Comput Sci Res Dev (2017) 32:257–267
DOI 10.1007/s00450-016-0333-4

SPECIAL ISSUE PAPER

A semantic model for business process patterns to support cloud deployment

Beniamino Di Martino[1] · **Antonio Esposito**[1] · **Stefania Nacchia**[1] · **Salvatore Augusto Maisto**[1]

Published online: 9 November 2016

Abstract Interest in cloud computing has steadily increased in the last few years, with new services continuously being introduced into the IT market by cloud vendors. Research efforts have been made to understand how business requirements can be mapped to cloud services, in order to support and ease the development of new cloud based applications, starting from business definitions expressed in standard formats, such as the business process model and notation. In most cases, business models greatly differ in structure and in the applied semantics, even when describing the very same objectives and scopes: this reflects the different points of view of the process designers. Such a situation can only make the mapping to cloud resources more complicated, since there is no clear correspondence between business tasks and cloud services. The use of Patterns for the description of both business processes and cloud applications can represent an efficient solution to such an issue, as they can help to systematize the huge variety of possible business and cloud definitions and to assess a set of pre-defined mappings which can be used as a basis for cloud deployment. In this paper we focus on the definition of a semantic based model for the representation of business process patterns, with an example of the mapping to cloud resources of such patterns.

✉ Beniamino Di Martino
beniamino.dimartino@unina.it

Antonio Esposito
antonio.esposito@unina2.it

Stefania Nacchia
stefania.nacchia@studenti.unina2.it

Salvatore Augusto Maisto
salvatoreaugusto.maisto@studenti.unina2.it

[1] Department of Industrial and Information Engineering, Second University of Naples, Aversa, Italy

Keywords Business process · Patterns · Cloud computing

1 Introduction

In the past years on-line services and applications have dramatically developed, and as of today companies wishing to reach the market are almost obliged to offer a web-enable access to their services to customers. In such a situation, cloud computing offers the best means for companies to develop and deploy their services, keeping their costs low at the same time. However, due to the huge variety of services and offers currently available, and a general lack of a common standard formalism for their description, it can be difficult for potential customers to choose the ones best suiting their need. Choosing the cloud services which best meet their requirement is even more difficult for business experts, who often need to rely on software developers to implement the applications which respond to their business requirements. While standards for the description of business processes have been defined [business process model and notation (BPMN) being currently the most adopted one], there is still the need to actually map the tasks and actors of business processes to actual cloud services and resources. This can be particularly difficult in some cases, because of the gap existing between the business and IT areas in terms of knowledge and expertise. Also it often happens that similar processes, sharing many goals and expressing almost the same requirements, are very different from a structural point of view or in the used semantics: this is caused by both a lack of communication within the company and by the ways different designers interpret the same requirements. In order to address both such situations, it would be useful to rely on a set of predefined business processes, corresponding to well known business requirements, the mapping of which to potential cloud solutions is

already known and has been tested in the past. In this paper, business process patterns are defined by using a semantic based model, to represent common business requirements. By means of patterns, similar processes are implemented by using the same, well-tested solutions. Also, once the mapping to cloud solutions has been assessed, such a knowledge can be represented in the very same semantic model proposed to describe business process patterns, and it will represent the basis for future deployments on the cloud. In addition, it is possible to exploit cloud patterns to further systematize and organize the different possible implementations available for the same set of business processes.

The remainder of this paper is organized as follows: Sect. 2 reports the current state of the art on the topic of business process patterns definitions and semantic models for their annotation; Sect. 3 describes the semantic model which has been defined for the representation of business process patterns, while Sect. 4 reports an example of a set of patterns which have been defined; Sect. 5 describes how, starting from a business process pattern, it is possible to suggest a developer the potential cloud services and resources to exploit, with the support of external models and ontologies; Sect. 6 briefly describes a web-based prototype tool which supports the annotation of existing business processes with external ontologies; Sect. 7 closes the paper with some final considerations on the current status of the work and future developments.

2 State of the art

The idea of applying patterns to business processes design and implementation is not new, as different research efforts have been carried out in the past to discover and define such patterns. The work presented in [1] presents an interesting set of patterns, not specifically related to business processes, but which can be easily represented in BPMN [2]. Such patterns describe both simple and complex combinations of basic control flow structures, identifying commonly adopted and re-usable solutions for workflow definitions. However, while such patterns can represent the building blocks for more complex business processes designs, they are too low level to be used stand-alone. Also, they miss semantic information which could be used to support the matching with semantically described cloud services. Higher level patterns, provided by the same authors in [3], provide representations of resource and data centred workflows which can be used in the definition of business processes. However, semantic-based representation are still missing, and in some cases a proper representation in BPMN could be not possible, due to the data and resource centric vision of such patterns, which is not fully compatible with the task-centric view of business processes. Activity patterns, representable both with UML activity diagrams and BPMN, are presented in [5]. Such patterns describe high level task workflows, which deal with basic activities in a process scenario, such as generic information requests, decision making, notifications and so on. One of the main drawbacks of such patterns is represented by their structural similarity: in most cases, the workflow is exactly the same, with the only difference being represented by task names. This can represent a problem since, without proper semantic annotations, it is impossible for an automatic framework to distinguish them. The practical benefits deriving from the application of semantics to business process modelling are presented in [6], in which the authors stress the important role played by semantic annotations in easing the design and development of new business processes.

