An OWL Implementation of OntoUML and BPMN Models to Unify Representation of Structure and Behavior of Complex Domains: Application to Routing Protocols

Mohamed Bettaz[a]
Faculty of Information Technology, Czech Technical University in Prague, Thakurova 9, Prague, Czech Republic

Keywords: Ontology, UFO, gUFO, OntoUML, BPMN, OWL, Dynamic Routing Protocols, Turtle, SPARQL.

Abstract: The objective of this paper is twofold. First, we propose an approach to map BMPN to OWL, and then we use OntoUML and BPMN (in their ontological form) to demonstrate the effectiveness of our approach through its application to a complex and irregular problem domain, namely dynamic routing protocols. This allows us to query their structural and dynamic aspects (and reason about them) in a uniform and transparent manner. It should be noted that for sake of readability, the models representing parts of routing protocols are intentionally kept simple, emphasizing key concepts and their relationships.

1 INTRODUCTION

OntoUML (Barcelos et al., 2013) is a conceptual ontology language used for the specification of structural aspects of problem domains. Business Process Modeling Notation (BPMN) (Silver, 2011) is a visual language used for modeling process aspects of problem domains. The web ontology language (OWL) (W3C, 2025a) is a computational and knowledge representation ontology language provided with a formal semantics. A transformation of OntoUML into OWL was proposed in (Barcelos et al., 2013). In this contribution we propose an approach to transform BPMN into OWL. The idea behind such a goal is that mapping OntoUML and BPMN into a "universal" knowledge representation ontology language allows us to build knowledge-based systems for complex domains that are query-able and lending themselves to uniform reasoning. Moreover, specifications from models representing different system structures and behaviors and which are also expressed in different languages become interoperable. In a recent contribution (Bettaz and Maouche, 2025), we showed how to use an association of OntoUML and SoaML (Service-oriented architecture Modeling Language) to specify IoT systems. OntoUML was used for the specification of the structure, while SoaML was used for the specification of the behavior of these systems. In this work we envisage to use BPMN (instead of SoaML) for its readability, flexibility and visual appeal. The objective of this paper is twofold. First, we propose an approach to map BMPN to OWL, and then we use OntoUML and BPMN (in their ontological form) to demonstrate the effectiveness of our approach through its application to a complex and irregular problem domain, namely dynamic routing protocols. The OWL ontology resulting from the models built using OntoUML and BPMN is implemented in Turtle and queried on a Fuseki (SPARQL) server. The implementation of this ontology can serve as a knowledge-based system that can be used to reason on dynamic routing protocols (cf.section 8)

The rest of the paper is organized as follows. Section 2 summarizes basic knowledge on BPMN, RDF (Resource Description Framework), OntoUML and routing protocols. This section aims at making the paper as self-constrained as possible; parts or all of this section can be skipped by the reader who is introduced to the topics covered in this section. In section 3, we present the used research method and formulate our research hypotheses. Section 4 presents related work. In section 5, we present and motivate the approach we use to map BPMN to OWL. Section 6 presents a case study (acting as a proof of concept) of the approach described in section 5. An implementation of the ontology resulting from the models built in our case study is given in section 7. Some concluding remarks and future work are then outlined.

[a] https://orcid.org/0000-0003-1346-0244

Bettaz, M.
An OWL Implementation of OntoUML and BPMN Models to Unify Representation of Structure and Behavior of Complex Domains: Application to Routing Protocols.
DOI: 10.5220/0013504800003970
In *Proceedings of the 15th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2025)*, pages 254-261
ISBN: 978-989-758-759-7; ISSN: 2184-2841
Copyright © 2025 by Paper published under CC license (CC BY-NC-ND 4.0)

2 BACKGROUND

2.1 BPMN

BPMN is used for the modeling of process aspects of problem domains. It comprises the following modeling constructs: activities, events and gateways. An activity can typically be atomic, in which case it is called a task, or decomposable, in which case it is called a subprocess (Author, 2024). Events fall into three main types: start, intermediary, and end. From these three basic types we can form various sub-types. Gateways are used to control the flow in BPMN processes, which means that they deal with branching in the processes. In a BPMN process, we typically encounter the gateways exclusive, inclusive, parallel, event-based, exclusive event-based and parallel event-based. For more details the kind reader can consult (Silver, 2011).

