

Representing BORM Process Models using OWL and RDF

Marek Suchánek^{1,2}^a and Robert Pergl¹^b

¹*Faculty of Information Technology, Czech Technical University in Prague,
Thákurova 9, Prague, Czech Republic*

²*Faculty of Business and Economics, University of Antwerp,
Prinsstraat 13, Antwerp, Belgium*

Keywords: Ontology, BORM, Business Process Modelling, RDF, OWL, Transformation.

Abstract: Business Object Relationship Modeling (BORM) is a business process analysis method based on communicating finite-state machines and Petri nets. However, because of its tooling support and non-interoperability of formats, it is rather niche. This work proposes a way of representing the knowledge from BORM process models in RDF by creating a BORM ontology. It benefits from previous work done on different conceptual and process modelling languages and their transformations to OWL. The resulting RDF representation brings increased interoperability and enhanced analysis possibilities, e.g., using SPARQL or RDF visualization tools. A part of this work is also a BORM-to-RDF export feature for the OpenPonk modelling platform. The resulting BORM ontology is ready for use in practice and further work.

1 INTRODUCTION


Business process modelling is an activity of describing processes for documentation, optimization, or automation purposes (Scholz-Reiter and Stickel, 2012). In terms of process engineering as part of conceptual modelling, there are various languages and methods to build a process model, e.g. BPMN, UML, LML, or BORM. Business Objects Relation Modelling (BORM) (Merunka et al., 2009) uses an object-oriented approach through simplistic notation, focusing on connecting business and software engineering. It is built on formal foundations of communicating finite-state machine. However, it turned not to be enough for this method to be successful.


The usability of a modelling language or method is directly related to the tooling support and interoperability to help users use it (Scholz-Reiter and Stickel, 2012). Despite several substantial advantages, BORM is becoming unknown as an artefact of the past. Several tools allow BORM modelling; however, the used formats do not allow interoperability and are blockers in further use (e.g. orchestration, semantic integration, or normalization). Various tools for modelling or simulation of BORM models use specific formats that do not interchange knowledge.

Technologies of semantic web and linked data are well-known for their interoperability, versatility, and machine-actionability. RDF can be used to capture practically any knowledge with the possibility to use multiple formats to store and exchange it (for instance, Turtle or RDF/XML). RDF schema or OWL ontology can be used to give the data in RDF meaning. Last but not least, SPARQL can serve to query and manipulate RDF data in a standard and universal way. (Powers, 2007; Breitman et al., 2010)

In this work, we aim to allow capturing BORM models with the use of RDF and OWL. Giving a new interoperable way of knowledge representation and way of modelling has the potential to revive the method. By using RDF and OWL, the formal foundations and critical features of BORM must stay original. It must support the transformation from and to representations used by the existing modelling tools.

In the Section 2, we briefly explain the BORM, related tools, and necessary background in terms of RDF. Section 3 describes the first step, building an ontology-based on BORM metamodel. Then, Section 4 shows how the ontology is used for representing BORM models in RDF. Section 5 demonstrates the advantages of RDF representation in several use cases. Section 6 provides a discussion and evaluation of the results together with an outline of possible future steps.

^a  <https://orcid.org/0000-0001-7525-9218>

^b  <https://orcid.org/0000-0003-2980-4400>

2 RELATED WORK

This section serves as a brief overview of the BORM modelling and important terminology needed for work with RDF and OWL. It provides references to relevant previous work and details.

2.1 BORM Method

Business Objects Relation Modelling (BORM) (as summarized in (Merunka et al., 2009) and (Molhanec et al., 2011)) is a process modelling method for capturing knowledge of typical business systems. Its development started in 1993 and turned to be an effective due to its understandability and formal foundations. It is possible to apply formalism for validation and simulation (Podloucký and Pergl, 2014). The relation between BORM and other process modelling languages (BPMN and UML Activity Diagram) is described (Suchánek and Pergl, 2019).

