Transformation of BPMN Model into an OWL2 Ontology

Mariem Kchaou[1], Wiem Khlif[1], Faiez Gargouri[1] and Mariem Mahfoudh[1,2]
[1]*Mir@cl Laboratory, University of Sfax, Sfax, Tunisia*
[2]*University of Kairouan, Kairouan, Tunisia*

Keywords: BPMN Model, Ontology, OWL2, Transformation Rules, Business Context.

Abstract: Each enterprise needs to have a clear vision of its business processes in order to increase the quality of its products/services. To fulfil this need, many enterprises rely on an Information System (IS). Most of the previous systems were previously framed by applying business process model. In addition, the current trend expresses a growing demand of reusing data from older information systems, which is very beneficial for the implementation of semantic knowledge. The transformation of a BPMN model into an ontology leads to reduce cost by reusing older systems. Although many studies are elaborated for transforming BPMN model into ontology, they have not fully proposed the transformation rules. This paper suggests the addition of rules for transforming annotated BPMN models to ontologies by accounting for the semantics of the BPMN model, and providing for all business objects and activities. In addition, the transformations have the merit of generating the OWL2 graphical representation.

1 INTRODUCTION

An automated information system (IS) gives important support to the business process if its capacities are best exploited. Indeed, the crucial advantage of designing system by utilizing BPMN notation lies in its ability to describe, and reflect the real world of information systems better. Furthermore, it has also got the further support of developers. Hence, the BPMN notation has gradually gained its popularity. However, the specification is comprehensive and partially conflicting. Therefore, several researches present an ontology that provides a formal definition of BPMN and can be used as a knowledge base (Annane et al., 2019). It is a formal representation of knowledge and consists of statements that define concepts, relationships, and constraints. According to (Noy and McGuinness, 2001), an ontology allows a shared common understanding, the reuse and the analysis of domain knowledge. An ontology is, therefore, suited to represent the BPMN metamodel.

In this context, many researchers proposed methods for transforming a Business Process Model (BPM) to the OWL2 ontology (Annane et al., 2019) (BPMN-onto, 2019). These works are based on the graphical notation (Annane et al., 2019) or on the XPDL language (Figueiredo and Oliveira, 2018).

Although these studies are elaborated for transforming BPMN model into ontology, they have not fully proposed the transformation rules. In addition, these transformation rules neglect the semantic information related to BPMN elements. A lack of information may reduce the number of possible components that can be found. For instance, the relation semantics between classes and their type such as “is composed of” and “is a part of”, etc.

More specifically, we propose seven transformation rules to transform BPMN model to an OWL2 graphical representation. These transformation rules use an *annotated* BPMN model and the proposed business context that describe semantic information related to BPMN elements.

The remainder of this paper is structured as follows: Section 2 overviews the business context and discusses related work. Section 3 shows the transformation rules to generate an OWL2 graphical representation from an annotated BPMN model with its business context. Section 4 evaluates the quality of the generated OWL2 graphical through the recall and precision rates and illustrates our transformation rules through a case study. Finally, Section 5 summarizes the presented work and outlines its extensions.

Kchaou, M., Khlif, W., Gargouri, F. and Mahfoudh, M.
Transformation of BPMN Model into an OWL2 Ontology.
DOI: 10.5220/0010479603800388
In *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2021)*, pages 380-388
ISBN: 978-989-758-508-1

2 BACKGROUND

2.1 BPMN Language

BPMN (ISO/IEC 19510. 2013), adopted by the OMG group, is the most used notation for modelling business processes (BP). The graphical objects are organized into several categories: Activity, Data, connecting objects (sequence/message flows), participants (lane, pool). An activity can be a simple representing Task or composed representing a sub-process. In BPMN 2.0, there is different Task types such as service Task which is used when an external service is called to perform a task. Send task is designed to send a message to an activity, process, or lane, while receive task is designed to wait for a message from an activity, process, or lane. Activities and processes often need data objects and data store in order to be realized. Connecting Objects (sequence flow, message flow) connect the Flow Objects to each other or other information to create the basic structure of a BP. Participants represent Pools and Lanes elements. A pool can be a specific entity or a role. It is divided into one or more lanes.

2.2 Business Context

Before introducing the business context, we extend the BPMN source meta-model presented in (Khlif et al., 2018) (See Figure 1).

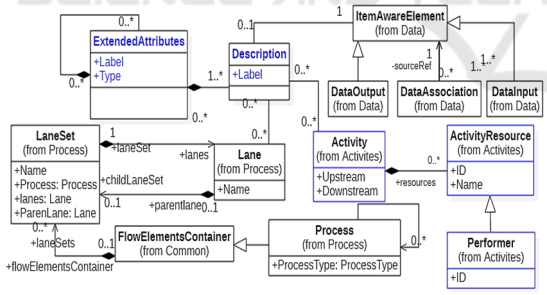

Figure 1: BPMN meta-model.

