Extending BPMN 2.0 to Enable Links between Process Models and ARIS Views Modeled with Linked Data*

Feng Gao, Wassim Derguech, and Maciej Zaremba

National University of Ireland, Galway,
Digital Enterprise Research Institute
firstname.lastname@deri.org
www.deri.org

Abstract. Business Process Modeling Notation (BPMN) is an emerging standard for modeling business processes. In its 2.0 version it defined formal semantics to its elements which allows a process execution engine to understand how processes should be integrated and executed. However, BPMN2.0 elements use shallow String datatypes for their identification (e.g., process participants, process resources) which does not explicitly identify entities that a given element pertains to. In this paper, we propose to extend BPMN 2.0 in order to allow for linking its elements to external entities following the Linked Data principle. Our proposal leverage the extension mechanism provided by BPMN 2.0 which does not result in the alteration of the language specification. Our extensions consider the ARIS views to support better integration and collaboration from different perspectives of the enterprise systems.

Keywords: BPMN 2.0, ARIS, Linked Data, RDF.

1 Introduction

Business processes are capable to provide automated support for the collaboration between people, departments, organizations and corporations by coordinating their resources and behaviors when integrated with the information systems. The term *Business Process Management* (BPM) is sometimes referred to as the "business process optimization process". The BPM life cycle is generally divided into four phases: design, deploy, execution and analysis. Similar to the principles of developing *supportable* enterprise systems proposed in [8], BPM iterates these phases to improve the supportability.

Process modeling in the design phase plays a central role in the BPM life cycle. A recent advancement in the standardized business process modeling language of BPMN 2.0 [10] defines the formal semantics of its elements thus the process execution engines have explicit knowledge about their behaviors. Moreover, BPMN 2.0 provides means for its extensibility. However, business processes

* This work is funded by the Lion II project supported by Science Foundation Ireland under grant number SFI/08/CE/I1380 Lion-2.

W. Abramowicz, L. Maciaszek, K. Węcel (Eds.): BIS 2011 Workshops, LNBIP 97, pp. 41–52, 2011.

models defined in BPMN 2.0 do not follow Linked Data principles and are defined without linking to external information. BPMN 2.0 elements use shallow *String* datatypes for their identification (e.g., process participants, process resources) what does not explicitly identify entity that the given element pertains to. According to [1,11] both private and collaborative process models can benefit from semantic technologies, while the work in [14] elaborate the necessity of associating process models with the other views in the enterprise systems. Our aim is to improve the BPMN 2.0 process models in these aspects.

We leverage the BPMN 2.0 extension mechanism to make the process models interlinked with relevant knowledge of the enterprise systems, i.e., with the functional view, data view, organizational view and control view. We model the information in these views with our RDF based descriptions. Therefore, the extended process models can benefit from the machine processable knowledge and make improvements during the phases of the BPM life cycle. Current research efforts concentrate much on utilizing semantic technologies to provide automated execution support (i.e., ontology-backed service composition) for process models and bridging the gap between the business perspective and technical perspective, however we have a distinct interest and emphasize on using these technologies to improve the other phases in the BPM life cycle as well.

The remainder of this paper is organized as follows. We give account of the technologies and building blocks that we use in Section 2. We introduce our approach and an example scenario in Section 3. We describe related work in Section 4. Finally, in Section 5 we conclude the paper and discuss future works.

2 Background

In this paper, we leverage the **BPMN** 2.0 extensibility in order to achieve better integration between the external knowledge and business processes. The **ARIS** enterprise system modeling methodology is a guideline for our work. **Linked Data** principles are followed while we build our framework.

2.1 Business Process Modeling Notation

The BPMN language provides a graphical representation for the business managers and process designers to organize and manage their business process models. It has recently evolved to its 2.0 version which defines the formal semantics of its elements. It also provides a mapping from BPMN to WS-BPEL, which is an block-structured execution language for Web services. The vision of the BPMN 2.0 is to make the processes executable in a top-down manner where process designers can deploy, test and run developed processes without having to deal with the low level process execution details, which makes the execution of the language more "friendly" to upper layer users. Another interesting feature introduced in the latest release of the language specification is its extensibility. Both BPMN 2.0 *Flow Objects* and *Artifacts* can be extended to allow the process designers to express additional features of BPMN 2.0 process models.