We focus on Object Relations (OR) and Business Architecture (BA) diagrams. OR captures the process using communicating finite-state machines. Each machine represents a role of particular subject, i.e., the flow in process for a given participant. Through communication it composes a process with multiple roles. BA diagram provides an overview of the whole business and shows how processes are used in scenarios and related to business functions. (Merunka et al., 2009; Knott et al., 2000)

2.2 Craft.CASE

Craft.CASE (CRAFT.CASE Ltd., 2015) is an enterprise tool for modelling with BORM. As also described in (Merunka et al., 2009) and (Podloucký and Pergl, 2014), it supports the entire BORM method and not just the modelling part. It guides the analyst through the various stages of business analysis. The export and import are done through custom XML serialization of a whole project using `.crx` file extension. However, it also offers partial exports to CSV and JSON that allows simpler processing. As it is an enterprise and proprietary tool, the feature set is limited in free version. The development of the tool does not seem to be active (last update in 2015).

2.3 OpenCASE

OpenCABE (formerly OpenCASE) (Pergl and Tuma, 2012) is a BORM modelling tool based on the Eclipse Modeling Framework (EMF). It focuses primarily on the Object Relations and Business Architecture diagrams and their relations. Still, it also offers print-

able HTML reports that can be used in process operations as a scenario or decision helper. Furthermore, a project can be exported and imported through XMI format that uses a custom profile. Unfortunately, the development of this tool is discontinued, and OpenPonk can be seen as its successor.

2.4 OpenPonk

OpenPonk modelling platform (Uhnák and Pergl, 2016) is a metamodeling platform and a modelling tool implemented in Pharo (Smalltalk). The core feature of OpenPonk is to support various metamodels and use them for modelling purposes. There are variants of OpenPonk with different metamodels: UML Class Diagram, OntoUML, Petri Nets, Finite State Machines, and BORM ORD. The open-source project is published on GitHub¹ and actively developed (last release for BORM in March 2021). It has also proven its usefulness in terms of modularity in live visual modelling (Bliznicenko et al., 2017).

The serialization of models for import and export in OpenPonk is done as a ZIP archive containing several JSON files and Smalltalk Object Notation (STON) file(s). It directly serializes objects representing the model in Smalltalk. Another option is to export the diagram as a PNG picture. However, OpenPonk is designed to be extended including custom export and import functionality.

2.5 RDF and OWL

The ontologies and knowledge representation in information technologies are tightly related to RDF (Resource Description Framework) and OWL (The Web Ontology Language). Its use is widespread across various domains (conceptual modelling, Semantic Web, bioinformatics, FAIR data, artificial intelligence, and others). In many places, it replaces XML and JSON representations. Although some aspects in linking can be seen similar in XML (and XPath, XSD, XSLT, or XMLNS), the simplicity and versatility of RDF prevails. (Breitman et al., 2010)

The core ideas of keeping the information in triples composed of subject, predicate, and object, where each can be uniquely identified resource enable simple knowledge modelling and referencing. Then, to give meaning to the triples (describe classes and their properties including constraints), ontology can be used, e.g. RDF Schema (RDFS) or OWL with higher expressiveness. Finally, SPARQL can be used in a standard way to query knowledge from RDF

¹<https://github.com/OpenPonk>

knowledge representation as well as manipulate it, including transformation. (Powers, 2007)

2.6 Conceptual Modelling with RDF

The versatility of ontologies, including OWL and RDF, is also manifested in several process-oriented ontologies. Our work can benefit from some that describe novel approaches or contain important lessons learned. For example, (Garanina et al., 2019) design a process ontology with a focus on verification. (Natschläger, 2011) proposes an ontology for BPMN2 standard. (Kchaou et al., 2021) describes a transformation from BPMN to OWL, which is similar to our goal. Another interesting approach in terms of business process modelling is presented in (von Rosing et al., 2015). It aims to create a vocabulary of the terminology used in business process models. It is a potential source for linking (in linked data way) with RDF representing BORM models. To support reasoning and queries over UML models, UML to OWL transformation (Wei et al., 2018) is showing to be a promising way which further allows to utilize complex SPARQL queries (Wei and Sun, 2021).