2.2 Resource Description Framework

RDF (W3C, 2025b), which is the main building block of OWL, is a data model designed by the World Wide Web Consortium (W3C) for describing resources using semantic triples (subject, predicate, object). A resource is anything that can be uniquely identified by an IRI (Internationalized Resource Identifier), which is a URL-like object identifier consisting of a name space and a local identifier. RDFS (Resource Description Framework Schema)(W3C, 2025) adds structure to RDF by distinguishing resource types and defining their properties, thus forming a simple ontological language including a taxonomy of individuals. OWL is a family of languages (OWL Lite, OWL DL and OWL Full), that allow to specify not only types and relationships among them but also constraints such as multiplicities and generalization sets among others. For a description of the full set of OWL language constructs, the reader may consult (W3C, 2025a)

2.3 OntoUML

OntoUML is a UML profile using stereotypes defined by the Unified Foundational Ontology (UFO) (Guizzardi et al., 2021). An OntoUML model consists of (shared) concepts and relationships used to describe structural aspects of problem domains. In OntoUML both concepts (UML classes) and relationships (UML associations) are stereotyped. In OntoUML models, stereotypes are enclosed in double angle brackets. According to the stereotypes, types are either sortals or non-sortals. Instances of sortal types have one principle of identity. Instances of non-sortal types can have different principles of identity. Sortal and non-sortal types can be either rigid or anti-rigid. Many notions of UFO and OntoUML are formalized. For more details on UFO and OntoUML, the kind reader can consult for instance (Guizzardi et al., 2021) and related literature.

2.4 Routing Protocols

Communication protocols in general and routing protocols in particular are considered as a complex domain. Using more than one language to specify their various facets is not new in the discipline. Indeed, from the very beginning of the 1990's, specialists of the domain tried to associate different specification languages to cope with their various facets. As examples we mention Lotos (Turner, 1993), an association of abstract data types and process calculus; Estelle (Turner, 1993), an association of Pascal and finite state automata; ECATnets (Bettaz et al., 1992), an association of rewriting logic and abstract data types.
In this paper we focus on (centralized) dynamic routing protocols, where contrary to static routing protocols, the elaboration of the forwarding tables is the responsibility of the routers. The entity responsible for elaborating these tables is called control plane. The tables elaborated by the control plane are handed to the data plane, which is the entity responsible for using these tables to forward data packets.

3 METHOD AND HYPOTHESIS

In this work we use a qualitative research method aiming at providing BPMN with a declarative semantics based on the use of RDF semantic triples. Details on the used approach are given at the very beginning of section 5. The proposed method is supported by a case study acting as a proof of concept (Hustadt, 2024). Our first research hypothesis can be stated as follows. While some research works tend to define the semantics of BPMN through its mapping to other languages (such as Petri nets, abstract state machines or activity diagrams), we propose to use RDF, a declarative language that is also used for knowledge representation and reasoning (KR^2) (Salihoglu, 2024). Our second research hypothesis can be stated as follows. While many research works use different languages to specify various aspects of problem domains, we propose to use languages that can be unified through their mapping to a "universal" knowledge representation ontology language.

4 RELATED WORK

In (de Brock, 2024), the author elaborates a declarative semantics for BPMN, based on "modeling the semantics of (individual) actions as state transitions represented as states".

In (Kchaou et al., 2021), the authors propose rules to transform BPMN models into OWL2 graphical representation. Dixit: "The transformation rules are based on an annotated BPMN model to generate an aligned OWL2."

In (Suchánek and Pergl, 2021), the authors propose an approach that allows to create a BORM ontology by showing how to represent the knowledge carried by its process models in RDF. For the conceptualization using RDF, the work in (Kchaou et al., 2021) and the work in (Suchánek and Pergl, 2021) are similar as regards to the transformation into OWL.

Compared to (de Brock, 2024), (Kchaou et al., 2021), and (Suchánek and Pergl, 2021), our transformation approach from BPMN to RDF uses a declarative approach based directly on the definition of an ontology in terms of concepts that are abstractions of objects (activities, events and gateways), and relationships that are representations of lines with arrows connecting these objects.