2.2 ARIS Architecture

The Architecture of Information System (ARIS) provides a widely accepted enterprise modeling methodology. The ARIS framework defines five different views: organization view, functional view, data view, control view and product/service view. ARIS House shown in Figure 1 shows these five views and relations between them. As depicted in the figure, the process models are the controlling dispatcher of resources and actions, they act as the central role in the ARIS house, associating with all the other views and integrate them in a uniformed way. A process model respecting these associations between different views is understandable among involved parties despite of their different work perspectives and is therefore capable of providing better understandability, maintainability and scalability.

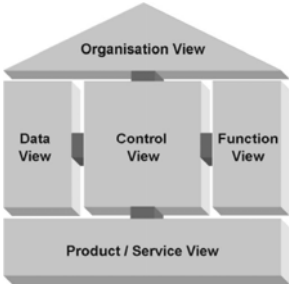

Fig. 1. The ARIS House (from [14])

2.3 Linked Data

Linked Data [3] is a set of best practices for publishing and interconnecting structured data on the Web. Linked Data provides explicit links between data from diverse domains such as social networks, organizational structures, government data and many others. The ultimate benefit of the Linked Data paradigm is the increased machine-readability of published and interconnected data. Linked Data is published using RDF where URIs are the means for connecting and referring between various entities on the Web.

Over the last years we have observed an increasing adoption of Linked Data principles and an explosion of datasets specified in RDF. Early adopters included mainly academic researchers and developers. However, more recently we have observed a considerable interest from organizations in publishing their data in RDF. Some of the most prominent examples include BBC music data[1], British government data[2] or Library of Congress data[3]. At the same time, we have observed an increasing number of public vocabularies (ontologies) and their interconnectedness. Data published in RDF uses and refers to these public vocabularies what further improves understanding of published data by data consumers.

[1] `http://www.bbc.co.uk`
[2] `http://data.gov.uk`
[3] `http://id.loc.gov`

3 Approach

The Business Process Management life cycle consists of four phases: Design, Deploy, Execution and Analysis as illustrated in Figure 2. In our approach, the extended BPMN processes are typically created and annotated in the design phase. However, we do not exclude the possibility that other phases may also produce the extensions for the BPMN documents, e.g.: some dynamic process meta-data like event triggering time, process completing time and other control information may be annotated during the process execution. BPMN 2.0 elements provide a shallow information i.e.: string types for element names. In order to improve the interoperability of the knowledge on the business function, organization and data etc, BPMN 2.0 elements should be described in more details with respects to all ARIS house views. With such extended annotations we establish the links between the process model and the different views in the ARIS house, while these views are modeled with linked data, they are interlinked with each other and can be further linked to the external open data to obtain better data integration using linked data mechanism. Thereafter, process management environments will be able to provide meaningful answers for the queries on the process models and instances with specific interests based on the common machine-understandable knowledge.

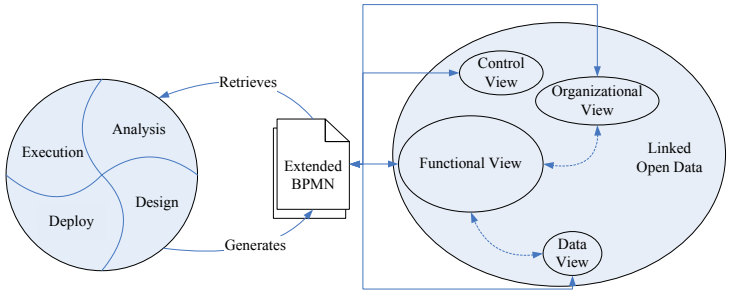

Fig. 2. The Overview of The Framework

In the following, we will detail our concept of *Structured Web Resource* for describing the information model in the ARIS views, the extension mechanism for the BPMN 2.0 and a use case scenario to demonstrate our methodology.