3 BORM ONTOLOGY

This section described the OWL ontology based on BORM for capturing the same knowledge represented by OR and BA diagrams. OWL is used instead of RDF Schema for required features in terms of constraints and meta-modelling. The examples and relation to diagrams from Craft.CASE is provided in Section 4 based on the ontology.

The BORM Ontology² is designed based on BORM metamodel for OR and BA diagrams. The ontology was developed using Protégé tool (Musen, 2015) and then structurally improved in its Turtle textual format. It is a single ontology describing both metamodels for OR and BA diagrams as those are highly interlinked. It cannot use only RDFS as OWL is used for metamodelling support. The suggested prefix for BORM ontology is `borm:.` The naming is taken directly from BORM metamodel; changes and additional elements are explicitly mentioned further.

3.1 Business Architecture

The business architecture part of the ontology contains four core classes:

- `borm:Business` is not directly defined in the BORM method. However, we need a single top-level entity that bounds others. It represents a modelled business organisation.
- `borm:Function` represents a business function. It has two subclasses `borm:InternalFunction` and `borm:ExternalFunction` (disjoint, union).
- `borm:Scenario` supports a certain function a business and serves as a container for related processes.
- `borm:Process` represents a single object-relation diagram, i.e., description of a business process that is part of a certain scenario.

All of these classes require standard `rdfs:label` for human-readable name. Use of `rdfs:comment` is recommended for brief explanation. There are three object properties for relating entities `borm:hasFunction`, `borm:hasScenario`, and `borm:hasProcess` (in the ontology with corresponding domain and range). As inverse relations, `borm:ofBusiness`, `borm:ofFunction`, and `borm:ofScenario` are defined. The semantics of the properties of BORM are different from generic part-of taxonomies which is the reason for not using existing like it is done for labels.

For expressing relations between scenarios, the corresponding object properties are included: `borm:usesScenario`, `borm:extendsScenario`, and `borm:followsScenario`. Again, with its inverted variants. The properties *follows* are used to express transitions between scenarios.

The following properties are defined for the class `borm:Scenario` to provide additional details based on BORM: `borm:hasInitiation`, `borm:hasAction`, and `borm:hasResult`. Initially, those were datatype properties using a string literal. However, to allow linking triggers, actions, and results in linked-data sense, those are designed as object properties and additional classes were created: `borm:Initiation`, `borm:Action`, and `borm:Result`. With that, one can specify just a description for (anonymous) individual as well as create own subclass that has additional properties and links. For example, a result may be a completed order that requires total price computed.

3.2 Object Relations

The object relations part of the ontology is more complex than the business architecture. First, there are five classes to express participants in processes and their roles (flows):

²<http://purl.org/ontoborm>

- `borm:Participant` represents a type of stakeholders that may participate in processes. There are three subclasses (disjoint, non-union) inspired by OpenPonk and OpenCABE: `borm:Person`, `borm:System`, `borm:Organization`.
- `borm:Role` is used to relate a participant with a process, i.e., a participant has a role in a process. This could be seen as an object property; however, we need to specify additional details and have a reference to distinguish the different roles of the same participant in multiple processes.
- `borm:State` represents a state within a role. There are two special subclasses for start and end state.
- `borm:Activity` represents an activity within a role that serves for transition between two different states (that are within the same role).
- `borm:Transition` is used to express the transition between two states via activity. It is not done through object relation for the same reasons as role – there is additional information required for its instances.