The work presented in [7] provides the definition of an ontology for the semantic description of a BPMN's structure: all the graphical elements of the standard are represented by OWL [8] classes, with object properties marking their exact relationship. Such an ontology is then used to validate the process model and ensure that all of the constraints imposed by the notation are respected. The approach proposed in our previous work [9] exploits such a structural representation, but it also extends it in order to support the recognition of specific process patterns and to provide a context-aware analysis of the process. However, such a semantic representation is too tied to the BPMN standard which, whilst being among the most used ones for business process representation, could be one day substituted with more advanced language. That is why in this work we propose an OWL-S [10] based representation of business processes, which at the same time can provide effective support for the mapping of tasks to cloud services and guarantee independence from the business process modelling language adopted. A remarkable research effort involving both business process patterns and semantics is represented by the European funded project *SUPER— Semantics Utilized for Process management within and between EnteRprises* [11]. Within such a project, a complex and exhaustive framework has been developed, composed by a set of tools for the annotation of BPMN documents and a multi-layered ontology structure for the description of both structural aspects of the BPMN and domain specific concepts. However, the framework is completely based on the *Web Service Modeling Ontology* (WSMO) [12] to provide the semantic support which, according to independent researchers, is less mature and supported than OWL-S [13].

3 Semantic model for business process representation

The semantic representation of business processes proposed in this paper is strongly based on the OWL-S standard. In par-

ticular, it is possible to represent the business process workflow with OWL-S constructs, without losing any knowledge about the original process: instead, semantic annotations are enabled by this kind of approach, thus allowing for an enrichment of the process description. The mapping between the business process description and the OWL-S structure is executed as follows:

- Process' tasks are mapped to OWL-S *processes*; in particular, simple tasks are translated to *atomic processes*, while complex tasks (represented via Sub-Processes in BPMN) are mapped to *composite processes* which are, in turn, composed by atomic processes;
- A series of two or more consecutive tasks are automatically mapped to a *sequence* control structure, in which the first task is mapped to the first process of the sequence. OWL-S uses a recursive definition of sequence, which is composed by a (head) process and a (tail) sub-sequence of consecutive processes, organized in the same way. The sequence is completed when a "null "process is encountered.
- The parallel execution of two or more tasks is represented via the *split* and *split and join* constructs. The tasks to be executed in parallel are still represented via processes. The main difference between these two constructs relies in the lack of synchronization of the former one: that is, if a simple split is used, the workflow can continue after the parallel Section even if some of its tasks/processes have not been completed. Using a combination of split and split and join clauses can help in defining complex behaviours, which in a business process language such as BPMN are handled via particular gates.
- OWL-S is able to express *pre-conditions* on processes, via an object property *hasCondition*. Such conditions are written using a logical language, such as SWRL [14] or KIF [15]. Using preconditions on control structures or processes, it is possible to retain information on the conditional execution of tasks. As an instance, in BPMN, such information is encoded in gates.
- *Start events* of different categories can be represented as *pre-conditions* on the first process of the workflow. In this way, if the business process can be started as a result of different events, enabling different tasks/processes, it would still be possible to reproduce the scenario by adding pre-conditions to each of the involved processes. The same applies to all intermediate events occurring in the workflow: as an instance, in BPMN, we could have a task A waiting for a notification message from another task B in a different lane. This situation could be modelled as a pre-condition on task A.
- Since each process in an OWL-S description reports and *effect*, it is possible to model an *end event* as an effect of the last task/process of the workflow. Intermediate events (such as the sending of a notification in BPMN) can be treated in the same way.
- Actors and task executers defined in the business process (BPMN *Lanes* contain information on task executers) can be easily mapped into an OWL-S based representation by simply exploiting the *hasClient* property. In OWL-S, such a property defines the agent from whose point of view a process is executed. In our case, it is the best candidate to express the actual executer of a task.

One of the main advantages of using an OWL-S based representation for a business process is to be found in the possibility to extend the original ontology, by adding new properties and classes, without losing compatibility with the existing ones. In our case we want to annotate business process elements with external ontologies in order to infer, by using queries and logical rules, equivalences between tasks and suitable cloud services. For our purposes, an additional property *equivalentTo* connects elements of the OWL-S representation to an external ontology, which describes both cloud services and patterns. Such an ontology is described in [16] and offers a comprehensive collection of systematized cloud service, with their parameters and functions, together with the description of semantic cloud patterns. This external ontology has two functions: first of all, it hides details regarding cloud services descriptions, since it is organized in hierarchical navigable categories, which are the only elements exposed to the user; second, cloud patterns organized according to the very same hierarchical category of services, in order to make the querying simpler. Patterns organize groups of related services, by proposing a ready-to-use and customizable solution to implement those task which cannot be substituted by a single cloud service exposed by a vendor.

3.1 Model correctness

In order to verify if the semantic models, derived from the BPMN diagrams, correctly represent the original process, we analyse the logs derived from a simulated execution of both the OWL-S and the BPMN representations. In particular, the simulation of the processes described in BPMN has been carried out through the IBM Process Manager tool [17], which offers the possibility to define different sets of inputs for the processes, and to execute them while registering an event log of the simulation. In order to obtain the same results for the OWL-S simulation we have implemented a Java program, which uses OWL-API [18], to read the semantic-based description and simulated its execution by using the same sets of inputs exploited with the BPMN tool. Since we want to obtain a semantic-based representation which perfectly adheres to the original process definition, we consider it correct only if the obtained logs match, under the same inputs. However, tasks executed in different lanes are meant to be

run in parallel and no order is enforced between the corresponding log entries. The same applies to tasks appearing immediately after gates.

4 Representation of a business process pattern

By using the semantic-based representation provided in Sect. 3 it is possible to describe business processes and annotate them in order to retrieve cloud services to implement them. However, even if the cloud service ontologies is very easy to understand and employ, it still requires some effort from users, especially because of the high number of service categories available. The approach we propose here consists in using the very same semantic representation to describe business process patterns and to let users select and compose them, in order to later move to the cloud. Since business process patterns are defined before creating the actual business process by experts, as they provide pre-configured combinations of tasks to reach specific goals, users will not have to navigate any cloud service ontology: the connections will already exist.