3.1 Structured Web Resource

We propose the Structured Web Resource (SWR) model to describe the information model from the ARIS views. SWRs are described with a series of attribute-value pairs. The *attribute* is a subclass of *rdf:Property* and its value is equal to *owl:Thing*. The basic ontology for SWR is very simple and intuitive, as shown in Table 1:

Table 1. Basic Concept Definition

:BusinessFunction	a	rdfs:Class, owl:Class.
bf:Attribute	rdfs:subClassOf	rdfs:Property;
	rdf:domain	:BusinessFunction;
	rdf:range	bf:AttributeValue.
bf:AttributeValue	owl:equal	owl:Thing.

On top of a simple framework like that, we define further rules and relations to promote the process management in the following aspects:

– **Reusability.** The *subClassOf* relation in RDFS[4] and OWL[5] does not imply the similar semantics as the inheritance relation in Object-Oriented Programming, since all the attributes and properties are defined at instance level and will not be retained through the *subClassOf* relation. We argue that this will lead to limitations on reusability. We propose to define relationships between SWR concepts and maintain a hierarchy of them. In order to achieve this, we firstly define a basic relation named *variantOf* on the resources. A resource r1 is *variantOf* a resource r2 if r1 is an instance *(rdf:type)* or a subclass *(rdfs:subClassOf)* r2, formally, for all resources r1 and r2:

$$r1\ swr:variantOf\ r2 \iff r1\ rdfs:subClassOf\ r2 \,||\, r1\ rdf:type\ r2 \quad (1)$$

We define extended relations, i.e., *extends, specifies* based on the mentioned basic relation.

Given two SWR S_1 and S_2, we denote A_1 and A_2 as their set of attributes, respectively; and for any attribute a in SWR S, we denote $S(a)$ as the attribue-value of a in S. We say S_1 *specifies* S_2 *iff*:

1. all the attributes of S_2 are also attributes of S_1,
2. for every shared attribute a_i, $S_1(a_i)$ is either equal or *variantOf* $S_2(a_i)$,
3. there exists at least one shared attribute a_j such that $S_1(a_j)$ *variantOf* $S_2(a_j)$.

Or more formally:

$$\begin{aligned} &S_1\ specifies\ S_2 \iff \\ &\{\forall a_i \in A_1 | ((a_i \in A_2) \& (S_1(a_i) = S_2(a_i)) || (S_1(a_i)\ variantOf\ S_2(a_i))\} \& \\ &\{\exists a_j \in A_1 | S_1(a_j)\ variantOf\ S_2(a_j)\}. \end{aligned} \quad (2)$$

Similarly, we say S_1 *extends* S_2 *iff*:

1. all the attributes of S_2 are also attributes of S_1,
2. for every shared attribute a_i, $S_1(a_i)$ is either equal or *variantOf* $S_2(a_i)$,
3. there exists at least one attribute a_j in S_1 that is not an attribute of S_2.

[4] http://www.w3.org/TR/rdf-schema/
[5] http://www.w3.org/TR/owl-features/

Or more formally:

$$S_1 \; extends \; S_2 \Longleftrightarrow$$
$$\{\forall a_i \in A_1 | ((a_i \in A_2) \& (S_1(a_i) = S_2(a_i)) || (S_1(a_i) \; variantOf \; S_2(a_i))\} \& \quad (3)$$
$$\{\exists a_j \in A_1 | a_j \notin A_2\}.$$

Then, a process designer is able to reengineer on a process model rapidly by leveraging these relations and inherit from a proper parent in the hierarchy of SWR classes.