For each BPMN element, (Khlif et al., 2018) associate a *Description* that adds a specific information to BPMN elements in terms of the relationships between them. The *ExtendedAttributes* class specifies the properties of each BPMN element.

In (Khlif et al., 2018), the authors describe the business context to annotate different BPMN elements. The business context add semantic and structural information specific to all BPMN elements.

Activity node can be simple, representing a task, or composed that expressing a sub-process. We enhance each activity with a business context that contains the following information: 1) the unique activity identifier (ID), 2) Lane ID which is the unique identifier of the lane containing the activity, 3) Performer (actor) ID that express the unique identifier of the actor responsible of performing the activity, 4) Upstream and downstream ID is the unique identifier of the activity on which this activity directly depends, 5) extended attributes which can be a pure value or a complex one representing a business entity, 6) activity description indicating the relationships between the business entities and/or the activity's extended complex attributes, 7) resources expressing the data objects/stores that are required by an activity to fulfil its goal. The resources are described in terms of name, extended attributes and description.

The data objects/stores' extended attributes and description have the same semantic than the activity's extended attributes and description.

The lane and pool elements are described with the following informations: 1) Unique identifier of lane (IDL)/pool (IDP), 2) their labels, 3) Lane Description (LD)/Pool Description (PD) to indicates the semantic relation between the lane/pool and 4) the tasks/data object or stores (respectively the lanes or tasks/data object or stores) that belong to it, 5) E*xtended Attributes to* describe the lane/pool properties. As the same of the extended attributes related to the activity, each one can be a pure value or complex. The annotated BPMN elements will be transformed into OWL2 components.

2.3 Related Work

Many researchers proposed a number of methods for transforming a Business Process Model to the OWL2 ontology (Annane et al., 2019) (BPMN-onto, 2019).

In this context, (Annane et al., 2019) developed the BBO (BPMN 2.0 Based Ontology) ontology for business process representation, by reusing existing ontologies and meta-models like BPMN 2.0. Another ontology (BPMN-onto, 2019) has been automatically extracted from BPMN 2.0, but there is no documentation about how it was generated. Moreover, this ontology contains no annotations and less information than the specification document. (Figueiredo and Oliveira, 2018) propose a systematic process for the automatic generation of an ontology from a BP model transformed to the XML Process Definition Language.

(Ternai, et al., 2016) propose an approach to transform the BP into process ontology and to combine it with the knowledge base as a domain ontology in a well-controlled solution.

Overall, the above works focus on transforming

BPMN model into ontology from its graphical notation (Annane et al., 2019) or from the XPDL language (Figueiredo and Oliveira, 2018). However, they have not fully proposed the transformation rules. In addition, these works rely only on the BPMN elements to produce the corresponding component in OWL2. In fact, several semantic relations are not extracted such as "is composed of", "is a part of", etc.

In this paper, we propose a set of transformation rules to transform BPMN model to an OWL2 graphical representation. These transformation rules use business context that describe semantic information related to BPMN elements.

3 TRANSFORMATION OF BPMN MODEL INTO OWL2

In this section, we propose a set of rules for transforming BPMN into OWL2. The proposed rules exploit the business concept that describes semantic information related to BPMN elements i.e. activity description, lane description, etc. (Khlif et al., 2018).

Before introducing our transformation rules, we define the following notation: in the BPMN model, the business object is Bo_i (Bo_j), the extended attribute is *Exatt*, and the relationship between business objects (Bo_i, *Boj, etc)* is expressed by the description field (*FD*). In OWL2, the class is $C(Bo_i)$, data type property is *Exatt* and object property is Bo_iRBo_j (*R* is the relationship between two classes Bo_i and Bo_j).

The OWL2 (W3C, 2005) graphical representation is described by WebVOWL editor, an open source software for the visualization of ontologies.

The transformation rules are based on an annotated BPMN model to generate an aligned OWL2. It supposes that:

a. The description field of BPMN element follows this linguistic pattern: «BusinessObject +VerbalGroup + [Quantifiers] +BusinessObject».
b. The BPMN tasks are labeled according to the following linguistic syntax patterns:
   - ActionVerb +BusinessObject |NominalGroup]
   - CommunicationVerb + BusinessObject |NominalGroup + [[to ReceiverName] | [from SenderName]]

We mean by BusinessObject any entity that describes the business logic. The NominalGroup is a set of pre/post-modifiers, which are centred around a HeadWord that constitutes the BusinessObject. The pre-modifiers (respectively post-modifiers) can be a noun, adjective, or an ed/ing-participle (respectively, a noun, adjective, or adverb). The VerbalGroup indicates the relationship type between BusinessObjects. Relationships' semantic (semantic of VerbalGroup) must follow these linguistic patterns: BusinessObject+VerbalGroup+ Quantifiers +BusinessObject. The verbal group indicates the relation type as follows:

- "is entirely made of" or "is part of" expresses an aggregation relationship between the business objects.
- "is composed of" designates a composition relation
- "Is a" denotes the generalization/ specialization relation. We note that the generalization can be disjoint or complete and disjoint. If the verbal group doesn't belong to this set of keywords or any synonyms, then it specifies a relation between the business objects. The *Quantifiers* gives an idea of the multiplicity.