As regards to the building of ontologies for dynamic routing protocols, there are no contributions, to the best of our knowledge, using associations of OntoUML and BPMN. Same observation concerning the uniform conceptualization of both OntoUML and BPMN using RDF semantic triples.

5 MAPPING BPMN TO OWL

Our mapping approach is based on a simple but effective observation, that from an ontological point of view, a BPMN model can be perceived as a set of concepts and relationships, where concepts are materialized by BPMN objects (i.e., activities, events, and gateways), while relationships are formed by lines with arrows linking these objects. Expressing our concepts and relationships in the form of RDF triples provides our BPMN models with declarative semantics. In the following conceptualization of objects (in their various forms) and relationships, only samples of triples are given for illustration as suggested by some of the reviewers. A full specification can be provided (on demand) by the author.

5.1 Conceptualization of BPMN Tasks and Activities

The concept task forms a type, the (high-level) instances of which are themselves types. These instances are called manual, user, service, and script; their (low-level) instances (or individuals) are units of work called activities. Indeed, the icons used to denote a manual, user, service or script activity, figuring on the graphical representations of BPMN activities are sorts of stereotypes leveraging the semantics of such activities. Such stereotypes (regrouping individuals of the same sort) will be considered as OWL classes. The RDF triples will thus take the following form. A colon preceded by a blank means that we do not worry about the IRIs for the moment; they will be defined in section 7. Examples of description of BMPN tasks by semantic triples are:
:manualTask rdf:type owl:Class
:activity rdf:type :manualTask
:activity rdf:type :userTask
:activity rdf:type :serviceTask
:activity rdf:type :scriptTask
:manualTask rdfs:subClassOf :task
...
Subprocesses are classes of classes: ad-hoc subprocesses, loop subprocesses, multi-instance subprocesses, compensation subprocesses, compensation and ad-hoc subprocesses. Such stereotypes will also be considered as OWL classes. Their instances are called structured-activities. Our semantic triples take the following form.
:adHocSubprocess rdf:type owl:Class
:structured-activity rdf:type :adHocSubprocess
...

5.2 Conceptualization of BPMN Events

BPMN start, intermediate and end events are seen as different types of events; start event is in turn seen as a subclass of catching event and also as a subclass of non interrupting event; intermediate event is seen a subclass of catching event, as a subclass of throwing event and as a subclass of non interrupting event; end event is seen as a subclass of throwing event.
Examples of semantic triples describing BPMN event modeling constructs are:
:event rdf:type :startEvent
:startEvent rdfs:subClassOf :catching
...

5.3 Conceptualization of BPMN Gateways

The concept gateway is a type with six instances, named exclusive, parallel, inclusive, event based, exclusive event based, and parallel event based (cf. section 2.1). Examples of semantic triples used to specify BPMN gateways are:
:exclusive rdf:type :gateway
:parallel rdf:type :gateway
:inclusive rdf:type :gateway
...

5.4 Conceptualization of BPMN Relationships

Relationships are morphisms (abstractions of BPMN lines with arrows) the domain and range of which are types representing concepts (abstractions of various BPMN modeling constructs such as tasks, events, and gateways). BPMN lines with arrows will be represented by a semantic triple as follows.

:< *arrow_name* > rdf:type owl:ObjectProperty
rdf:domain :< *type_name* >
rdf:range :< *type_name* >

6 CASE STUDY: DYNAMIC ROUTING PROTOCOLS

We begin with the specification of structural aspects and then proceed to the specification of process aspects, focusing on key concepts and relationships. Unless otherwise stated, we will henceforth use the term routing protocols to refer to dynamic routing protocols.