- **Granular-diversity.** Unlike the mechanism used in the SUPER[6] project and sBPMN [2] that treats all process activities/objects as the instances from the ontology, we distinguish between abstract process elements and concrete process elements, since we can inherit attributes from their "super" classes. For instance, from the functional view in the ARIS house, SWRs representing the Business Function (BF) can be either abstract categories or concrete offers, which are distinguished by the *Consumability* attribute. Concrete consumable BF offers have their value of *Consumability* attribute set to *true*.

- **Dynamicity.** Instances of *swr:AttributeValue* can be statically specified. However, in realistic and dynamic scenarios we cannot assume that this is the case. *AttributeValue*s are often: (1) context-dependent (e.g., availability depends on the customer location), (2) sensitive from the business perspective (e.g., insurance price is available after multi-step interaction), (3) dynamic (e.g., currency exchange rates). Therefore, in many cases instances of *swr:AttributeValue* have to be obtained on-the-fly. We support this scenario in the twofold manner.

  We describe the features of an attribute-value in the *desc* namespace. Attributes can be derived using a *desc:Specification* that describes the rules of computing an *swr:AttributeValue* based on inputs. On the other hand, some *swr:AttributeValues* have to be dynamically obtained by interacting with an external entity without explicitly defined rules, i.e., they are either unintended to be exposed (e.g., insurance calculating formulas) or unable to be defined statically (e.g., concurrency exchange rates). We make use of the data-fetching mechanism [15] to retrieve detailed and real-time information (after analyzing consumer parameters, constraints, etc.) about a given resource. Notice that these dynamic parts should not cause any real world state changes, as invocation of these data-fetching interfaces may happen outside of the execution phase and are used for analytical or discovery purposes only. Details in the *desc:DataFetching* rely on the available technical implementation, e.g. service interface for the Web services and URIs for the RESTful services.

[6] `http://www.ip-super.org`

3.2 Extension Mechanism

Two extension mechanisms are defined in the BPMN 2.0 specification. A process element can be either extended by *Extension* element or external *Relation*. Both methods can be used for our purpose. Their suitability varies in the category of the extended process element: *Flow objects*, e.g., *Subprocesses*, *Activities* and *Tasks*, can only be extended using *Extension* element according to the BPMN 2.0 specification, on the other hand, *Artifacts* like *Group*, *Data Object* and *Text Annotation* can only be extended with the *Relation* element. To define a BPMN *Extension* element, one must specify a boolean-valued attribute named *mustUnderstand* for it. The attribute indicates the essentiality for a process engine to correctly parse this extension. A *definition* attribute of the extension tells the URI of the extension definition, where the schema of the extended information is specified. Finally these extended elements and attributes can be added to the BPMN document by referencing to their paths/query names.

For the *Artifacts*, we can use the *Relationship* element to link them to their extended information model. Adding an external *Relationship* is straightforward. A relationship between *Artifacts* exists in a non-intrusive manner (i.e., without affecting the nature of the artifacts), it can be realized simply by indicating a *Relationship* element with attributes like *type*, *id* and *direction*, sub-elements named *source* and *target* will give the relative references.

We propose to extend the process models in BPMN 2.0 in the following views:

- **Organisation View** defines the roles, the participants, the organizational entities and the relationships between them. BPMN 2.0 *Activities* should be annotated with organizational information like: who is responsible with executing the activity (individual or group), which organization/group does he belong to and what role dose he have. Similarly, the meaning of BPMN 2.0 *PartnerRole*, *Participant* and *PartnereEntity* should be leveraged from a basic string label to that given by concepts from dedicated organizational ontologies for roles, participants and organizational entities respectively.
- **Data View** describes the information objects, the concepts in the messages exchanged by process tasks. Many BPMN 2.0 elements are using data which is encapsulated in the BPMN *ItemAwareElement*. There are also many BPMN elements related to *Events* and message flows that carry data with them. They should be annotated with concepts from domain specific ontologies.
- **Control View** describes the control flow of BPs that carry out the business operations inside the enterprise and it is an integration view between the other views. During business process execution, information like the occurrence time of an activity or event are very important for monitoring and auditing. BPMN *Activity*, *Event* and *EventDefinition* instances should carry with them such information.
- **Function View** defines the functions required to satisfy the objective of the enterprise. BPMN *Activity* should be defined as a domain specific business function instead of just a label.