In this paper we will take in consideration the BPMN representation of a business process pattern as the basis to produce the semantic representation. The steps followed to produce such a representation are here resumed:

1. Create a process for each of the tasks described in the model. If the task is executed after a gate with a condition on it, add it as a pre-condition to the process.
2. Consecutive tasks, or tasks connected with a gate having just one outgoing flow (used just to apply a condition to that task execution) are included in a sequence structure.
3. Tasks connected to the outgoing flows of a gate are included in a split structure. If a task is the head of a sequence, then the entire sequence becomes part of the split structure. If the tasks/sequences included in the split structure terminate with a gate requiring a synchronization, turn the split structure in a split-join one.
4. Repeat steps 2 and 3 until all tasks have been included and all gates have been evaluated.
5. Finally consider events. Start and end events without conditions or effects can just be ignored: the first task to be executed has already been modelled and added to a sequence/split construct for execution. Start and end events with conditions and effects are eliminated, as conditions and events are associated to the first/last task of the process. Intermediate events' conditions and effects are automatically associated to the previous/next tasks.

Consider the business process pattern shown in Fig. 1, which has been extracted from [5]. The pattern reported here describes a situation in which a *requester* asks for the permission to execute a generic action and a *reviewer* gives or refuses such permission. There are two variants of the same pattern: in the first one (left side of Fig. 1) requests and responses are continuously sent and received; in the second one (right side), once the response has been received and the action has been performed, the process stops. Lets consider the first version of such a pattern.

- The start and end events of the Requester lane do not have any pre-conditions or effects, so they wont appear in the OWL-S representation.
- Tasks *send approval request* and *receive approval result* are modelled as simple processes. However, since they