R1. For each description field of BPMN element, extract the object property in OWL2 and the cardinality between the generated classes in OWL2 according to the semantic of VerbalGroup. If it is:

a. "Boi" is entirely made of "Boj" or "Boi" is part of "Boj" or any synonyms:

We note that "is entirely made of" indicates that the existence of the part is dependent on the whole. It represents the relationship between the two business objects in which the business object of an entity consists of some objects of the other, but does not exist in its interior. It is a combination of asymmetric business objects and this relation indicate that a business object is not associated with itself.

Therefore, the existence dependency leads to transform "is entirely made of" into a pair of inverse object properties with corresponding constraint. The transformation rule is presented as follows (Figure 2):

- Add two classes $C(Bo_i)$ and $C(Bo_j)$.
- Add the object property Bo_jRBo_i for representing the relationship between classes $C(Bo_i)$ and $C(Bo_j)$, with domain is $C(Bo_j)$, range is $C(Bo_i)$;
- Set InverseFunctionalObjectProperty for object properties Bo_jRBo_i.
- Set IrreflexiveObjectProperty for the object attribute Bo_jRBo_i.
- Set AsymmetricObjectProperty for the object attribute Bo_jRBo_i
- Add min/max cardinality constraint to the corresponding classes.

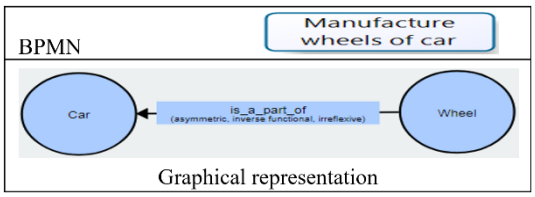

Figure 2: **R1** illustration: Case (a).

b. "Bo_i is composed of Bo_j" or any synonyms:

The expression "is composed of" in the *description field* represents the relationship between two business objects in which the business object of an entity consists of all objects of the other business object. It correspond to a pair of inverse object properties in OWL2 with corresponding constraint.

The term "is composed of" is a combination of irreflexive, so when transforming into OWL2, it must use the "IrreflexiveObjectProperty" syntax. "is composed of" expression is not recursive, a business object isn't composed with itself, so must set the AsymmetricObjectProperty syntax. Therefore, we propose the transformation rules as follow (Figure 3):

- Add two classes $C(Bo_i)$ and $C(Bo_j)$.
- Add two object properties for representing the relationship between classes $C(Bo_i)$ and $C(Bo_j)$: Bo_iRBo_j with domain is $C(Bo_i)$, range is $C(Bo_j)$; Bo_jRBo_i with domain is $C(Bo_j)$, range is $C(Bo_i)$.
- Set IrreflexiveObjectProperty for the object attribute Bo_jRBo_i.
- Set AsymmetricObjectProperty for the object attribute Bo_jRBo_i.
- Add min/max cardinality constraint to the corresponding classes.

In Figure 3, the description field contains a relationship "*is composed of*" between the "book" and the "chapter". Each book is composed of chapters. These business objects are transformed to their corresponding classes which are related by two object properties: "is composed of" and "is founded in". In particular, the object property "is founded in" has the Irreflexive and Asymmetric characteristics.

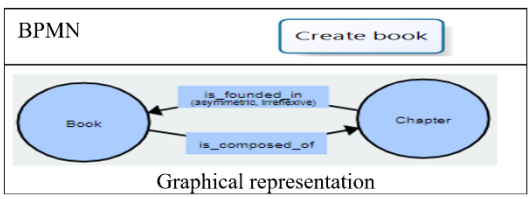

Figure 3: **R1** illustration: Case (b).

c. "Bo_i Is a/an Bo_j",

The expression "is a/an" in the description field shows the top and bottom views of business objects hierarchy. It is classified in two kinds of constraints which are also annotated in the description field: completeness ("disjoint and complete") and disjointness ("disjoint"). The top and bottom views of business objects correspond to the class and subclass in OWL2. The constraints "disjoint" corresponds to the syntax "owl:disjointWith" and the constraint "disjoint and complete", corresponds to "owl:disjointUnion" in OWL2. Therefore, we present the transformation rules as follow (See Figure 4):

- Add two classes $C(Bo_i)$ and $C(Bo_j)$ which is subclass of $C(Bo_i)$.
- If the description field contains the term "disjoint", use the syntax "owl:disjointWith".
- If the description field contains the term "disjoint and complete", use syntax "owl:disjointUnion"..