6.1 Specification of the Structures

Structural aspects are specified by OntoUML models that describe key concepts of the dynamic routing protocols' domain, such as static and dynamic routing, autonomous systems, centralized vs decentralized algorithms, and routing tables. For lack of space we will just specify the model depicted by figure 1. From the specification of this model, it should be clear how to specify the models depicted by the other figures. Figure 1 should be read as follows. IP routing protocols (<< *Category* >> IP Routing) are either static (<< *Kind* >> Static) or dynamic (<< *Kind* >> Dynamic). Static routing protocols are characterized by the creation of routing tables at boot time (<< *Mode* >> Routing Table Created at Boot Time). Dynamic routing protocols are characterized by the use of complex routing algorithms (<< *Mode* >> Complex Routing Algorithms). In OntoUML, the stereotype Mode is used with special classes to emphasize significant entity properties, hence the association of the stereotype Characterization (<< *Characterization* >>) with the relationship between the given entity and its emphasized property. Dynamic routing algorithms (<< *Mode* >> Complex Routing Algorithms) are used to compute an optimal route (<< *Relator* >> Optimal Route Determinator) among the set of all possible routes. To this end, our dynamic routing algorithms should advertise (<< *Role* >> Routing Information Adviser) and learn (<< *Role* >> Routing Information Learner) routing information about IP subnets to and from neighbouring routers. An optimal route is determined using a given metric (<< *Quality* >> Metric) (cf. (Kurose and Ross, 2021) and (Peterson and Davie, 2021)). As for the stereotype Mode, it is worth mentioning that classes with the stereotype Quality are also special classes used to emphasize significant entity properties. The difference between << *Mode* >> and << *Quality* >> is that the second is measurable, while the first is not.

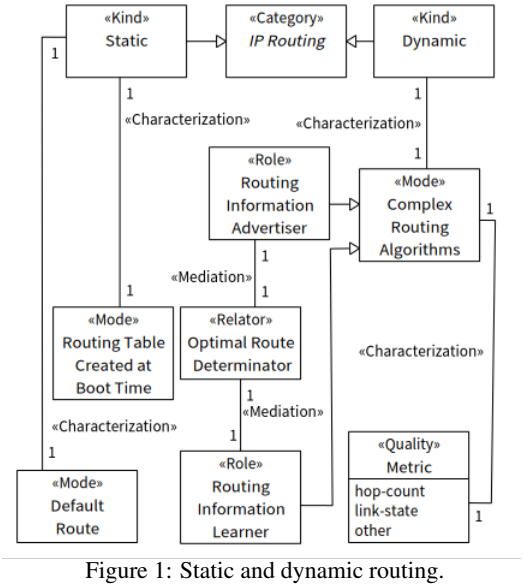

Figure 1: Static and dynamic routing.

Note: The OCL (Open Constraint Language) predicate (OMG, 2025) in the model depicted by figure 4 states that a router is not a neighbour of itself.

6.2 Specification of the Behaviors

In (Bettaz and Maouche, 2025), we utilise SoaML (a language used for specifying cyber-physical systems) for the specification of process aspects of IoT sys-

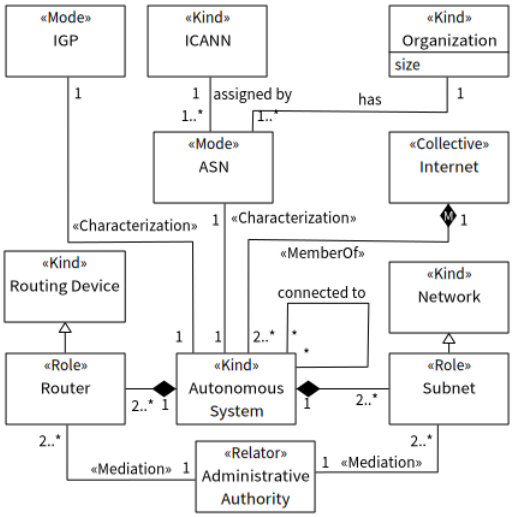

Figure 2: Autonomous systems (AS).

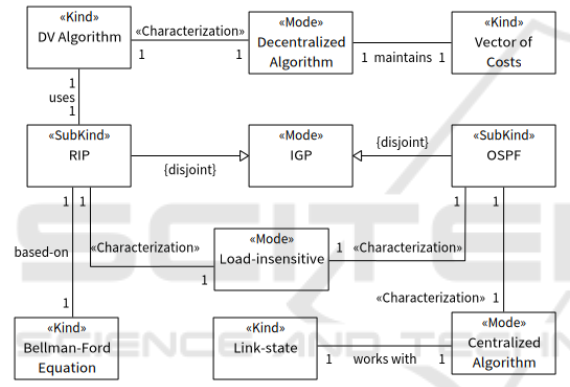

Figure 3: Centralized vs decentralized algorithms.

