

Proposal for Formalization Using Description Logic of Undesirable Models in Business Process Management

Jean Elder Santana Araújo^a and Cleyton Mário de Oliveira Rodrigues^b

PPGEC, Universidade de Pernambuco, Benfica, 455, Recife/PE, Brazil

Keywords: Description Logic, Business Process Management, Modeling, BPMN.

Abstract: Proposal of a method to detect and correct errors in BPMN (Business Process Model and Notation) models using ontologies and Descriptive Logic reasoning to formalize and identify common errors related to the use of gateways, elements that control the flow and decisions in a process. The research highlights how the misuse of gateways can lead to inefficiencies and failures in process execution. Gateways are explored in the article as a demonstration of an experimental structure with a focus on expanding the application of the method to other elements of BPM modeling.

1 INTRODUCTION

Business Process Management (BPM) is essential for companies to achieve their objectives efficiently. Business process modeling is crucial for organizations, and in this context, BPMN has established itself as the most relevant notation because of its expressiveness and simplicity. However, the modeling process can present errors that compromise the efficiency of the process (Weske, 2019).

Analyzing the literature on approaches to optimizing business process management supported by ontologies, we identified some studies that focus on two main fields (Coşkunçay and Demirörs, 2022) (Mejri and Ghannouchi, 2023) (Cherfi et al., 2013) (Ternai et al., 2015) (Pham and Thanh, 2016) (Yingbo and Xia, 2019) (Fengel, 2014):

Integration. In this field, ontologies are used to improve some process of integration of systems or data related to business processes;

Compliance. Understanding compliance as the adequacy of processes or systems to rules, policies, requirements of some business segment.

The importance of error detection and correction in business modeling is already consolidated in the literature, for example, (Chinosi and Trombetta, 2012) which reports how BPMN stands out in the recording of business process modeling, ensuring consistency of

representation in contrast to the use of case descriptions and documentation of complex procedures that are often very difficult to understand and prone to errors.


This paper addresses the formalization of the logical description of business process modeling errors with the aim of improving consistency gains with the use of BPMN. In order to demonstrate its potential, it will focus on the use of gateways, an essential element in BPMN to control the divergence and convergence of flows. Inadequate use of gateways can lead to unexpected behaviors and errors in the process model, resulting in loss of efficiency.

The objective of this study is to present a proposed solution to increase the efficiency and accuracy of BPM through the detection and correction of errors in gateway modeling. The research utilizes the BBO (BPMN 2.0 Based Ontology) ontology and logic description language to formalize errors and infer solutions.

This article also describes how the proposed formalization can be applied using jBPM, a process execution engine. The approach includes reading a jBPM XML file, converting it to OWL format, and then using Protégé to infer and present the results.

The formalization of error detection and correction in BPMN models benefits the creation of more accurate, flexible, and efficient models, significantly influencing the efficiency of business process management.

^a  <https://orcid.org/0000-0003-2084-617X>

^b  <https://orcid.org/0000-0003-3816-656X>

2 BACKGROUND

This section will address the theoretical foundation of the knowledge groups that support the proposed solution to be described in the article, namely: Business Process Management in BPMN notation, ontologies, and description logics.

2.1 BPMN (Business Process Model and Notation)

A standardized graphical notation for modeling business processes. Using symbols, it represents different stages of a process such as:

- Events: Circles that represent something that happens in the process (start, end, receiving a message, etc.).
- Activities: Rounded rectangles that represent tasks to be performed
- Gateways: Diamonds that represent decision points or branching in the process flow.
- Sequence flows: Arrows that indicate the order in which activities are performed.

BPMN offers significant advantages in facilitating the understanding of complex processes, communication between different areas and levels of the organization, and process improvement, as the representation makes it easier to identify bottlenecks and automate tasks. Its application is diverse across various areas, such as sales, in mapping the initial customer contact to product invoicing, in the healthcare sector, in recording the flow of care upon a patient's admission to the ICU, and in software development.

2.2 Description Logics

Description Logics (DLs) are formal languages for representing knowledge and enabling automated reasoning. They combine concept descriptions with a rigorous logical semantics. This combination allows for efficient querying of a knowledge base, providing timely answers. The problem-solving capabilities and complexity of the inferences depend on the expressivity of the DL used.

There are various types of Description Logics, each with varying expressivity, depending on the logical elements it offers. The family of description logics explored in this paper is \mathcal{AL} that allows the negation of atomic propositions, intersection of concepts, and universal restrictions, offering limited existential quantification, namely, only the concept \top can be used within the scope of the existential quantifier.

2.3 Ontologies

An ontology is an explicit, formal specification of a shared conceptualization (STUDER, R.; SURE, Y.; STAAB, S., 2004). They formalize the knowledge of a domain by defining concepts, their properties, and relationships. When coupled with reasoners, ontologies can extract meaning from text, perform complex inferences, diagnose faults, predict events, or even automate tasks.