Again, well-known `rdfs:label` is required and `rdfs:comment` are recommended for human-readability. The union of classes `borm:State` and `borm:Activity` forms `borm:RoleElement` which has property `borm:ofRole`. A role uses object properties `borm:startsWith` and `borm:endsWith` with an corresponding state subclass instances. A transition can be related to states using `borm:sourceState` and `borm:targetState`, and with an activity by `borm:transitsThrough`. Then, a participant is related to its role by `borm:hasRole`, and role to process by `borm:ofProcess`. The last part is to capture relations between roles, and related constraints:

- `borm:Communication` represents a links between activities of different roles within the same process. It has two subclasses (union, disjoint) for synchronous and asynchronous communication.
- `borm:DataFlow` can be attached to a communication. It is intended to be used by domain-specific class, e.g., `Order` can be `borm:DataFlow`.
- `borm:Constraint` can be used for both communication and transition and specifies a condition as in BORM which result into two subclasses (disjoint, union). It should be described by text, but can be related to domain-specific entity similarly to data flow.

A communication is related to activities by `borm:sourceActivity` and `borm:targetActivity` object properties. To distinguish the directionality of data flows, there are two sub-properties of

`borm:hasDataFlow`, namely `borm:hasInputFlow` and `borm:hasOutputFlow`. The object property `borm:ofCommunication` forms link between a communication constraint and a communication, and correspondingly for transitions (`borm:ofTransition`).

Then, to support sub-process (more specifically a process flow within a state), a `borm:State` is a subclass of that abstracts the common properties, i.e., the ability to contain a process flow (have role elements, starting and ending states).

3.3 Constraints

There are identified five rules that must hold and are implemented using SHACL:

- A transition relates two non-equal states within the same role.
- A communication relates two activities of two different roles within the same process.
- If a role has a state, it must have at least one start state and at least one end state.
- All flows must start in a start state.
- All flows must ultimately end in an end state.

3.4 Levels of Abstractions

As mentioned, some of the classes are designed to be instantiated by domain-specific entities. For example, it allows `org:SalesPerson` to be of type `borm:Person`. Then, an instance of a salesperson can be a particular human being, e.g., John to be of type `org:SalesPerson` (which is also `owl:Class`). The same principles may be applied to all of our classes. There are three abstraction levels:

1. Metamodel level = BORM ontology in OWL that defines how to define processes.
2. Model level = BORM model in RDF+OWL that defines the processes in an organisation in compliance with BORM ontology.
3. Instance level = BORM model in RDF that captures instances of processes (e.g. who and when participated in the process).

Our target in this work is the first two levels. However, the third level is fully supported and can help organisations track processes, simulate and verify them.

3.5 Hierarchies in BORM

The last missing part of the BORM metamodel related to BA and OR diagrams is the option to specify hierarchies of participants and products (used

as data flows). Such hierarchies fall into level 2, i.e., are part of the model, where it can be expressed through standard `rdfs:subClassOf` property. For example, `org:SalesPerson` may have subclass `org:SalesChiefPerson`. Such hierarchies often exist in domain ontologies, and including a special one in the ontology would add an additional burden.

4 BORM MODEL IN RDF

This section demonstrates how the ontology can be used to represent a BORM model in RDF. It uses one of the default examples of Craft.CASE tool with *E-Shop* model. In the following examples, we omit prefixes and other details for the sake of clarity and brevity. Similarly, the example identifiers also do not use long UUIDs but short names (e.g. `fun1`).

4.1 Business Architecture

In the BA diagram in Figure 1, four business functions are captured, two external (light grey) and two internal (dark grey). One internal function has a single scenario. In external functions, there are five scenarios linked with various relations. Three of them have processes (linked OR diagrams). There is the single top-level individual of `borm:Business` with those four functions. The scenarios are linked to the functions and between themselves. The processes are linked to scenarios as shown in Source Code 1.

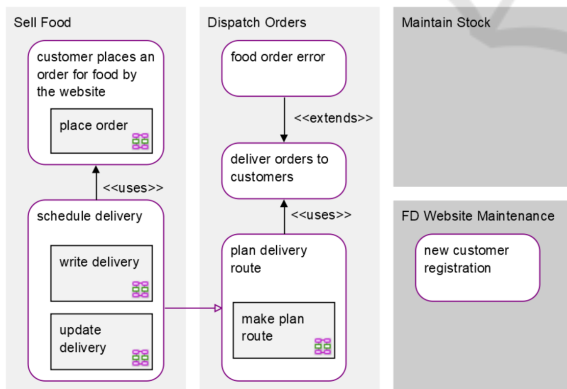


Figure 1: Business Architecture Diagram of E-Shop.