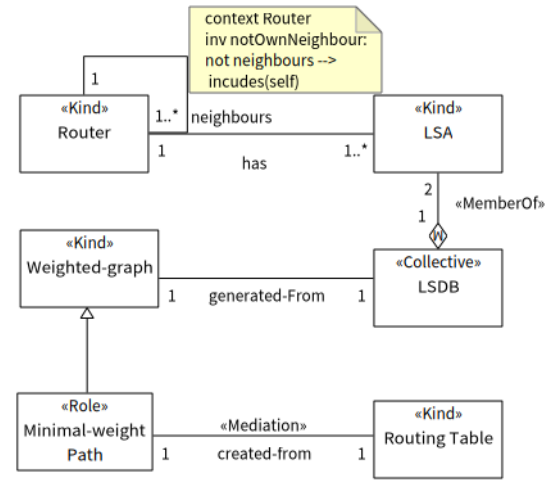

Figure 4: Generation of routing tables.

tems. In this contribution we utilise BPMN models for the specification of process aspects of routing protocols. These models are depicted in figures 5, 6, 7, 8, 9, and 10. Figure 5 depicts a scenario with three participants (routers R1, R2, R3). In this scenario R2 is supposed to be the router starting the process by advertising available routing information to both R1 and R3. R1 is supposed to be the router ending the process after updating its routing table and advertising updated routing information.

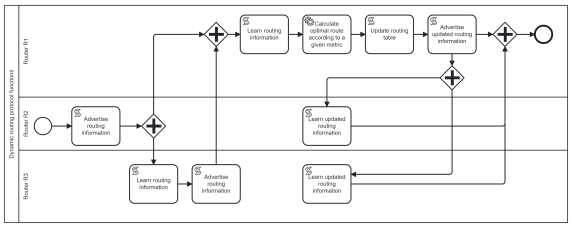

Figure 5: Basics from routing protocols.

Figure 6: Protocol service.

The model depicted in figure 6, describes the protocol service; its interpretation follows in a straightforward way from the the interpretation of the used BPMN constructs and their (sequential) composition.

The chosen routing protocol scenario supposes a network consisting of four routers (R1, R2, R3, R4) interconnected according to the following topology. R1 has two adjacencies: R2 and R3. R2 has three adjacencies: R1, R3, and R4. R3 has three adjacencies: R1, R3, and R4. R4 has two adjacencies: R2 and R3. This scenario describes how R4 proceeds to build its routing table (cf. Figure 7). Once R4 receives all the LSAs from all the routers, it proceeds by building its LSDB and its weighted graph. Then the minimal-weight graph algorithm is executed and the routing table created.

6.3 Conceptualization of the Structures

For illustration, we consider the OntoUML model depicted in figure 1. This model describes notions of static and dynamic routing. The conceptualization is based on gUFO, a lightweight implementation of UFO in OWL (Almeida et al., 2020). In the following we just write down a semantic triple representing a class and a semantic triple representing a relationship (cf. figure 1). From these, it should be clear how to write down semantic triples for the other classes and relationships.

example class) class Static
:Static rdf:type owl:Class ;
rdfs:subClassOf gufo:Object ;

rdfs:subClassOf :IP_Routing ;
rdf:type gufo:Kind ;
rdfs:subClassOf owl:Restriction ;
owl:onProperty :characterizes;
owl:qualifiedCardinality 1;
owl:onClass :Default_Route.
. . .

example relationship) relationship "characterizes" (between class Static and Class Default_Route)
:charaterizes rdf:type owl:ObjectProperty ;
rdf:type gufo: Characterization ;
rdfs:domain :Static ;
rdfs:range :Default_Route .

. . .