Both Description Logics and ontologies are formalisms used to represent knowledge. Description Logics can be used as a foundation for building ontologies, which aim to represent the knowledge of a specific domain in a way that is understandable to both humans and machines. Description Logics, on the other hand, focus on reasoning capabilities, performing inferences, and answering complex questions that can be posed about the domain represented in an ontology.

3 BUSINESS PROCESS MODELING ERRORS

This section will present examples of errors in business process modeling that result in a loss of process flow efficiency. One common error lies in the misuse of gateways, which are essential elements in BPMN for controlling the divergence and convergence of sequence flows, representing decisions and bifurcations within the process. However, the improper use of gateways can lead to unexpected behaviors and errors in the process model. Below are some common errors related to the use of gateways in BPMN.

1. OR Gateways without a default flow: Inclusive (OR) gateways must have a defined default flow for when none of the conditions of the other flows are met. The lack of this default flow can cause the process to stall.
2. XOR Gateways with unconditional flows: This situation creates ambiguity about which path the process will take and consequently leads to a loss of operational efficiency.
3. Lack of synchronization in parallel (AND) gateways: Parallel flows that are not synchronized correctly: If activities in parallel flows need to be completed before proceeding, a parallel (AND) gateway should be used to synchronize the flows.

The undesirable situations in business process modeling using BPMN notation are not limited to the cases listed above, and errors in the use of gateways extend to other applications.

To demonstrate the proposed approach in this ongoing research, the focus is on the above-listed cases and is further substantiated below.”

3.1 OR Gateways Without Default Path

When used to enable alternative paths in processes, the OR Gateway becomes problematic when no true condition exists to direct the process flow. This can lead to the process becoming blocked or stalled.

For example, consider a credit approval process that evaluates a customer’s history in different categories to determine the flow. In the case of a new customer with no history, the process would be stuck. Similarly, if the evaluation is based on criteria that do not fit into any condition, for instance, in a credit approval process that considers income ranges, if the customer’s income falls outside all defined ranges, the process would also be halted.

A solution would be to define a default output that will be followed if no other condition is true

3.2 XOR Gateways Have Conditionless Paths

Used to direct a process to a single path, the problem arises when one or more output paths do not have an explicitly defined or valid condition.

This situation results in a process error due to ambiguity, potentially halting the process if the execution engine is unable to determine the path to follow.

As a solution, in addition to reviewing the model itself, a default path can be adopted by defining an else condition that covers all other conditions, ensuring a directed path.

3.3 Lack of Synchronization in Parallel Gateways (AND)

Used to split the process flow into multiple paths that are executed concurrently, a lack of synchronization between paths can result in errors due to incomplete executions if the process continues without the completion of a task in any path. Errors can also arise when these parallel paths share information.

As a solution, the existence of a parallel convergence path that verifies the exit conditions of all paths would ensure that all paths have fully completed their tasks before the process continues in a consistent and correct manner.

4 BBO: BPMN 2.0 BASED ONTOLOGY FOR BUSINESS PROCESS REPRESENTATION

The BBO (BPMN 2.0 Based Ontology) (Annane et al., 2019) is an ontology developed for the representation of business processes based on the reuse of existing ontologies (Falbo and Bertollo, 2009); (Karray et al., 2012); (Chungoora et al., 2013); (Fraga et al., 2018) and metamodels such as BPMN 2.0 (Object Management Group (OMG), 2013)

This ontology provides a formalization that serves as a foundation for the construction proposed here. Below is a figure representing a segment of the BBO.

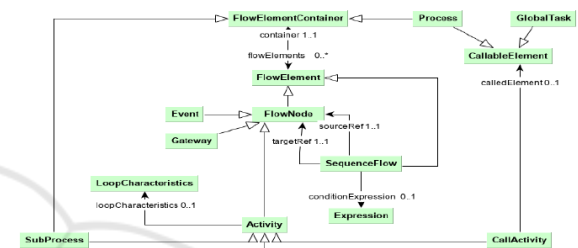


Figure 1: Process class properties and related concepts from BPMN reused in BBO.

In this context, the main elements of Figure 1 are described below.

- Activity is the work to be performed. The Activity class has three subclasses:
 1. Task: an atomic task
 2. Sub-Process: a complex task that contains multiple Tasks
 3. CallActivity: an activity that calls a CallableElement which can be a Global-Task (i.e., a reusable task) or Sub-Process.
- Event is something that "happens" during the course of a process. Events affect the flow of the process and generally have a cause or an impact and may require or allow a reaction.
- Gateway is used to control how SequenceFlows interact as they converge or diverge within a Process.