4.2 Participants

Before capturing the knowledge from OR diagrams, the next step is to describe the participants that are used across these diagrams. The hierarchy of participants may also be included. For the E-Shop case,

Source Code 1: Example of BORM BA in RDF.

```

:myEShop a borm:Business ;
  rdfs:label "My E-Shop Example" ;
  borm:hasFunction :f1, :f2, :f3, :f4 ;

:f1 a borm:InternalFunction ;
  rdfs:label "Sell Food" ;
  rdfs:comment "...".

:sce2 a borm:Scenario ;
  borm:usesScenario :sce1 ;
  borm:followedByScenario :sce3 ;
  borm:ofFunction :fun1 ;
  borm:hasInitiation :init21 ;
  rdfs:label "schedule delivery" ;
  rdfs:comment "...".
    
```

there are eight participants with a hierarchy for suppliers as shown in Source Code 2.

Source Code 2: Example of participants in RDF.

```

:par1 a borm:Participant, owl:Class ;
  rdfs:label "Supplier".

:par2 a borm:Participant ;
  rdfs:label "Supermarker Supplier" ;
  rdfs:subClassOf :par1.
    
```

4.3 Object Relations

Then for each OR diagram, i.e., a process in terms of the ontology, roles and their parts with links can be captured. From the E-Shop example, the *update delivery* (Figure 2) process has been selected. Part of its RDF representation is shown in Source Code 3. Notice that states `:st1` and `:st4` (start and end states) do not have to define label due to the use of the subclasses. For the data flow `:df1`, it would also be possible to specify the data structure used for the flow.

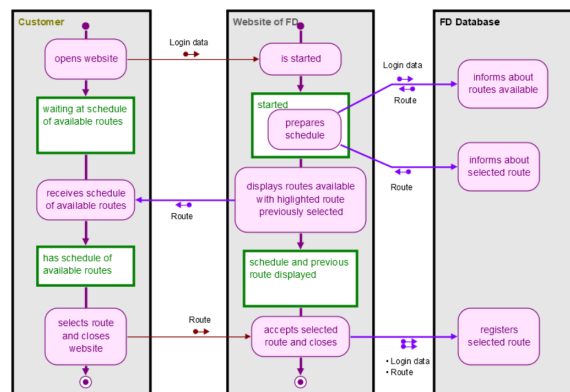


Figure 2: Object Relations Diagram of E-Shop.

Source Code 3: Example of BORM OR in RDF.

```

:rol a borm:Role ;
  borm:startsWith :st1 ;
  borm:endsWith :st4 .

:st1 a borm:StartState ;
  borm:ofRole :rol .

:act1 a borm:Activity ;
  borm:ofRole :rol ;
  rdfs:label "opens website" .

:tr1 a borm:Transition ;
  borm:sourceState :st1 ;
  borm:targetState :st2 ;
  borm:transitsThrough :act1 .

:df1 a borm:DataFlow ;
  rdfs:label "Login data" .

:col a borm:SynchronousCommunication ;
  borm:sourceActivity :act1 ;
  borm:targetActivity :act4 ;
  borm:hasInputFlow :df1 .

```

5 EXAMPLE USE CASES

SPARQL as part of the Semantic Web stack (Breitman et al., 2010) can be used to efficiently query information from a BORM model in RDF. A business analyst can have several queries prepared and execute them with different models. Helpful might be queries that count certain elements or patterns in the model, for example, number of activities for each participant across all processes as shown in Source Code 4. Also interesting are ASK queries that yield true or false, for instance, number of communications between two participants as shown in Source Code 5. It can be even used to validate the model, a more complex query can ask if it is valid (does not violate any of the five constraints stated in this paper). DESCRIBE can help to find out more information about a particular entity in a model, especially if it is linked with other RDF data. Finally, it can be used to CONSTRUCT new statements from a model.