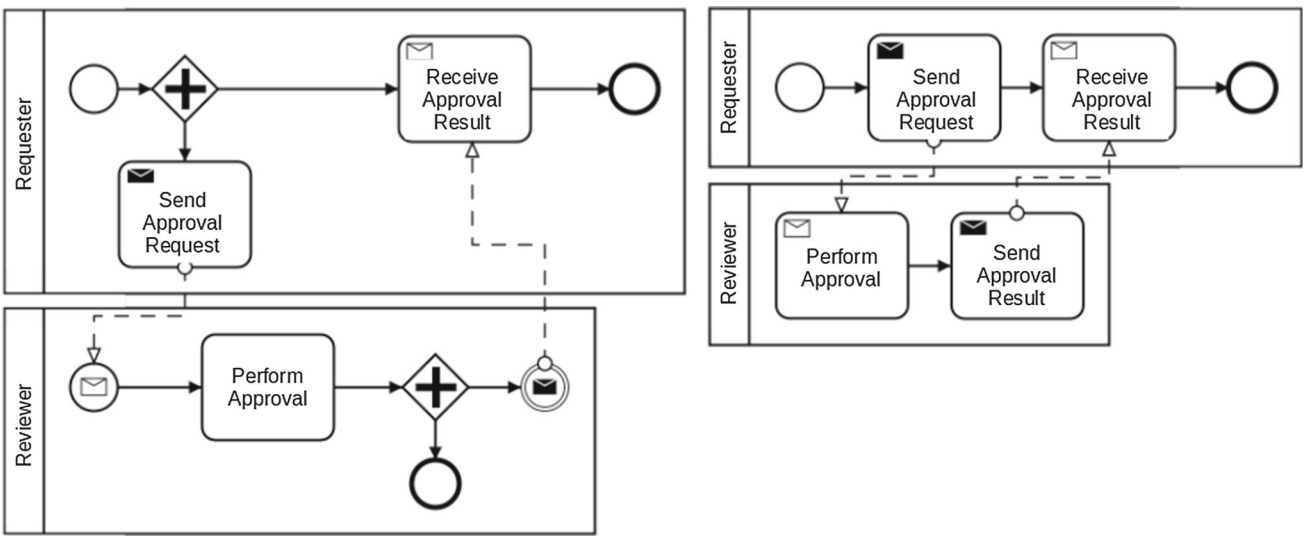


Fig. 1 Business process pattern for request approval

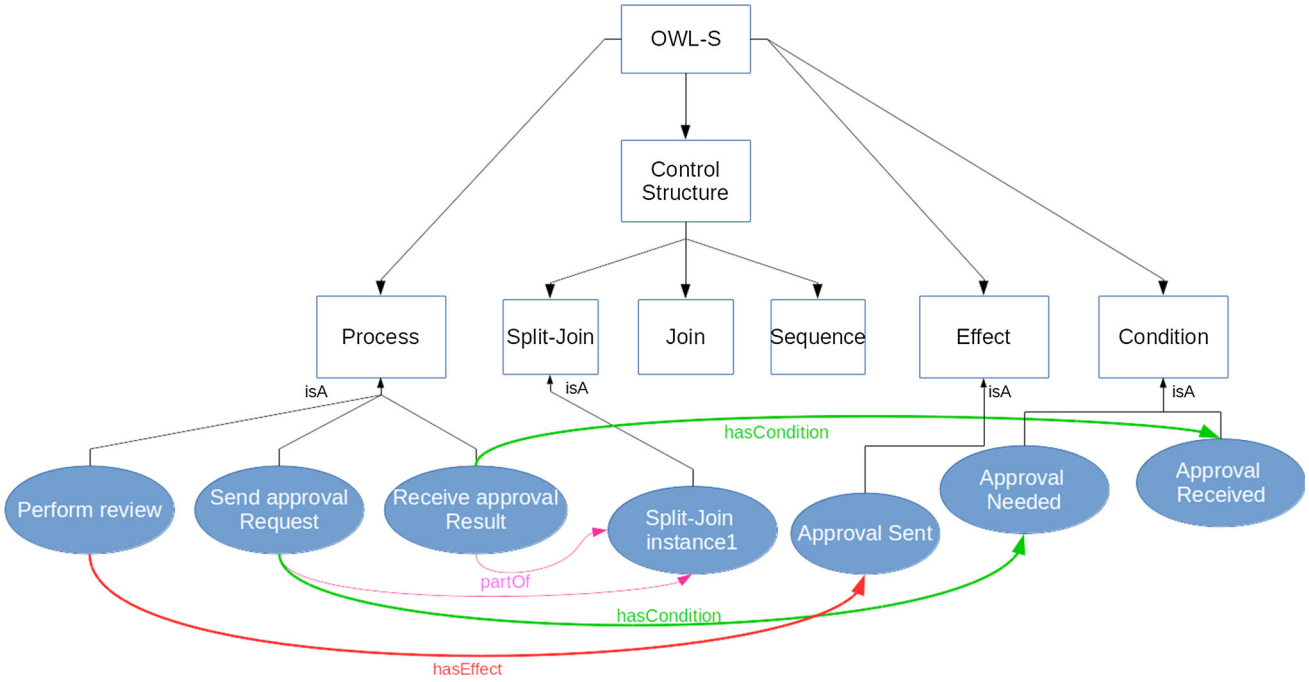


Fig. 2 Semantic representation of the business process pattern reported in Fig. 1

are executed alternatively, they are enclosed into a Split construct, and each of them has a pre-condition: the send task has a *approval needed* condition; the receive task has a *approval received* condition. Both tasks have effects: the send task will switch a *approval request sent* condition to true; the receive task will set the *approval needed* condition to false (so that no new approval requests will be sent).

- The start event on the reviewer lane has a pre-condition *approval request sent*, which is set to true when the send task of the requester lane has been executed.
- The *perform review* task has no pre-condition (already expressed by the start event), and it is represented via a simple process with an effect, which consists in switching the *Approval received* condition to true.
- The end event has no effect, so it is not modelled.
- The intermediate event which sends a notification to the requester lane is included in the effect of the reviewer task.

The pattern models a situation in which the request always receives an answer, but it does not care if it is positive or negative. It can be used to regulate access to some kind of shared resource, the use of which is limited but eventually granted to everyone asking for it. Figure 2 reports the actual semantic representation of the pattern already presented in Fig. 1. In figure, class elements belonging to the OWL-S model (process, control structure, condition and effect) are shown as white rectangles, while instances of such classes, created on the base of the business process information, are represented by blue ovals. The hasCondition (in green) property connects the send and receive tasks, modelled as instances of Process, to the corresponding conditions (Approval required and Approval received), in turn represented by instances of effect and condition classes. The hasEffect property (in pink) connects instead the process instance called perform review to the effect instance approval sent. The last action to take regards the actual annotation of the different tasks/processes with one or more service categories, which will enable the mapping to the cloud. In particular, the *send approval request*, *receive approval request* and *perform review* tasks/processes have been annotated with *notification* and *communication* services, since they actually send/receive notifications. However, the *perform review* process has been additionally annotated with the *compute* service, since a minimum computation is needed to manage the requests.

While the entire process can be built by interconnecting the services belonging to the categories retrieved via the proposed annotations, it is also possible to refer to already existing compositions of cloud services. In order to do so, the entire process has been annotated as equivalent to a well known and semantically described cloud pattern, namely the *queuing chain pattern* pattern, originally proposed by *Amazon* and described in [19]. For such a pattern, we also propose an *Azure* based version, described in [16]. The information contained in our representation comprehend not only the connection to the cloud patterns corresponding to the analysed

business process pattern, but also the direct relationships between the tasks and their cloud counterparts. That is, while using a manual annotation of patterns via the equivalentTo property supports the discovery of all the possible candidate cloud services, a pre-defined mapping to a cloud pattern will report only those services used in the cloud pattern itself. Also, precise information on how such service interoperate is immediately available.

The semantic representation of a business process pattern is very similar to the one used to describe a generic process, since the elements composing them are practically the same. However, to keep track of the patterns and of their components (aka "participants"), a *BusinessProcessPattern* OWL class has been added to the OWL-S representation, with a *hasParticipant* object property connecting it to each of the processes defined in the pattern. When creating a new business process patterns, two new steps are necessary in addition to those already reported for the definition of a semantic business process:

1. Create an individual of class *BusinessProcessPattern* and connect it to all of its participants (processes in the OWL-S representation).
2. Use the *equivalentTo* property to connect the newly created pattern with the corresponding cloud patterns. The same needs to be done for each participant of the business process pattern and corresponding service of the cloud pattern.

The result of semantically annotating the business process representation with the service ontology is reported in Fig. 3. In such a figure, we show the connection between the patterns and their participants, together with the relations existing among such participants. Each of the participants of the business process pattern is not only connected to a participant of the cloud pattern via the *equivalentTo* property, but also to other compatible services using the very same property. However, we have not reported them in the figure. The extraction of the semantic-based representation from a BPMN diagram is done automatically, via the graphical interface exposed by a prototype tool, which is described in Sect. 6.

4.1 Correctness check

As we have stated in Sect. 3, the correctness of the semantic model is verified by simulating both the BPMN and the OWL-S described processes and then by checking the obtained logs.