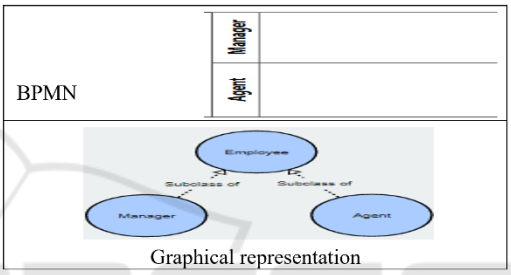

Figure 4: **R1** illustration: Case (c).

d. Else, transform the expression in the description field into an object property. Therefore, we present the transformation rules as follow (See Figure 5):

- Add two class $C(Bo_i)$ and class $C(Bo_j)$
- Add two inverse object properties Bo_iRBo_j and Bo_jRBo_i which show relationship between class $C(Bo_i)$ and $C(Bo_j)$.
- Add min/max cardinality constraint to the corresponding object properties.

In Figure 5, the description field of the task "Create customer account" indicates a relationship between two business objects "Customer" and "Account". The business object Customer has (1..n) accounts but the business object Account is related to one customer. These business objects are transformed to classes in OWL2 that are related by two inverse object properties "Has" and "concerns".

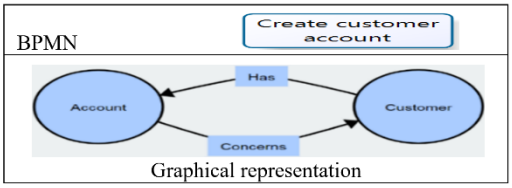

Figure 5: **R1** illustration: Case (d).

R2. For each extended attribute of the BPMN element, transform it to (See Figure 6):

a. Data type property in the OWL2 if the type of the extended attribute merely represents a pure value or primitive data type.
b. Else, a new class with the name extendedAttributeLabel, and an object property between the two generated classes by applying R1 if the type of the extended attribute is complex representing a business object.

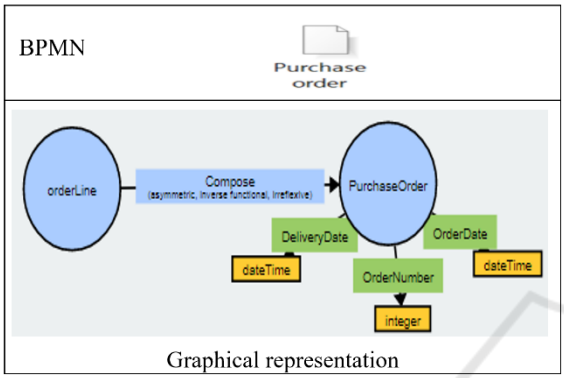

Figure 6: **R2** illustration.

R3: Transform a business object pool/lane representing a process respectively to a class and a subclass in OWL2.

R3.1: The business object pool/lane is transformed respectively to class/subclass in OWL2. The class name depends on the participant type which is a performer or an entity. If the participant is a performer, then the class name corresponds to the business object name and the word "Space" or "area". Else, the class name is a concatenation of the business object name and the word "Management" (Figure 7).

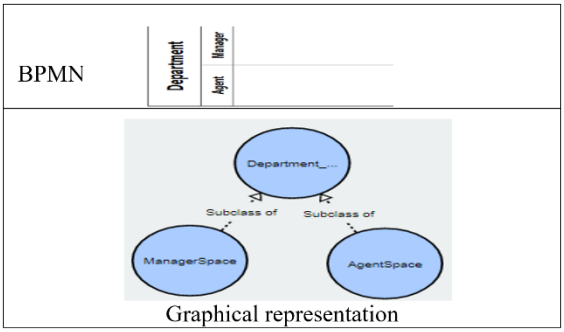

Figure 7: **R3.1** illustration.

R3.2: The business object (pool/lane) has as many extended attributes. It is transformed to class. The class name corresponds to the business object (pool/lane) name. The extended attributes are transformed to data type properties (See **R2**). Each business object (pool/lane) has a description field which is transformed to object properties (See **R1**).

In Figure 8, the business object "Department", "Manager" and "Agent" are transformed into classes. The description field of business object Department (pool), defined in its business context, indicates that the "department contains many mangers and agents". So that, this description is transformed into object properties between the whole side (Department class) and the part side (Manager and Agent classes) as well as the minimum cardinality is 1 on the part side.

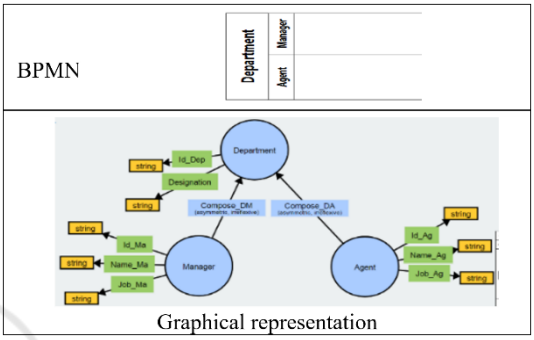

Figure 8: **R3.2** illustration.

R4. For each service task, we apply **R1** and **R2**
In addition, if the service task label respects the renaming pattern:

R4.1: « Action verb + BusinessObject », then transform (See Figure 9):

a. The BusinessObject into a class having the same name of the BusinessObject.
b. The actor that performs the tasks into a class with the same name of that actor.
c. The Action verb into an object property between the generated classes in (a) and (b).