As already shown in shorter examples in previous sections, knowledge from different sources can be linked to create a more precise domain description. We tried to transform a UML Class Diagram to OWL according to (Zedlitz et al., 2011) and link it with an overlapping BORM model in RDF/OWL. An example for this approach is shown in Source Code 6. Then, we also experimented with additional ontologies. In the integration, some of the classes from UML were identical to participants and data flow en-

Source Code 4: Example of SPARQL query to count activities for each participant.

```

SELECT
  ?p, ?p_label, (count(distinct ?a) as ?cnt)
WHERE {
  # PREFIXes
  ?a rdf:type borm:Activity .
  ?a borm:ofRole ?r .
  ?r borm:ofParticipant ?p .
  ?p rdfs:label ?p_label .
} ORDER BY DESC(?cnt)

```

Source Code 5: Example of SPARQL query to check if there is communication from p1 towards p2 in any process.

```

ASK {
  # PREFIXes
  :p1 borm:hasRole ?r1 .
  ?r1 borm:hasElement ?e1 .
  :p2 borm:hasRole ?r2 .
  ?r2 borm:hasElement ?e2 .
  ?c a borm:Communication .
  ?c borm:sourceActivity ?e1 .
  ?c borm:targetActivity ?e2 .
}

```

titles in BORM. Such a mapping could be done even semi-automatically with use of natural language processing (NLP) methods.

Source Code 6: Example of Linked BORM Model.

```

# ... common imports
@import borm: http://purl.org/ontoborm
@import m2: https://example.com/my-uml
@import : https://example.com/my-borm

:SalesPerson a borm:Person, owl:Class .

<https://orcid.org/0000-0001-7525-9218>
  a borm:Person, m2:Human, foaf:Person ;
  rdf:label "John Walker" ;
  foaf:name "John Walker" ;
  m2:firstName "John" ;
  m2:lastName "John" ;
  borm:hasRole :role01 .

```

We managed to implement an export feature in the OpenPonk modelling platform for BORM models with the proposed RDF representation. It plainly creates XML as a serialisation of the model. However, there were two limitations encountered with OpenPonk. First, there is a lack of support of RDF in Pharo, limiting the use of RDF/XML format. Second, the current implementation of BORM in OpenPonk supports only OR; therefore, on import, the BA part is ignored. Nevertheless, it enables the use of OpenPonk to visualise OR diagrams from BORM in RDF.

6 DISCUSSION

The BORM ontology is published with standard metadata and is ready to be used and enhanced in the future. A community of adopters can manage such contributions. Eventually, a new version may be released with suggested ways of transforming the underlying models. The resulting ontology is classified to be valid by Protégé (Musen, 2015) as well as the reasoners Hermit (1.3.8) and Apache Jena (4.1.0) used during the development. The ontology does not have any significant issues in terms of quality assessment (Mc Gurk et al., 2017) of completeness, accuracy, understandability, compliance, availability and consistency. It has been achieved by applying the best practices during the development; however, the ontology must retain its quality when changed by various contributors. That is supported by set contribution rules and reviewing process.

The ultimate goal of promoting interoperability for BORM models has been achieved and demonstrated in the previous section. All entities in the RDF representation of a BORM model may have identifiers through which can be referenced. These references are internal, e.g., a participant in multiple processes can be identified as one entity. However, it can also be external; one may add more statements about an entity as shown for participants and particular people. Finally, having BORM in RDF allows various tooling designs for RDF and OWL.