The example used for our case study is extremely simple, as no inputs are actually required to simulate its execution: the process simply requires that messages are sent and received in the correct order, without focusing on their content. As an instance, if the Perform Approval message contains a negative response, the process ends correctly. Table 1 contains the logs obtained from the simulated execution of the modelled Process: the results are the same for the BPMN and the OWL-S simulations. In particular, since there is an implicit synchronization due to the condition imposed on the *receive approval result* task, there will be no difference in the order of execution. The only difference may exist for the last two log entries, regarding the *Reviewer: End* and *Requester: End*, which can be swapped without altering the overall execution off the process: thus, this kind of discrepancies is ignored. As of now, the correctness check is done manually for each of the modelled Processes, but an automated check is going to be implemented in the future.

5 Support to the deployment of a process to the cloud

Once the business process has been correctly annotated, either because the user has entered the right connection to the external cloud service ontology, or she has used pre-defined patterns to build the process, it is possible to identify the needed services just by executing a set of SPARQL queries.

Lets suppose the user has built her own business process and has then annotated it using the service ontology. The query reported in Listing 1 identifies all of the candidate services which can be used in order to implement the process on cloud.

```
PREFIX process: <http://process.
   ontology.iri#>
PREFIX services: <http://services
   .ontology.iri#>
SELECT    ?task ?service
WHERE {?task process:equivalentTo
    services:serviceCategory.
       ?service aKindOf services:
          serviceCategory
      }
```

Listing 1 SPARQL query to retrieve services for the annotated process

Results of the query are reported in Table 2, where cloud services retrieved from the service ontology as possible candidate for implementing the business process on cloud are reported. Note that, once the user has retrieved the candidate services, it is still up to her to actually compose them and finally deploy the application. By slightly modifying the query it is furthermore possible to limit the results to those services belonging to a specific vendor.

If instead of annotating a brand-new business process the user composes patterns from a catalogue, then it is possible to retrieve the corresponding cloud patterns automatically, with information on the needed services and on their connections.

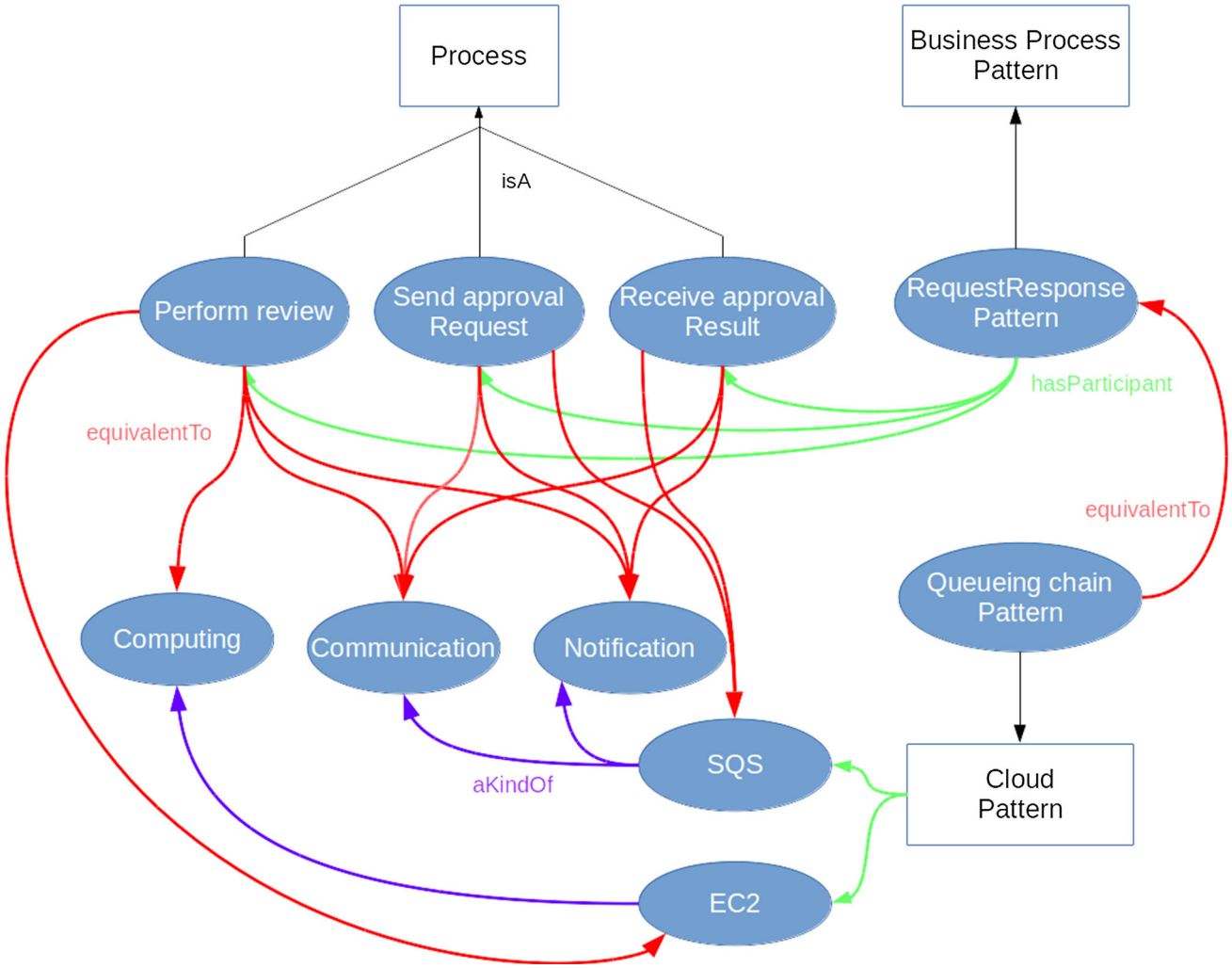


Fig. 3 Semantic annotation of the business process pattern reported in Fig. 1

Table 1 Logs obtained after the execution of the Process

Log entries
Requester: Send approval request
Reviewer: Perform review
Reviewer: Approval sent
Requester: Receive approval result
Reviewer: End
Requester: End