Specifically, regarding the Gateway that will be the focus of this article, we extend the already presented Gateway concept, exposing that it has an attribute that determines the type of gateway: "convergent", "divergent", "multiple".

The BBO has been extended with elements to encompass the formalization and analysis of modeling errors using Gateway constructs, which have been presented as illustrative examples.

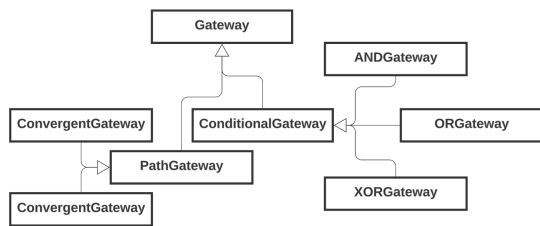


Figure 2: BBO Extension - Gateway Construct (Part 01).

Continuing the extensions for constructing solutions and alternatives to undesirable situations, or rather, those that impact the efficiency of process flow in modeling using BPMN notation, the focus is now differentiated towards Sequence Flows.

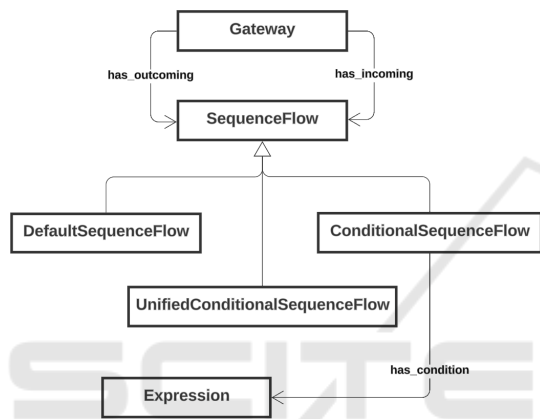


Figure 3: BBO Extension - Gateway Construct (Part 02).

5 FORMALIZATION OF BPMN ELEMENTS IN DESCRIPTION LOGIC

In extension to the most comprehensive representation of the BPMN elements, the formalization in Description Logic is called upon for this task.

Focusing mainly on the formulations involving the Gateways, the following records are presented

5.1 Gateway as the Branches

Throughout the process, there will be branching points in its flow, and the BPMN element that represents this phenomenon is the Gateway. Thus, this branching has two basic behaviors: convergence or divergence, which already represents the fundamental characteristic for the use of this element.

A Gateway **MUST** have either multiple incoming or multiple outgoing sequence flows. Based on this combination, it is classified as Convergent, Divergent, or Multiple.

Convergent. A Gateway with a gatewayDirection of converging **MUST** have multiple incoming Sequence Flows, but **MUST NOT** have multiple outgoing Sequence Flows.

Divergent. A Gateway with a gatewayDirection of diverging **MUST** have multiple outgoing Sequence Flows, but **MUST NOT** have multiple incoming Sequence Flows.

Mixed. A Gateway **MUST** have either multiple incoming and multiple outgoing sequence flows.

Formalization Gateway	
<i>Gateway</i>	<i>SubClassOf</i>
$((has_incoming \min 2 \text{ SequenceFlow})$ or $(has_outgoing \min 2 \text{ SequenceFlow}))$	
Formalization Converging	
<i>ConvergingGateway</i>	<i>equivalentTo</i>
$Gateway$ and $(has_incoming \min 2 \text{ SequenceFlow})$ and $(has_outgoing \text{ exactly } 1 \text{ SequenceFlow})$	
Formalization Divergent	
<i>DivergingGateway</i>	<i>equivalentTo</i>
$Gateway$ and $(has_outgoing \min 2 \text{ SequenceFlow})$ and $(has_incoming \text{ exactly } 1 \text{ SequenceFlow})$	
Formalization Mixed	
<i>MixedGateway</i>	<i>equivalentTo</i>
$Gateway$ $and (has_incoming \min 2 \text{ SequenceFlow})$ $and (has_outgoing \min 2 \text{ SequenceFlow})$	

5.2 Gateway as to the Conditions

Regarding the conditionals of incoming and outgoing sequence flows at Gateways, their presence or absence determines the path that the process will follow. They share the construct *ConditionalSequenceFlow*, which is formally defined below.

Formalization	
<i>ConditionalSequenceFlow</i>	<i>equivalentTo</i>
$SequenceFlow$ and $(has_condition \text{ Expression } some \text{ Expression})$	
<i>Expression</i>	<i>equivalentTo</i>
$(True \text{ or } False)$	
<i>ConditionalGateway</i>	<i>equivalentTo</i>
$(ConvergingGateway$ or $DivergingGateway)$ and $(has \text{ only } ConditionalSequenceFlow)$	

The focus of this paper is on Gateways and their conditional requirements for sequence flow progression, specifically the Sequence Flows and their corresponding logical values for process continuation

AND. All logical values must be true.