Although interoperability was the primary objective of this work, the use of RDF and OWL to represent BORM models also improved model (and business process) analysis possibilities. As shown in the example use cases, SPARQL can be used to extract specific knowledge. For example, how many processes the participant has a role if there is any communication between two participants in the whole business, or the most complex process by counting states and transitions. With RDF, it is also possible to look for similarities across processes in order to eliminate duplicity and, as part of optimization, add a common sub-process. The BORM models in RDF can be validated in the same manner as is described by Natschläger (Natschläger, 2011) for BPMN 2.0 or queried as Wei and Sun (Wei and Sun, 2021) proposes for UML. The validation can also be done using a more modern approach with SHACL and ShEx shape expressions. Finally, semantic reasoners can be used to infer new statements of a model, for instance, concerning class hierarchies.

Future work can focus both on managing BORM models in RDF in terms of linking, querying, or evaluating its quality. In terms of development, other

import and export feature for Craft.CASE and OpenCABE formats would be helpful for historical models in addition to our support in OpenPonk (as an open-source and actively developed tool). The RDF representation and use of the OpenPonk modelling platform should streamline prototyping of such improvements and their subsequent refining.

7 CONCLUSIONS

In this paper, the BORM ontology has been proposed to promote the interoperability of BORM process models. The ontology is designed according to the metamodel of BORM; more specifically, its part related to business process modelling. Standard ontology metadata and documentation has been used to improve its practical usability. It has been also described how business processes can be represented in RDF by using the ontology—representing the knowledge from Object Relation and Business Architecture diagrams using RDF proven to open multiple possibilities for business analysis. The primary advantage lies in the ability to link to other domain knowledge in a linked data way. Moreover, other consequential advantages are discussed in the paper, e.g., using SPARQL to extract specific information from BORM models or visualization. Finally, possible future steps are outlined, mainly providing BORM-like visualization of the diagrams and transformation for Craft.CASE project format in addition to our RDF export from OpenPonk modelling tool.

ACKNOWLEDGEMENTS

The research was supported by the grant of Czech Technical University in Prague No. SGS20/209/OHK3/3T/18.

REFERENCES

- Bliznicenko, J., Papoulias, N., Pergl, R., and Stinckwich, S. (2017). Towards modularity in live visual modeling: A case study with openponk and kendrick. In Laval, J. and Etien, A., editors, *Proceedings of the 12th edition of the International Workshop on Smalltalk Technologies, IWST 2017, Maribor, Slovenia, September 4-8, 2017*, pages 3:1–3:10. ACM.
- Breitman, K., Casanova, M., and Truszkowski, W. (2010). *Semantic Web: Concepts, Technologies and Applications*. NASA Monographs in Systems and Software Engineering. Springer London.