By referring to the pattern described in Fig. 1 in Sect. 4, the query reported in Listing 2 retrieves the corresponding cloud patterns and their composing services.

```
PREFIX process: <http://
   process.ontology.iri#>
PREFIX services: <http://
   services.ontology.iri#>
SELECT  ?cloudPattern ?service
    ?task
WHERE { process:
   RequestResponsePattern
   process:hasParticipant ?
   processParticipant.
        ?processParticipant
           process:equivalentTo
            ?cloudParticipant.
        ?CloudPattern services
           :hasParticipant ?
           cloudParticipant.
     }
     GROUP BY ?cloudPattern ?
        service ?task
```

Listing 2 SPARQL query to retrieve services for the annotated Business Process Pattern

Table 2 Results of query 1

Original task	Cloud service(s)
Send approval request	BlueMix Twilio, Amazon SimpleMail, BlueMix SendGrid, Azure NotificationHUB, Amazon SimpleNotification, Amazon Simple Queue Service, Azure Service Bus
Receive approval result	BlueMix Twilio, Amazon SimpleMail, BlueMix SendGrid, Azure NotificationHUB, Amazon SimpleNotification, Amazon Simple Queue Service, Azure Service Bus
Perform review	BlueMix Twilio, Amazon SimpleMail, BlueMix SendGrid, Azure NotificationHUB, Amazon SimpleNotification, Amazon Simple Queue Service, Azure Service Bus, Openstack Nova, Amazon EC2, Azure Virtual Machine

Table 3 reports the results of the query, organized by pattern. Note that, while the query reports a smaller number of candidate services than query 1, this does not limit the pattern-based approach. Instead, here we are collecting only those services which are known to work well together and the relations among which is documented in the retrieved patterns. Also, as pointed in [16], it is still possible to use equivalences expressed in the external services ontology to modify the original pattern and adapt it to a different vendor or to a multi-cloud environment.

Once the user has retrieved a set of proposed cloud services to implement her process, the semantic model provides information on each service regarding their exposed operations, the input and output parameters they exchange and the relations existing among them. Furthermore, the knowledge base contains stubs of code and empty skeletons which can be filled to implement the connections among the proposed services.

6 The prototype annotation tool

In order to support the semantic annotation of business processes and to allow users to define and exploit Patterns, a tool is being developed based on the semantic model described in this paper.

Currently the prototype requires the user to define the process in a standard language, namely BPMN, and then it analyses it to automatically build the semantic representation. The graphical interface of the tool, shown in Fig. 4, allows users to load their BPMN definition (which appears on the left side of the window) and to annotate it with a pre-defined or custom ontology (the structure of which is shown on the right, under the *Domain Ontology* tab). The same panel used to show the loaded BPMN will be used in future to compose business process patterns, by allowing users to select them from a catalog.

The OWL-S based representation of the BPMN is automatically obtained and the user does not see it, unless she explicitly requires to download it. The ontologies used to annotate the BPMN are shown as a hierarchical navigable directory, in which the individuals to be used as a target of the annotations represent the leaves of the structure. Since it would not be feasible for the user to navigate the whole ontology in search of the correct concepts to use for the annotations, the tool provides suggestions: based on an ontology matchmaking operated between the BPMN semantic-description and the service ontology, a set of possible annotation candidates is highlighted to help the user selecting the best suiting ones. Obviously, the user can select concepts which are not in the suggested set, or she can notify the tool that one or more of the proposed suggestions are wrong. Currently, the tool exploits properties which have not been described in the present paper, as they will be subject to changes and will be replaced by the ones reported here.

6.1 Current limitations

By using a semantic-based representation of a business process, which is in our case derived from a BPMN descrip-

Table 3 Cloud patterns components retrieved by query 2

Cloud pattern	Service	Task
AWS queuing chain pattern	Simple queue service	Send approval request
	Simple queue service	Receive approval result
	Simple queue service	
	Elastic cloud compute	Perform review
Azure queuing chain pattern	Azure service bus	Send approval request
	Azure service bus	Receive approval result
	Azure service bus	
	Azure virtual machines	Perform review