In figure 9, the description field of the task "Create account" indicates that the "Agent" represents the actor who creates the account. The Business Object "Account" corresponds to a class with a name "Account" in OWL2. The actor "Agent" corresponds to a class having the same name in OWL2. The Action verb "Create" is transformed to an object property between the "Account" and "Agent" classes.

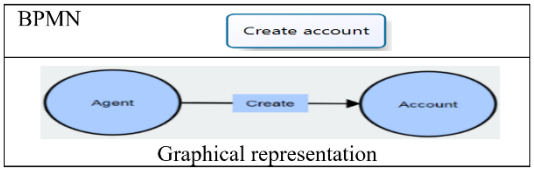

Figure 9: **R4.1** illustration.

R4.2: « Action verb + NominalGroup», apply **R1**. Then apply **R4.1** on the HeadWord and transform:

a. The HeadWord into a class in OWL2.

b. The pre/postmodifier into data type property in the class corresponding to the HeadWord if it is a noun that simply represents a pure value. The data type property has the same name of pre/postmodifier (See Figure 10);

c. The *pre/post*-modifier into a class with the name pre/post-modifier if it is a complex noun. The relation between the HeadWord and the *pre/post-modifier* to two object properties between the two generated classes (HeadWord and pre/postmodifier) (See Figure 11).

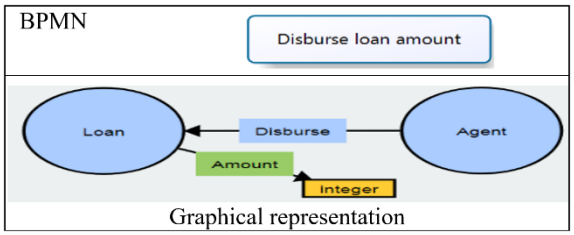

Figure 10: **R4.2** illustration.

We note that adjectives, and ed/ing-participles pre-modifiers as well as adjectives, and adverbs postmodifiers are ignored.

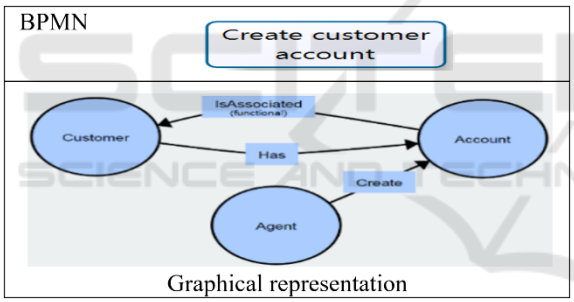

Figure 11: **R4.2** illustration.

R5. For each send/receive task, we apply **R1** and **R2**. In addition, when the task name follows this pattern:
R5.1: «CommunicationVerb+ BusinessObject + [[to ReceiverName] | [from SenderName]] », transform (See Figure 12):

a. The BusinessObject into a class with the same name of BusinessObject.

b. The "to ReceiverName" and "from SenderName" into two classes' SenderName and ReceiverName. These classes are related to the created Business Object class in (a); Then, add new data type property email or phoneNumber in each class with a name SenderName and ReceiverName;

c. The CommunicationVerb into object property between the SenderName and ReceiverName classes.

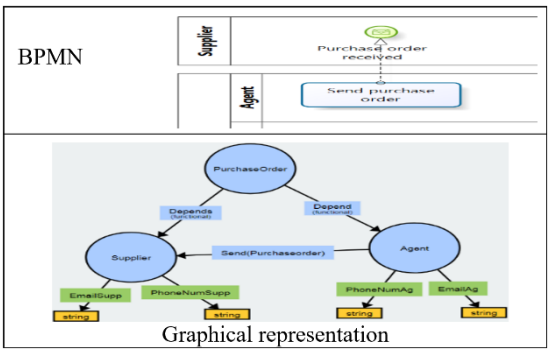

Figure 12: **R5.1** illustration.

R5.2: « CommunicationVerb+ NominalGroup + [[to ReceiverName] | [from SenderName]]», apply **R5.1** on the HeadWord and transform:

a. The HeadWord into a class in OWL2.

b. The pre/postmodifier into data type property in the class corresponding to the HeadWord if it is a noun that simply represents a pure value.

c. The *pre*/post-modifier into a class with the name pre/post-modifier if it is a complex noun. The relation between the HeadWord and the pre/post-modifier into two object properties between the two generated classes (HeadWord and pre/postmodifier).

We note when this expression [[to ReceiverName] | [from SenderName]] is omitted, then we can extract this semantic information from the description field of the activity element according to **R1**.

R6. Transform to a class each data store/object, identified by a name, if it is not already generated. The class name has the same data object name.

R7. If the business object (pool/lane) sends or receives respectively a message/sequence flow to/from another one, then transform the business object pool/lane to class/subclass in OWL2 (See **R3.1**) and the message/sequence flow into unidirectional object property with the name "Depends" between the associated sub classes.