6.4 Conceptualization of the Behaviors

We consider the BPMN model of the protocol depicted in figure 7 and present samples illustrating the conceptualization of few concepts (tasks/processes, events and gateways) and relationships (lines with arrows).

6.4.1 Subprocesses and Activities

We just give the semantic triples representing subprocess A and activity buildLSDB. From these, it should be clear how to write the semantic triples for the other subprocesses and activities.
(a1) subprocess A
:A rdf:type :adHocSubprocess .
:adHocSubprocess rdf:type owl:Class ;
rdfs:subClassOf :subProcess .
. . .
(a4) activity buildLSDB
:buildLSDB rdf:type :scriptTask .
:scriptTask rdf:type owl:Class ;
rdfs:subClassOf :task .
. . .

6.4.2 Events

Figure 7 shows three events (a start event, an intermediate event, and an end event).
(b1) start event s1
:s1 rdf:type :startEvent .
:startEvent rdfs:subClassOf :event .
(b2) intermediate event R4gotAllTheLSAs
:R4gotAllTheLSAs rdf:type :intermediateEvent ;
rdf:type :message ;
rdfs:subClassOf :catching .
:intermediateEvent rdfs:subClassOf :event .
(b3) end event t1

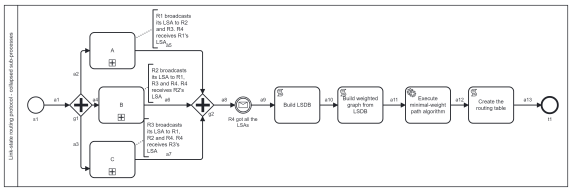

Figure 7: Routing protocol model with collapsed subprocesses.

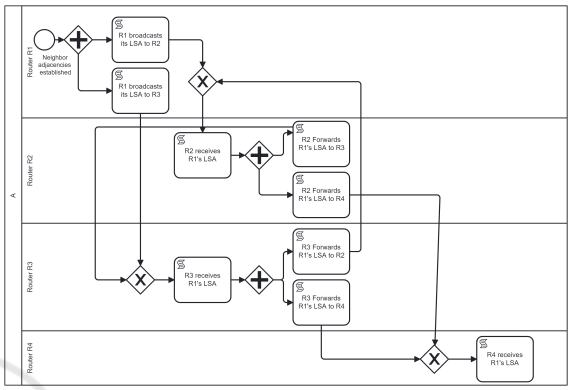

Figure 8: Subprocess A - expanded.

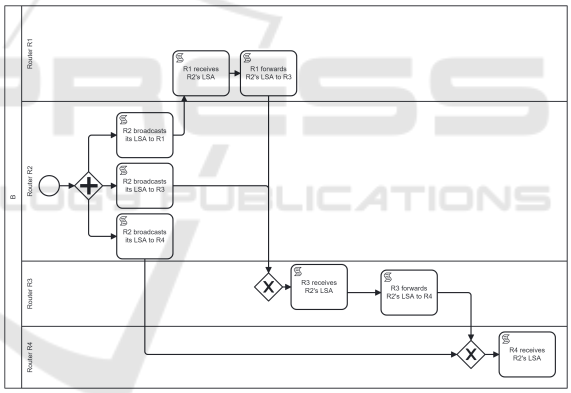

Figure 9: Subprocess B - expanded.

:t1 rdf:type :endEvent .
:endEvent rdfs:subClassOf :event .

6.4.3 Gateways

Figure 7 shows two gateways (both of them are parallel).
(c1) parallel gateway g1
:g1 rdf:type :parallelGateway .
:parallelGateway rdfs:subClassOf :gateway .
(c2) parallel gateway g2
:g2 rdf:type :parallelGateway .
:parallelGateway rdfs:subClassOf :gateway .

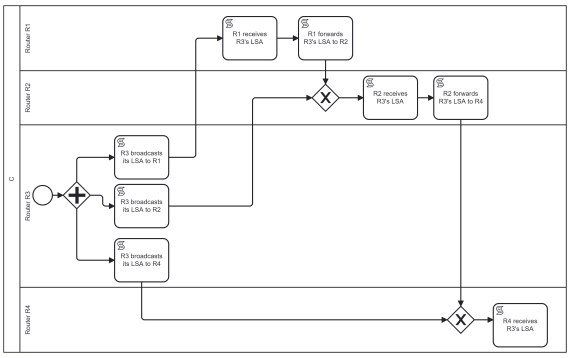

Figure 10: Subprocess C - expanded.

