning service for the retrieval of process fragments described in a declarative way. In particular, the WHERE clause of

a QuBPAL query can specify a combination of ontological, structural, and behavioral properties. 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 following query (names prefixed by '?' denote variables):

`SELECT ?a`

`WHERE precedes(?a,delivering,p) AND requiresSalesMgrAuth(?a)`

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

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

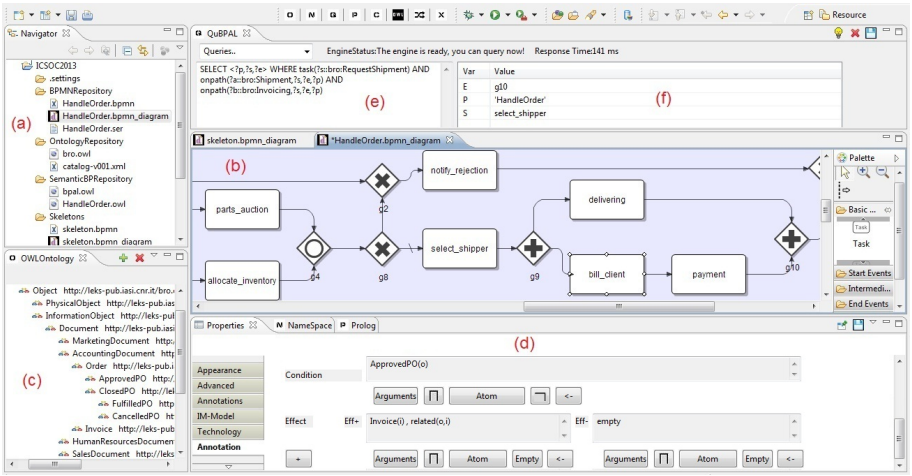

Fig. 1: GUI of the BPAL platform

3 Tool Description

The BPAL platform is implemented as an Eclipse Plug-in[1], whose main components are depicted in the functional view in Figure 2.

The **BPKB Editor** provides a graphical user interface to define a BPKB and to interact with the BPAL Reasoner. It encompasses: a tree view of the available resources (Fig. 1a), the STP[2] BPMN Modeler (Fig. 1b), a browser for the visualization of OWL

[1] `http://www.eclipse.org/`

[2] `http://www.eclipse.org/soa`

ontologies (Fig. 1c), an annotation panel (Fig. 1d), and finally a query prompt (Fig. 1e) to submit queries and collect the results (Fig. 1f).

The **BPAL Reasoner** provides the means to process and query the BPKB. Process schemas are imported into the BPKB from BPMN process models 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 rule-based reasoner can deal indifferently with RDF, RDFS and OWL (restricted to the RL profile). The *BPKB Manager* handles the set-up and the interaction with the LP engine by initializing and updating a BPKB. After populating the BPKB, inference is essentially performed by posing queries to the *XSB Prolog* engine[3], connected through a Java/Prolog interface. XSB extends conventional Prolog systems with an operational semantics based on tabling, i.e., a mechanism for storing intermediate results and avoiding to prove sub-goals more than once. In our setting, XSB has a crucial advantage with respect to other Prolog systems, because tabling ensures the termination of query evaluation over a BPKB. Finally, 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.

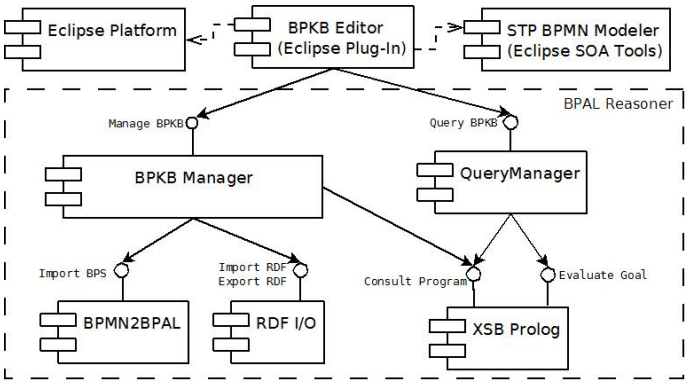

Fig. 2: Functional view of the BPAL platform

4 Discussion

The BPAL platform presented in this paper enables the combination of the procedural and ontological perspectives related to process knowledge in a very smooth and natural way. BPAL provides a uniform framework for modeling and semantically enriching BP models, in order to reason on properties that depend on the sequence of operations that occur during process enactment and also on the domain where the process operates. In doing this, our approach does not introduce a new BP modeling paradigm, but provides a framework where one can map and integrate knowledge represented by means of existing formalisms. This is very important from a pragmatic point of view, as one can express process-related knowledge by using standard modeling languages such as

[3] `http://xsb.sourceforge.net/`

BPMN for BP models and OWL for ontologies, and then automatically translate this knowledge into logic programs (see [8] for details), thus allowing the use of standard LP methods and tools to perform reasoning. This LP translation also enables the application of further, very sophisticated reasoning techniques recently developed in the field of logic programming. In this respect, interesting directions of future work include the enhancement of our framework with: (i) *process mining* facilities, by adopting Inductive Logic Programming techniques, such as the ones presented in [11], and (ii) verification techniques for BPs in the presence of data constraints, by following approaches based on Constraint Logic Programming such as, for instance, the one proposed in [12].

The approach has been applied to real-world scenarios coming from end-users involved in the European Project BIVEE[4] and from the pilot conducted within a collaboration between the italian CNR and SOGEI (ICT Company of the Italian Ministry of Finance). The former is related to the modeling of production processes in manufacturing oriented networked-enterprises, while the latter regards the procedural modeling 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 efficiency, 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.

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. Lamma, E., et al. (2008). Applying Inductive Logic Programming to Process Mining. In Proc. of the 17th Int. Conf. on Inductive Logic Programming, LNCS 4894, Springer.
12. F. Fioravanti, A. Pettorossi, M. Proietti, and V. Senni. Generalization strategies for the verification of infinite state systems. *Theo. Pract. Log. Prog.*, 13(2):175–199, 2013.

[4] BIVEE: Business Innovation and Virtual Enterprise Environment (FoF-ICT-2011.7.3-285746)