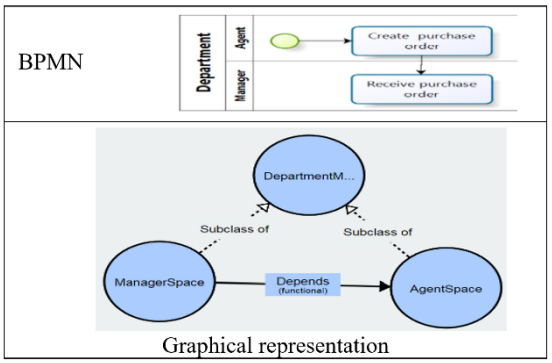

Figure 13: **R7** illustration.

4 EVALUATION OF OWL2 GRAPHICAL REPRESENTATION

In order to validate the transformation rules, we measure the quality of the generated OWL2 graphical representation through the recall and precision rates.

We recall that the precision is the ratio of real elements generated by our transformation that were identified by the expert. It indicates how accurate the transformation rules are in the generation of OWL2 graphical representation (class, data type property and object property) (see Formula 1).

The recall is the ratio that indicates the capacity of our transformations to return all elements specified by the expert. High scores for both ratios show that the transformations return both an accurate OWL2 graphical representation (high precision), and the majority of all relevant accurate OWL2 graphical representation elements (high recall). It means that the generated OWL2 graphical representation covers the whole domain precisely in accordance to the experts' perspective (see Formula 2).

To have the harmonic mean of recall and precision, we have used the F-measure. F-measure has a parameter that sets the trade-off between recall and precision. The standard F-measure is F1, which gives equal importance to recall and precision (see Formula 3). We calculate these rates according the following equations:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F1 = \frac{2 * Recall * Precision}{Recall + Precision} \quad (3)$$

Where:

- True positive (TP) is the number of existing real elements generated by our transformation;
- False Positive (FP) is the number of not existing real elements generated by our transformation;
- False Negative (FN) is the number of existing real elements not generated by our transformation.

4.2 Case Study

To illustrate the transformation rules, we use the BPMN model "Purchase department process" example as illustrated in Figure 14.

First, by applying **R3.1**, the *Purchase Department* pool and the *Supplier* pool are transformed to classes named "Purchase department management", and "Supplier Space". Each lane (*Agent, Manager*) in the *Purchase Department* pool is transformed to subclasses named "Agent space" and "Manger space" of the "Purchase department management" class. Second, we generate four classes by applying **R3.2**, which are "Purchase Department", "Agent" "Manager", and "Supplier".

The "Purchase Department" and "Agent" are subclasses of "Agent space". While the "Manager" and the supplier represent respectively subclasses of "Manager Space" and "Supplier Space" classes.

In addition, the description field of business object "Purchase Department", defined in its business context, indicates that the "Purchase Department contains many mangers and agents". This description is transformed into an object property between"Purchase Department" and "Manager" and "Agent" classes. Furthermore, the rule **R3.2** calls **R2**, which adds the data type property to all classes based on the business context of the corresponding pool and lanes. For example, we added the data type properties

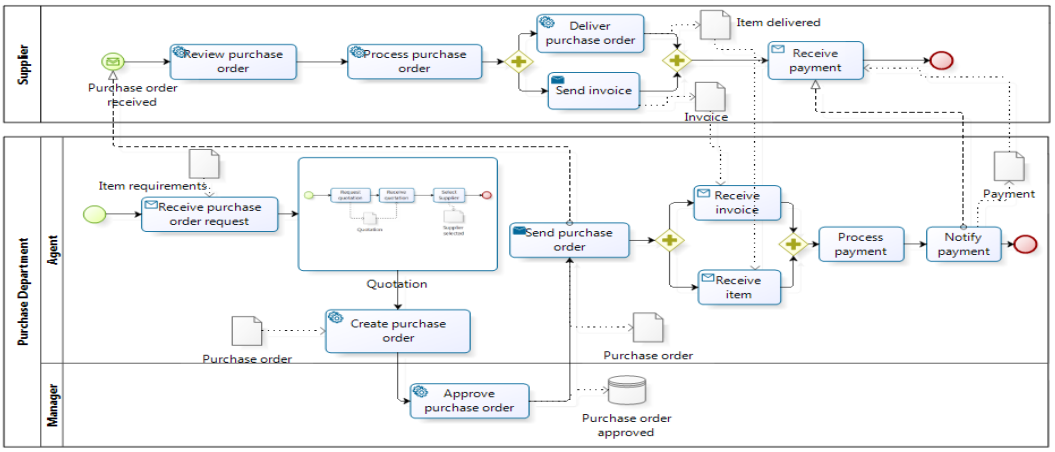

Figure 14: Purchase order Business Process model in BPMN (ISO/IEC 19510, 2013).

id, name and job to “Manager” and “Agent” classes, id and designation to “Purchase Department” class.