6.4.4 Relationships

Figure 7 shows thirteen lines with arrows (a1, a2, ..., a13). In the following we write down the triples for only two of them. From these, writing the triples for the remaining lines should be clear.

arrow a1

:(s1, g1) rdf:type owl:ObjectProperty ;
rdf:domain :startEvent ;
rdf:range : parallelGateway .

arrow a2

:(g1, A) rdf:type owl:ObjectProperty ;
rdf:domain :parallelGateway ;
rdf:range : adHocSubprocess .

...

7 IMPLEMENTATION

The ontology is implemented with Turtle and queried on a Fuseki (SPARQL) server running on a DELL computer (Intel Core i7, 16.0 GB RAM). The code can be provided (on demand) by the author. For lack of space, we just consider the implementation of the behaviors.

The code is saved in a file called fig7.ttl, containing the description of the prefixes, followed by the definition of the concepts and the relationships according to Turtle syntax.

A sample of the implementation results is presented by the screenshot depicted in figure 11, and a sample of query corresponding to these results is presented by the screenshot depicted in figure 12. Figure 11 shows the name of the server (Apache Jena Fuseki), the name of the uploaded Turtle file (Fig7.ttl), the size of this file (4.64kb), and the number of uploaded triples (74) that effectively corresponds to the number of semantic triples saved in the file. The status (100.00 in green) states that the implementation ran without errors. In turn figure 12 shows the prefixes

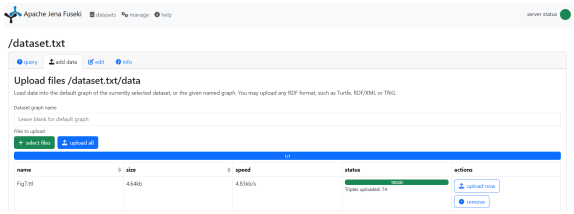

Figure 11: Routing protocol implementation.

Figure 12: Routing protocol implementation - sample query.

used in the query and the query itself. Downloading the results of the query from the server shows the following.

processName
http://uri/subprocess#A
http://uri/subprocess#B
http://uri/subprocess#C

Indeed, the routing protocol depicted in figure 7 shows effectively that we have three ad hoc subprocesses, namely A, B and C. This result can be exploited when implementing the protocol in a programming language by launching the execution of the three subprocesses in parallel.

8 CONCLUSION AND FUTURE WORK

The approach we proposed to map BPMN to OWL is a first result of our contribution. Indeed, mapping BPMN to a computational and knowledge representation ontology language allows to unify problem domain representations expressed in different languages. Moreover, the implementation of ontologies resulting from such representations can serve as a knowledge-based system to reason about dynamic routing protocols. Indeed, from our specifications we can infer new sentences in form of triples. For instance, from the specification presented in section 6.4.2, it should be clear that we can infer the following (and many other) triple(s) stating that R4gotAllTheLSAs is a catching event in our protocol specification.

:R4gotAllTheLSAs rdf:type :catching.

The effectiveness of our approach has been demonstrated by its application to a complex problem domain, namely dynamic routing protocols. Our models were (partially) implemented in Turtle and queried on a Fuseki (SPARQL) server. Examples of our implementation and associated queries are presented as screenshots. In addition to this, we have shown how to use OntoUML to build ontology fragments for the structural part of dynamic routing protocols, and BPMN (in its OWL form) to build an ontology for the process part of this kind of protocols. This can be considered as a second result of our contribution. From the problem domain perspective, our ontology can serve as an enrichment of the domain knowledge on which routing protocols are based. It is known that to develop an application/system for a domain, one must have (or collaborate with people who have) some expertise (knowledge) of the domain.