- CRAFT.CASE Ltd. (2015). Craft.CASE: Business Process Analysis, version 2.4.18.1. [online][visited 2021-06-05].
- Garanina, N. O., Anureev, I. S., and Borovikova, O. (2019). Verification-oriented process ontology. *Autom. Control. Comput. Sci.*, 53(7):584–594.
- Kchaou, M., Khlif, W., Gargouri, F., and Mahfoudh, M. (2021). Transformation of BPMN model into an OWL2 ontology. In Ali, R., Kaindl, H., and Maciaszek, L. A., editors, *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2021, Online Streaming, April 26-27, 2021*, pages 380–388. SCITEPRESS.
- Knott, R. P., Merunka, V., and Polák, J. (2000). Process modeling for object oriented analysis using BORM object behavioral analysis. In *Proceedings of the 4th International Conference on Requirements Engineering, ICRE '00, Schaumburg, Illinois, USA, June 19-23, 2000*, pages 7–16. IEEE Computer Society.
- Mc Gurk, S., Abela, C., and Debattista, J. (2017). Towards ontology quality assessment. In *MEPDAW/LDQ@ESWC*, pages 94–106.
- Merunka, V., Brozek, J., Sebek, M., and Polák, J. (2009). BORM - business object relation modeling. In Nickerson, R. C. and Sharda, R., editors, *Proceedings of the 15th Americas Conference on Information Systems, AMCIS 2009, San Francisco, California, USA, August 6-9, 2009*, page 788. Association for Information Systems.
- Molhanec, M., Merunka, V., and Merunková, I. (2011). Business knowledge modelling using the BORM method. In Salamapasis, M. and Matopoulos, A., editors, *Proceedings of the 5th International Conference on Information and Communication Technologies for Sustainable Agri-production and Environment (HAICTA 2011), Skiathos, Greece, September 8-11, 2011*, volume 1152 of *CEUR Workshop Proceedings*, pages 385–396. CEUR-WS.org.
- Musen, M. A. (2015). The Protégé Project: A Look Back and a Look Forward. *AI Matters*, 1(4):4–12.
- Natschläger, C. (2011). Towards a BPMN 2.0 ontology. In Dijkman, R. M., Hofstetter, J., and Koehler, J., editors, *Business Process Model and Notation - Third International Workshop, BPMN 2011, Lucerne, Switzerland, November 21-22, 2011. Proceedings*, volume 95 of *Lecture Notes in Business Information Processing*, pages 1–15. Springer.
- Pergl, R. and Tuma, J. (2012). Opencase - A tool for ontology-centred conceptual modelling. In Bajec, M. and Eder, J., editors, *Advanced Information Systems Engineering Workshops - CAiSE 2012 International Workshops, Gdańsk, Poland, June 25-26, 2012. Proceedings*, volume 112 of *Lecture Notes in Business Information Processing*, pages 511–518. Springer.
- Podloucký, M. and Pergl, R. (2014). Towards formal foundations for BORM ORD validation and simulation. In Hammoudi, S., Maciaszek, L. A., and Cordeiro, J., editors, *ICEIS 2014 - Proceedings of the 16th International Conference on Enterprise Information Systems, Volume 2, Lisbon, Portugal, 27-30 April, 2014*, pages 315–322. SciTePress.
- Powers, S. (2007). *Practical RDF*. O'Reilly Media.
- Scholz-Reiter, B. and Stickel, E. (2012). *Business Process Modelling*. Springer Berlin Heidelberg.
- Suchánek, M. and Pergl, R. (2019). Mapping UFO-B to bpmn, borm, and UML activity diagram. In Pergl, R., Babkin, E., Lock, R., Malyzhenkov, P., and Merunka, V., editors, *Enterprise and Organizational Modeling and Simulation - 15th International Workshop, EO-MAS 2019, Held at CAiSE 2019, Rome, Italy, June 3-4, 2019, Selected Papers*, volume 366 of *Lecture Notes in Business Information Processing*, pages 82–98. Springer.
- Uhnák, P. and Pergl, R. (2016). The openponk modeling platform. In Laval, J. and Etien, A., editors, *Proceedings of the 11th edition of the International Workshop on Smalltalk Technologies, IWST 2016, Prague, Czech Republic, August 23-24, 2016*, page 14. ACM.
- von Rosing, M., Laurier, W., and Polovina, S. M. (2015). The BPM ontology. In von Rosing, M., von Scheel, H., and Scheer, A., editors, *The Complete Business Process Handbook: Body of Knowledge from Process Modeling to BPM, Volume I*, pages 101–121. Morgan Kaufmann/Elsevier.
- Wei, B. and Sun, J. (2021). Leveraging SPARQL queries for UML consistency checking. *Int. J. Softw. Eng. Knowl. Eng.*, 31(4):635–654.
- Wei, B., Sun, J., and Wang, Y. (2018). A knowledge engineering approach to UML modeling. In Pereira, Ó. M., editor, *The 30th International Conference on Software Engineering and Knowledge Engineering, Hotel Pullman, Redwood City, California, USA, July 1-3, 2018*, pages 60–63. KSI Research Inc. and Knowledge Systems Institute Graduate School.
- Zedlitz, J., Jörke, J., and Luttenberger, N. (2011). From UML to OWL 2. In Lukose, D., Ahmad, A. R., and Suliman, A., editors, *Knowledge Technology - Third Knowledge Technology Week, KTW 2011, Kajang, Malaysia, July 18-22, 2011. Revised Selected Papers*, volume 295 of *Communications in Computer and Information Science*, pages 154–163. Springer.