Third, we apply **R4.1** on the following service tasks: “Create purchase order”, “Approve purchase order”, “Deliver purchase order”, “Review purchase order”, “Process purchase order”, “Process payment”, “Notify payment”, “Request quotations”, and “Select supplier”. This rule generates three subclasses: “Purchase order”, “Payment”, and “Quotation”. The “Supplier” class is already generated by **R3.1**. Each action verb of service tasks presented above is transformed into an object property between the corresponding business object and performer classes. For instance: “Create” and “Process” represent an object property between the “Purchase order” and “Agent” classes; “Approve” denotes an object property between the “Purchase order” and “Manager” classes; “Deliver” and “Review” express an object property between the “Purchase order” and “Supplier” classes; “Process” and “Notify” denote an object property between the “Payment” and “Agent” classes; “Request” express an object property between the “Quotation” and “Agent” classes; “Select” represent an object property between the “Supplier” and “Agent” classes.

Afterward, by applying **R5.1**, the tasks “Send purchase order” and “Send invoice” generate an object property with the name “Send” between the “Agent” and “Supplier” classes.

In addition, we apply **R5.1** on receive tasks which are “Receive invoice” “Receive purchase order request”, “Receive payment”, “Receive item”, and “Receive quotation”. For instance: “Receive invoice”, “Receive payment” and “Receive item” express object properties with the name “Receive” between the “Supplier” and “Agent” classes; “Receive purchase order request” represent an object property with the name “Receive” between the “Agent” and “Customer” classes;

By applying R6, the transformation of all data objects do not add new classes. However, R6 enhances the existing classes by calling R1 and R2, which add data type properties, classes and object properties. For example, we added the data type properties deliveryDate, orderDate, orderNumber to “Purchase order” class since these extended attributes are pure values. Furthermore, the extended attribute “OrderLine” is a complex entity. According to R2, we extract a new class “OrderLine”, and an object property between the latter and “Purchase order”.

Figure 15 shows the generated OWL2 graphical representation To evaluate the quality of the generated OWL2 graphical representation, we calculate the recall and precision rates presented in section 4.1.

Precision=81/(81+2)=0.97

Recall =81/(81+5) =0.94

F1= (2*0.97*0.94) /(0.97+0.94)= 0.95

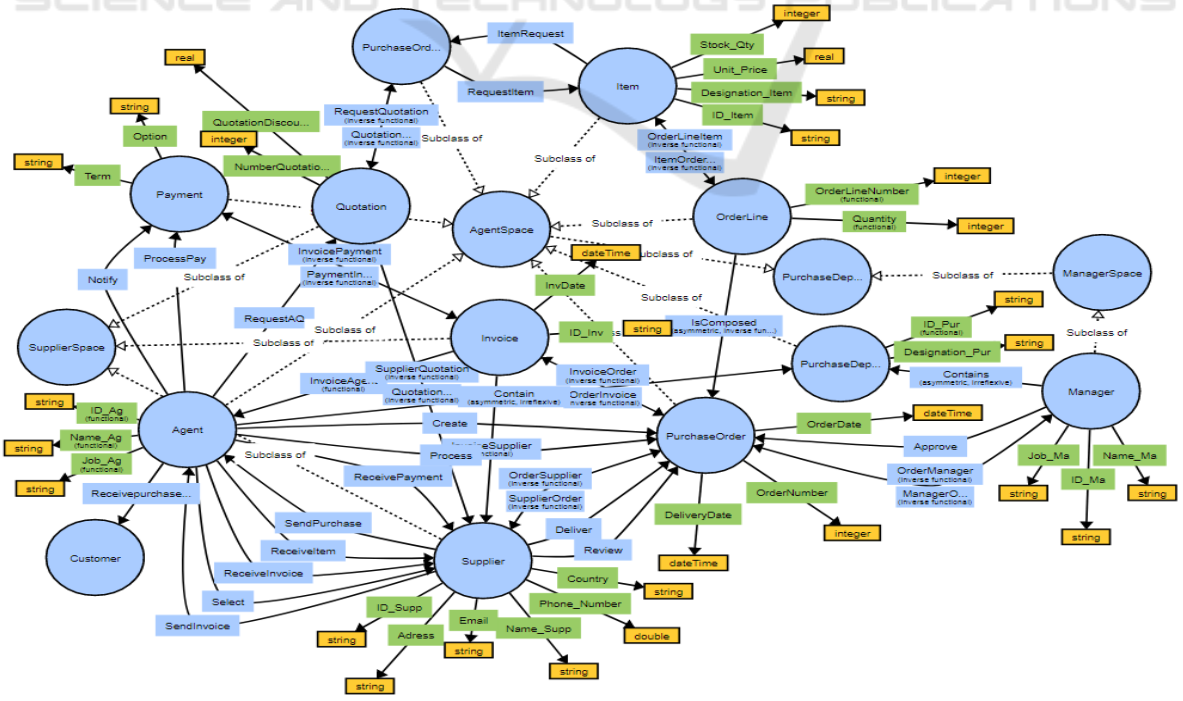

Figure 15: The generated OWL2 graphical representation for the purchase order process.