We plan in the near future to formalize some aspects of what is presented in this work. We plan to exploit two ideas. The first idea consists in expressing the protocol service as a many-sorted algebra, and expressing the implementation of the service (i.e., the protocol itself) as a many-sorted algebra. The objective is to show that both algebras are isomorphic, thus proving formally that the protocol implements its service correctly. The second idea consists in using rewriting logic (Diaconescu, 2025) to show that the semantic triples used to specify the protocol could be derived from the semantic triples used to specify the protocol service. This is an operational approach permitting to show that the protocol implements correctly its service. The other benefit of this idea is to "visualize" the parallelism inherent to the protocol by exploiting the concurrent computing permitted by rewriting logic. This could be efficiently applied for instance to simulate the parallelism exhibited by figures 7, 8, 9, and 10.

ACKNOWLEDGEMENTS

The author thanks the five anonymous reviewers for their valuable comments that helped improve the final version of this article.

REFERENCES

Almeida, J. P. A., Guizzardi, G., de Almeida Falbo, R., and Sales, T. P. (2020). gUFO: A Lightweight Implementation of the Unified Foundational Ontology (UFO). https://www.researchgate.net/publication/339747587_gUFO_A_Lightweight_Implementation_of_the_Unified_Foundational_Ontology_UFO.

Author, A. (2024). BPMN Activity Types Explained. https://www.visual-paradigm.com/guide/bpmn/bpmn-activity-types-explained/.

Barcelos, P. P. F., dos Santos, V. A., Silva, F. B., Monteiro, M. E., and Garcia, A. S. (2013). An Automated Transformation from OntoUML to OWL and SWRL. pages 130–141.

Bettaz, M. and Maouche, M. (2025). Towards an Ontological-based CIM Modeling Framework for IoT Applications. *Informatica An International Journal for Computing and Informatics*, 48(4):663–684.

Bettaz, M., Maouche, M., Soualmi, M., and Boukebeche, M. (1992). Using ECATNets for Specifying Communication Software in the OSI Framework. In *1992 Fourth International Conference on Computing and Information*, pages 410–413, USA. IEEE Computer Society.

de Brock, B. (2024). Assigning Declarative Semantics to Some UML Activity Diagrams and BPMN Diagrams. In Shishkov, B., editor, *Business Modeling and Software Design*, pages 65–82, Cham. Springer Nature Switzerland.

Diaconescu, R. (2025). *Institution-independent Model Theory*. Springer, 2nd edition.

Guizzardi, G., Benevides, A., Fonseca, C., Porello, D., Paulo, J., Almeida, A., and Sales, T. (2021). UFO: Unified Foundational Ontology. In *Applied Ontology 1 (2021)*. IOS PRESS.

Hustadt, U. (2024). Research methods in computer science. https://cgi.csc.liv.ac.uk/ ullrich/COMP516/notes/lect06.pdf.

Kchaou, M., Khlif, W., Gargouri, F., and Mahfoudh, M. (2021). Transformation of bpmn model into an owl2 ontology. In *ENASE*, pages 380–388. INSTICC, SciTePress.

Kurose, F. and Ross, K. (2021). *Computer Networking - A Top-Down Approach*. Pearson, 8th edition.

OMG (2025). Object constraint language.

Peterson, L. L. and Davie, B. S. (2021). *Networks - A Systems Approach*. MK, 6th edition.

Salihoglu, S. (2024). In praise of RDF. https://blog.kuzudb.com/post/in-praise-of-rdf/.

Silver, B. (2011). *BPMN Method & Style*. Cody-Cassidy Press, 2nd edition.

Suchánek, M. and Pergl, R. (2021). Representing BORM Process Models using OWL and RDF. In *KEOD*, volume 2, pages 170–177. SciTePress.

Turner, K. J. (1993). *Using Formal Description Techniques: An Introduction to ESTELLE, LOTOS and SDL (Wiley Series in Communication and Distributed Systems)*. Wiley, 1st edition.

W3C (2025). RDF Schema 1.1. https://www.w3.org/TR/rdf-schema/.

W3C (2025a). OWL Web Ontology Language Reference. https://www.w3.org/TR/owl-ref/.

W3C (2025b). RDF 1.2 Concepts and Abstract Syntax. https://www.w3.org/TR/rdf12-concepts/.