3.3 Use Case Scenario

We demonstrate our approach with a use case in a pizza ordering scenario[7]. The process flow is shown in Figure 3. According to what we proposed in 3.2, Pool and Roles in the Lanes can be extended to link with the Organizational View, Tasks can link to the Functional View, Events can link to the Control View and finally Messages to the Data View. Due to the limit of space, we will take only the extensions for the Functional View as an example to detail the methodology we use. Extending to the other views will not have a major difference. More specifically, we will elaborate how the informational model of the *Provide ingredients* functionality is built and how it is associated with the task. We will also detail the extension mechanism through *Relation* elements without the functional description of the group element.

Building the Functional Model. To provide customers with correct information about pizza ingredients, the clerk will use an internal service to query on the available servings. We build a hierarchy for the query functionality with a top level category (coarse-grained) and a concrete querying service (fine-grained) for the specific pizza shop. The top level category for the ingredient querying function is a *QueryService*. Table 2 shows a part of the *QueryService* Business Function.

Table 2. Query Service Functional Model

SWR	Attribute	AttributeValue
strg:QueryService	a	ser:ServiceVariant;
	swr:extends	ser:ServiceVariantRoot;
	ser:consumability	xsd:Boolean;
	ser:hasCoParam	strg:QueryRequest;
	ser:hasProParam	strg:ListOfGoods;

The "*swr*" namespace refers to the SWR basic ontology, "*strg*" describes the ontology for the storage of goods, "*ser*" refers to the general service ontology. The *swr:extends* attribute indicates its inheritance from an upper level service category, the *root* service. *bf:consumability* is a boolean value to indicate that the service is abstract and is agnostic to the concrete consumability, its children in the class hierarchy that specifying this attribute will tell if it is directly consumable. The *CoParam* and the *ProParam* refer to the consumer parameter and provider parameters, respectively. The provider of the business functionality is expecting a query request from the consumer, and will provide a list of goods based on the request. The request message and response message can also be a part of the information model in the data view, and hence we have a information model interlinked from different views. The abstract business function category can be further inherited to model a detailed category, namely *IngredientQueryService*, with which we use to model the *Provide ingredients* task. Table 3 shows a part of the functional model.

[7] Adapted from a similar one available at `http://www.omg.org/spec/BPMN/2.0/examples/PDF/`

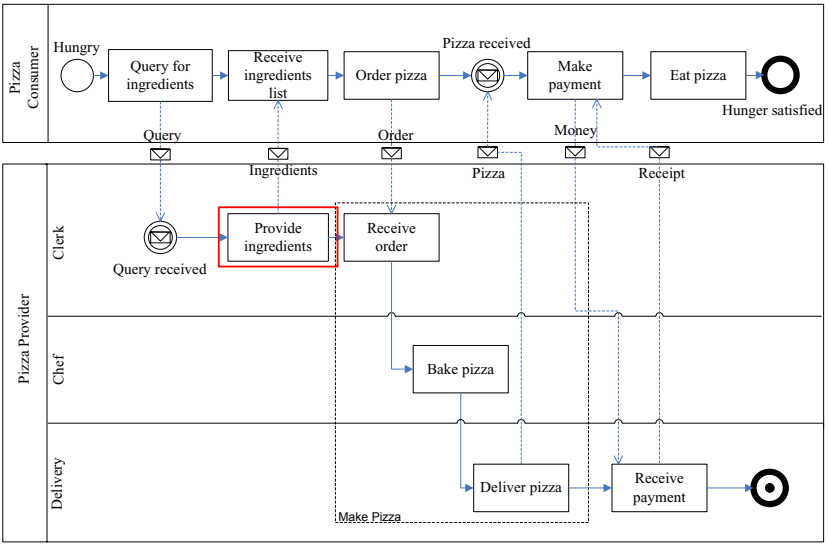

Fig. 3. The Pizza Purchasing Ordering Process Model. The process begins when a customer feels hungry, then he will pick up the phone and make an pizza order, a clerk answering the telephone in the shop will check the ingredients for the customer. According to the list of ingredients provided, the customer will order his pizza. When the clerk receives the order from the customer, he will inform the baker to bake to pizza according to the customer's demand. The delivery boy takes the pizza when ready and delivers it to the customer. The customer will pay for the pizza on receiving it, and the receipt will be provided by the delivery boy. After consuming the pizza, the process ends.