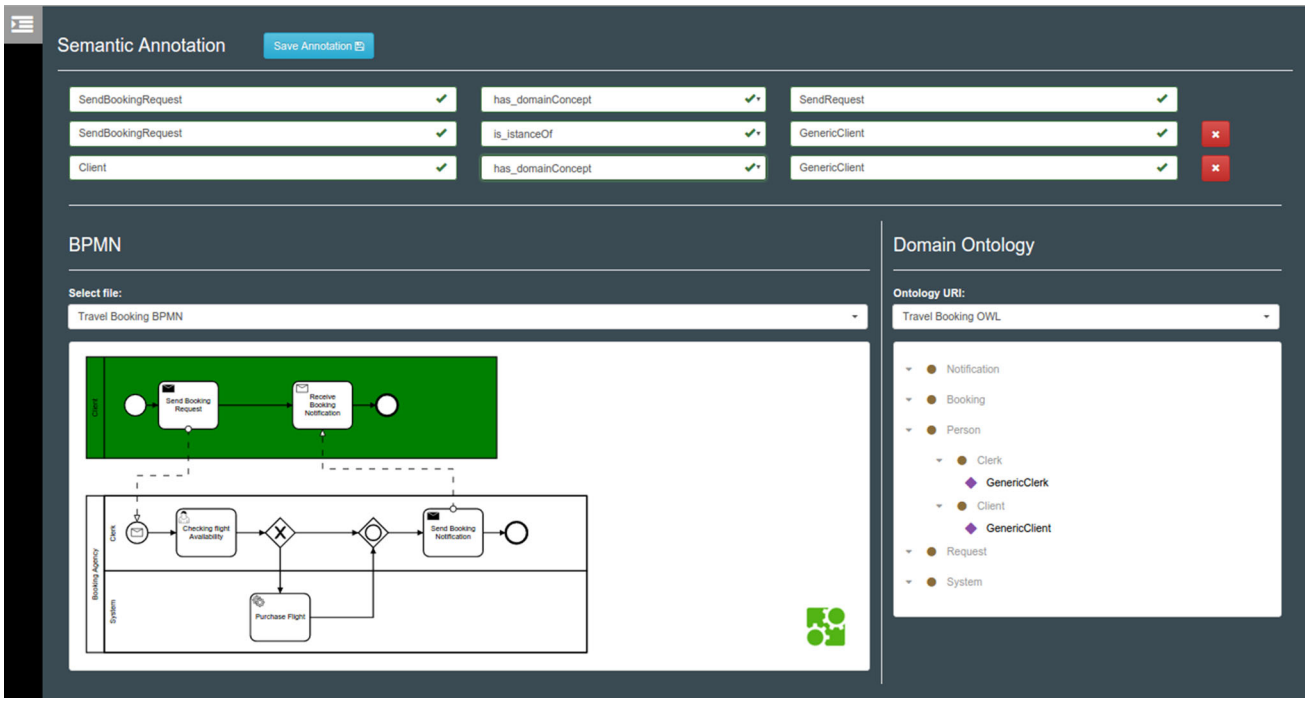


Fig. 4 The web-base interface of the prototype annotation tool

tion, it is possible to add annotations to the tasks, actors, gates and messages composing it. Such annotations can be used to retrieve a mapping between elements of the BPMN and cloud resources, that is it is possible to understand which cloud services, or a composition thereof, can be used to implement part of the process. As we have stated in this section, the prototype tool provides support to the annotation by applying ontology matchmaking techniques in order to suggest annotation candidate concepts from the service ontology. However, the results of the matchmaking strongly vary, according to the exact terms used to describe the elements of the BPMN: thus, the user may be required to navigate among many different concepts to choose the most correct ones.

The support to the deployment of the process onto the cloud is possible if the ontology used to annotate the BPMN provides the necessary information. As a default choice, the tool uses the ontology described in [16], which provides details on cloud services and can be interrogated via queries similar to the ones proposed in Sect. 5. Such queries are automatically produced, by using templates which are then filled and adapted according to the original process and the identified cloud patterns. As an instance, the very same query described in Listing 2 is very generic and can be applied to any kind of semantically described process. The template set is currently being filled with query skeletons.

If the user decides to use a different service ontology, the tool still provides support to the annotation of the BPMN, by instantiating the correct connecting properties. However, the queries will have to be adapted and re-written according to the actual ontology structure and no automatic support will be available.

At the moment, the automatic deployment of the process onto the cloud is not supported. Using our prototype tools the user can retrieve information on how to use and compose a set of cloud services, but it is up to her to manually deploy them.

7 Conclusion and future works

In this paper we have presented an OWL-S based semantic model for the representation of business processes, regardless of the original modelling language they were described in. Such semantic model has been then adapted to represent business process patterns. The represented processes and process patterns are used to support users in retrieving the list of candidate services for a cloud deployment of their modelled process. A set of SPARQL queries is used to retrieve such services or to obtain ready-to-use compositions of such services, in the form of semantically described cloud patterns. The entire procedure to transform the original business process to its semantic based representation has been reported. Even if BPMN is used for the reported example, the semantic model is suitable to represent processes regardless of the language they were originally expressed.

In the future, we intend to refine the matchmaking process, in order to provide accurate suggestions for the analysed process, and an evaluation of the matchmaking results will be

carried out. Second, we intend to provide a business process pattern recognizer which, starting from a set of already defined business process patterns, will identify their instances in the model provided by the user. In this way, the manual annotation could become completely redundant or would just integrate the automatic recognition/annotation process. Finally, in the near future an evaluation of the representation will be provided. At the moment we are considering several existing business processes, taken from web repositories [20] and tutorials [21,22] to verify if the representation is able to catch all BPMN elements, and to update it if necessary. No particular problems have been detected so far, but further tests will be run and results will be published.

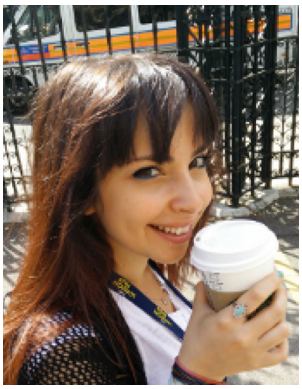
Acknowledgements This research has been supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement no 256910 (mOSAIC Project), by PRIST 2009, "Fruizione assistita e context aware di siti archeologici complessi mediante dispositivi mobili"and CoSSMic (Collaborating Smart Solar-powered Micro-grids—FP7-SMARTCITIES-2013).