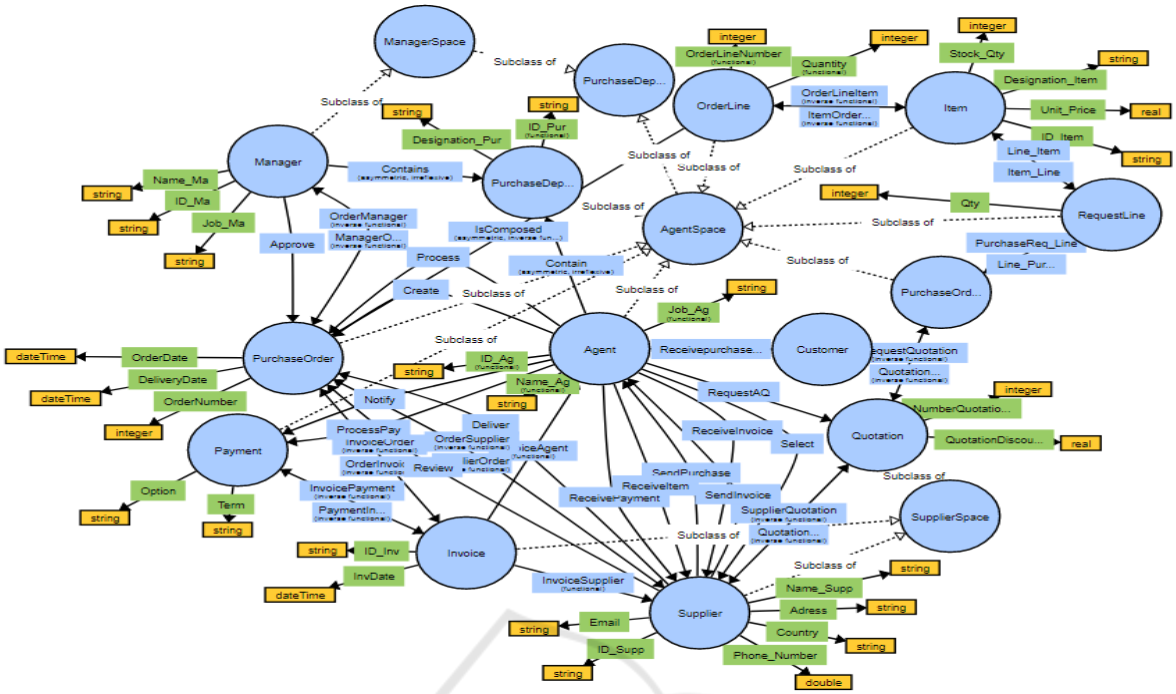

Figure 16: The elaborated OWL2 graphical representation by the expert.

The high values for both ratios mean that the generated OWL2 graphical representation covers the whole domain precisely in accordance with the experts' perspective (See Figure 16). We can deduce that the performance of our transformations approaches the human performance.

Our generated OWL2 graphical representation contains two object properties between the "Item" and "Purchase order request" classes. However, the OWL2 graphical representation generated by the expert contains: "Request Line" class, two object properties between "Request Line" and "Item" classes and two object properties between "Request Line" and "Purchase order request" classes.

5 CONCLUSION

This paper focuses on transforming the BPMN model into graphical representation of OWL2 and its generated code. Compared to existing works, we proposed a set of transformation rules that consider the semantic aspects of the business process model.

To do so, we are based on the business process context expressing the semantics relation and type.

Our future work focuses on two main axes: 1) enhancing the transformations in order to cover use case and sequence diagrams. 2) developing a tool for the automatic generation of an ontology from a business process model.

REFERENCES

Annane, A., Aussenac-Gilles, N., Kamel, M., 2019. BBO: BPMN 2.0 Based Ontology for Business Process Representation. In *ECKM'19, 20th European Conference on Knowledge Management*. pp. 49-59.

BPMN-onto, 2019. Available at https://dkm.fbk.eu/bpmn-ontology.

Figueiredo, L.R., Oliveira, H.C., 2018. Automatic Generation of Ontologies from Business Process Models. In *Inter. Conf. on Ent. Information Systems*.

ISO/IEC 19510. 2013. *Information technology -- Object Management Group Bus. Proc. Model and Notation*.

Khlif, W., Elleuch, N., Alotabi, E., Ben-Abdallah, H., 2018.Designing BP-IS Aligned Models: An MDA-based Transformation Methodology. In *13th Inter. Conf. on Evaluation of Novel Approaches to Software Engineering*, pp. 258-266.

Noy, N., McGuinness, D., 2001. *Ontology Development 101: A Guide to Creating Your First Ontology*.

Ternai, K., Török, M. and Varga, K., 2016. *Corporate Semantic Business Process Management,* Corporate Knowledge Discovery and Organizational Learning: The Role, Importance, and Application of Semantic Business Process Management, pp. 33- 57.

W3C, 2005. A survey of RDF/Topic Maps Interoperability Proposals W3C Working Draft.