Concrete service offers should indicate the information about the service providers, and such information can also be part of the organizational view. In our case, the service provider can be specified to a person, department, or pizza company etc. According to our defined semantics for the *extends* and *specifies* relationship, only the different part needs to be described comparing from the abstract functional model. The *consumability* is set to "true" to indicate that this concrete service is consumable. The *pizza:Ingredients* resource is an instance of the *ProParam* in the abstract model, its dynamicity is declared with the type of *desc:DynamicParameter*, in which the namespace "*desc*" is an ontology for describing the details of the parameters used in SWRs as we mentioned before. Dynamic parameters will have a binding element whose value is a segment of *SPARQL* construction query. The segment is expected to be utilized by the application to build the concrete instance of the resource at run-time, based on inputs either directly from user (using *desc:specification* we described before) or from the result of the data-fetching mechanism.

The last element in the table is the data-fetching element that specify the ways of retrieving the data, e.g., service endpoints, interfaces etc. Data-fetching is used here to provide real-time and on-demand details for queries from process analyzer

Table 3. Pizza Ingredient Query Functional Model

SWR	Attribute	AttributeValue
strg:IngredientQueryService	a	ser:ServiceVariant;
	ser:hasProvider	strg:QueryServiceProvider;
	ser:extends	strg:QueryService;
	ser:consumability	xsd:true;
	ser:hasProParam	pizza:Ingredients.
pizza:Ingredients	a	strg:ListOfGoods,
		desc:DynamicParameter;
	desc:hasBinding	"some_sparql_construct_pattern"
	desc:hasDataFetching	:IngredientQueryDataFetching.

or discovery agent, without invoking the process and having post conditions. The technical information contained in *IngredientQueryDataFetching* for executing the data-fetching will not be detailed due to the space limit.

Referencing from BPMN. Referencing to the external business function models with an *Extension* requires a extension definition schema for an XML attribute, which can be specified as:

```
<xsd:attribute name='hasBusinessFunction' type='xsd:anyURI'/>.
```

We can attach this extended attribute to a *task* while having indicated the extension element for the process, as shown below:

```
<extension mustUnderstand="true" definition="ext:hasBusinessFunction" />
<process>
<task id='id-t3' name='Provide ingredients'
ext:hasBusinessFunction='strg:IngredientQueryService'>
...
</task>
...
</process>.
```

As the readers may have noticed, three tasks in the use case are grouped together with a dotted box, indicating they serve as sub-functionalities of the "make pizza" business function. Logically this may be a bad example, however, the intent is to demonstrate how we could extend the *Artifacts* using a *Relationship* element. We can define a relationship with the id "er-1" and relate the group "group1" to the business function "MakePizza" as follow:

```
<relationship type='swr-reference' id='er-1' direction='both'>
<documentation>
Reference from a group of activities to their information model from the
functional view
</documentation>
<source ref='src:group1'/>
<target ref='strg:MakePizza'/>
</relationship>
```

4 Related Work

We propose SWR to be a lightweight advancement for describing concepts with RDFS and OWL ontologies, respecting the concreteness and dynamicity of the concepts. SWR is similar to WSML *concept* [13] which also introduces inheritance between classes. However, WSML focus on the service domain and is considered "heavy weight" as it is equipped with explicitly defined axioms, relations and different mediators. However it's not standardized and does not benefit from linked data.