References

1. van Der Aalst WMP, Ter Hofstede AHM, Kiepuszewski B, Barros AP (2003) Workflow patterns. Distrib Parallel Databases 14(1):5–51
2. BPMI.org. Business process pattern examples (2003). http://www.workflowpatterns.com/vendors/documentation/BPMN-pat.pdf. Retrieved 10/03/2016
3. Russell N, Ter Hofstede AHM, Edmond D, van der Aalst WMP (2005) Workflow data patterns: Identification, representation and tool support. In: Proceedings of the 24th international conference on conceptual modeling, Klagenfurt, Austria, October 24-28, 2005. Springer, Heidelberg, pp 353–633. doi:10.1007/11568322_23
4. Russell N, van der Aalst WMP, Ter Hofstede AHM, Edmond D (2005) Workflow resource patterns: Identification, representation and tool support. In: Advanced Information Systems Engineering. Springer, Heidelberg, pp 216–232. doi:10.1007/11431855_16
5. Thom LH, Reichert M, Iochpe C (2009) Activity patterns in process-aware information systems: basic concepts and empirical evidence. Int J Bus Process Integr Manag 4(2):93–110
6. Wetzstein B, Ma Z, Filipowska A, Kaczmarek M, Bhiri S, Losada S, Lopez-Cob J-M, Cicurel L (2007) A life cycle based requirements analysis. In: Proceedings of the 3rd workshop on semantic business process and product lifecycle management, Technical University of Aachen, Innsbruck, June 7, 2007, pp 1–11
7. Rospocher M, Ghidini C, Serafini L (2014) An ontology for the business process modelling notation. In: Proceedings of the Eighth International Conference (FOIS 2014) Formal Ontology in Information Systems, vol 267. IOS Press, p 133
8. McGuinness DL, Van Harmelen F et al (2004) Owl web ontology language overview. W3C Recomm 10(10):2004
9. Di Martino B, Esposito A, Stefania N, Maisto SA (2015) Semantic annotation of bpmn: current approaches and new methodologies. In: Proceedings of The 17 th International Conference on Information Integration and Web-based Applications and Services (iiWAS2015), ACM, pp 95–99
10. Burstein M, Hobbs J, Lassila O, Mcdermott D, Mcilraith S, Narayanan S, Paolucci M, Parsia B, Payne T, Sirin E, Srinivasan N, Sycara K (2004) OWL-s: Semantic markup for web services. http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/
11. (2009) Super-semantics utilized for process management within and between enterprises. http://projects.kmi.open.ac.uk/super/. Integrated Project (IP) financed by the 6th Framework Programme of the EU. Finished on 31 March 2009
12. Roman D, Keller U, Lausen H, de Bruijn J, Lara R, Stollberg M, Polleres A, Feier C, Bussler C, Fensel D et al (2005) Web service modeling ontology. Appl Ontol 1(1):77–106
13. Lara R, Roman D, Polleres A, Fensel D (2004) A conceptual comparison of WSMO and OWL-S. In: M Jeckle, LJ Zang (eds) Web services. European Conference, ECOWS 2004, Erfurt, Germany, September 27–30, 2004. Proceedings. Springer, Berlin, pp 254–269. doi:10.1007/b100919
14. Horrocks I, Patel-Schneider PF, Boley H, Tabet S, Grosof B, Dean M et al (2004) Swrl: a semantic web rule language combining OWL and ruleml. W3C Memb Submiss 21:79
15. Genesereth MR, Fikes RE et al (1992) Knowledge interchange format-version 3.0: reference manual
16. Di Martino B, Esposito A, Cretella G (2015) Semantic representation of cloud patterns and services with automated reasoning to support cloud application portability. IEEE Trans Cloud Comput. doi:10.1109/TCC.2015.2433259
17. IBM process manager (2015). http://www-03.ibm.com/software/products/en/business-process-manager-family. Accessed on July 2015
18. Horridge M, Bechhofer S (2011) The OWL API: a java API for OWL ontologies. Semant Web 2(1):11–21
19. AWS cloud design patterns (2015). http://en.clouddesignpattern.org. Accessed on July 2015
20. Genymodels-BPMN online examples (2016). https://repository.genmymodel.com/diagrams/bpmn. Retrieved on July 2016
21. Camunda-BPMN 2.0 best practices (2016). https://camunda.org/bpmn/examples/. Retrieved on July 2016
22. Quick BPMN-examples (2016). http://www.bpmnquickguide.com/quickguide/index.html?bpmn_examples.htm. Retrieved on July 2016

Prof. Beniamino Di Martino is full professor of Information Systems at the Second University of Naples (Italy). He participated to various research projects supported by national and international organizations, with role of Unit Responsible. He is author of 8 international books and more than 200 publications in international journals and conferences. He is Editor of four international journals and editorial board member of many international journals. His research interests include: Knowledge Discovery and Management, Semantic Web and Semantic Web Services, Semantic based Information Retrieval, Cloud Computing, High Performance Computing and Architectures, Mobile and Intelligent Agents and Mobile Computing, Reverse Engineering.

Dr. Antonio Esposito graduated in July 2013 with a master thesis on the recognition of Design Patterns from UML documentation through semantic approaches. Currently he is a Ph.D. student at the Second University of Naples and he is working on extending his previous research to analyse source code for patterns recognition and to apply semantic based technologies to solve portability and interoperability issues in Cloud Computing. His main interests are Software Engineering, Cloud Computing, Design and Cloud Patterns, Semantic based Information Retrieval.



Dr. Stefania Nacchia is a PhD Student of Computer and Electronic Engineering at the Department of Industrial and Information Engineering at the Second University of Naples. She received her Master Degree in Computer Engineering in 2016. She is involved in research activities dealing with Semantic Web, Knowledge Discovery, Cloud Computing and Business Process Management.

Dr. Salvatore Augusto Maisto is a PhD Student of Computer and Electronic Engineering at the Department of Industrial and Information Engineering at the Second University of Naples. He received his Master Degree in Computer Engineering in 2016. His interests include research activities dealing with Semantic Web, Knowledge Discovery, Cloud Computing and Business Process Management.