A relevant research work to ours is the sBPMN [2], it has defined semantic variants of BPMN language. It used a WSML ontology language [4] to semantically describe business processes. It aimed to provide the Semantic Business Process Management (SBPM) [6]) framework. However, the approach taken by the project resulted in the alteration of language specification and has not been standardized. In our work, we use an extension mechanism of BPMN 2.0 which is compatible. Another work of BPEL4SWS [9] follows the similar principles we use to extend a "hosting" language with existing mechanisms to enable the semantic relations non-intrusively. However it is mainly concerned with the semantical process executions, including semantical service discovery and composition. While we are also interested in providing dynamic information about the process model, thus more concrete and up-to-date information can be realized during process analysis and discovery.

Other approaches such as those proposed in [7,12,5] proposed BPMN extensions while considering some particular perspectives. [7] proposed a BPMN extension with business process goals and performance measures, [12] proposed a BPMN extension for modeling security requirements and [5] proposed a BPMN extension with a resource information layer. However, our work is aligned with the ARIS views and categorize all the essential information in the enterprise applications accordingly. It supports better integrations and collaborations from different perspectives of the enterprise systems.

5 Conclusion and Future Work

In this paper we justify the need of providing semantics to the process models from different perspectives according to ARIS architecture. We elaborate our novel approach of leveraging the BPMN 2.0 extension mechanism to introduce a compatible way of establishing the semantic links for process models in BPMN 2.0. We also propose a *StructuredWebResource* framework to model the information related in the enterprise applications and provide means to enhance the reusability, granular-diversity and dynamicity. We argue that we can make improvements with semantic technologies not only in the execution phase but also during other phases in the BPM life cycle. As a future work we plan to expand the relations in SWRs for more complex resource clouds beyond class hierarchies.

References

1. Abramowicz, W., Haniewicz, K., Kaczmarek, M., Zyskowski, D.: Semantic modelling of collaborative business processes. In: eKNOW (February 2009)
2. Abramowicz, W., Filipowska, A., Kaczmarek, M., Kaczmarek, T.: Semantically enhanced business process modeling notation. In: SBPM (2007)
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
4. de Bruijn, J., Lausen, H., Polleres, A., Fensel, D.: The Web Service Modeling Language WSML: An Overview. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 590–604. Springer, Heidelberg (2006)
5. Großkopf, A.: An extended resource information layer for bpmn. Systems Engineering, 1–17 (2007)
6. Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic business process management: a vision towards using semantic web services for business process management. In: ICEBE (October 2005)
7. Korherr, B., List, B.: Extending the epc and the bpmn with business process goals and performance measures. In: ICEIS (3), pp. 287–294 (2007)
8. Maciaszek, L.A.: Roundtrip architectural modeling. In: Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modelling, vol. 43. APCCM (2005)
9. Nitzsche, J., van Lessen, T., Karastoyanova, D., Leymann, F.: BPEL for Semantic Web Services (BPEL4SWS). In: Chung, S., Herrero, P. (eds.) OTM-WS 2007, Part I. LNCS, vol. 4805, pp. 179–188. Springer, Heidelberg (2007)
10. OMG: Business Process Model and Notation (BPMN) Version 2.0 Beta 2 (2010), `http://www.omg.org/spec/BPMN/2.0`
11. Pedrinaci, C., Domingue, J., Brelage, C., van Lessen, T., Karastoyanova, D., Leymann, F.: Semantic business process management: Scaling up the management of business processes. In: ICSC (2008)
12. Rodríguez, A., Fernández-Medina, E., Piattini, M.: A bpmn extension for the modeling of security requirements in business processes. IEICE Transactions 90-D(4) (2007)
13. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web service modeling ontology. Applied Ontology 1(1) (2005)
14. Scheer, A.-W., Schneider, K.: Aris architecture of integrated information systems. In: Handbook on Architectures of Information Systems (1998)
15. Zaremba, M., Vitvar, T., Moran, M.: Towards optimized data fetching for service discovery. In: ECOWS (November 2007)