OR. At least one logical value must be true.

XOR. Exactly one logical value must be true.

Formalization
$ANDGateway \text{ equivalentTo } CondicionalGateway$ $and \ \forall x \in Expression \sqsubseteq True$
$ORGateway \text{ equivalentTo } CondicionalGateway$ $and \ \exists x \in Expression \sqsubseteq True$
$XORGateway \text{ equivalentTo } CondicionalGateway$ $and \ \exists! x \in Expression \sqsubseteq True$

The extension of this formalization for the aforementioned Gateways is presented above.

5.3 Descriptive Logic-Based Solution Formulation

BPMN notation is a powerful tool for modeling business processes; however, some situations may compromise the effectiveness of the business process in question. In Section 3 of this work, some of these situations were listed within the construct of the Gateway and their corresponding associated solutions.

The formalizations in Description Logics presented in this section are built upon the ontological structure in conjunction with the BBO extensions. This enables reasoning and inferences that support the identification of undesirable situations in modeling and the corresponding proposed solutions.

Given that the current scope is limited to constructing a proposal with an exemplary focus on Gateway constructs (Christiansen et al., 2010), the simplified ontological structure presented in the following is restricted to this aspect for now, with the intended future extension objective.

For illustrative purposes, the figure below depicts the proposed solution's structure with a gateway, outlining the paths for incorporating future handling of undesirable situations.

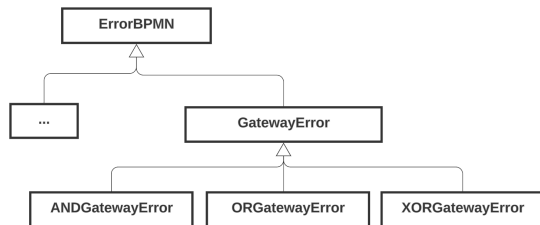


Figure 4: Error structure.

Gateways are essential elements in BPMN modeling, serving as decision points and flow control mechanisms within a business process (Marin-Castro and Tello-Leal, 2021). They enable the process to fol-

low different paths based on specific conditions, thus modeling the logic and behavior of the process more accurately and comprehensively.

Although formalizations are limited to the use of gateways, their impact as powerful tools in BPMN modeling contributes to the creation of more precise, flexible, and efficient models, significantly influencing the efficiency of business process management.

At this moment, the formalization of these solutions in Description Logic, which supports the object of study of this work, will be exposed.

5.3.1 OR Gateways Without Default Path

Solution: Default Path

Formalization
$ORGatewayError \text{ equivalentTo } ORGateway \text{ and } (\neg \exists x \in Expression \sqsubseteq True)$
$ORGatewayErrorSolved \text{ equivalentTo } ORGatewayError \text{ and } (has_outgoing \text{ exactly } 1 \text{ SequenceFlowDefault})$

5.3.2 XOR Gateways Have Conditionless Paths

Solution: Default Path

Formalization
$XORGatewayError \text{ equivalentTo } XORGateway(G) \text{ and } has_Condition(G,x) \text{ and } (\neg \exists x \in Expression)$
$XORGatewayErrorSolved \text{ equivalentTo } XORGatewayError \text{ and } (has_outgoing \text{ exactly } 1 \text{ SequenceFlowDefault})$

5.3.3 Lack of Synchronization in Parallel Gateways (AND)

Solution: parallel convergence path

Formalization
$ANDGatewayError \text{ equivalentTo } ANDGateway \text{ and } (\neg \exists x \in Expression \sqsubseteq True)$
$ANDGatewayErrorSolved \text{ equivalentTo } ANDGatewayError \text{ and } (has_outgoing \text{ exactly } 1 \text{ SequenceFlowUnifiedCondition})$
$SequenceFlowUnifiedCondition(G1) \text{ equivalentTo } \forall x has_condition(G1,x) \text{ and } Expression(x) \text{ and } ANDGateway(G2) \text{ and } has_condition(G2,x)$

6 APPLICATION

The approach to applying formalization in the description logic of errors in BPMN models utilized adaptations of methods employed in previous research. (Ternai et al., 2015)

Although there are other tools capable of performing processes, jBPM seemed to be the most appropriate tool to achieve our goals.

Considering, for the purpose of analysis and correction, the execution of a fictitious process starting with flow 1 leading to an inclusive OR gateway that diverges into two flows, 2 and 3, respectively, which execute tasks A and B if their conditional expressions are true. At this point in the execution, flow 4, originating from the aforementioned gateway, leads to a join gateway that triggers an end event.

However, if the conditional expressions of flows 2 and 3 are false, neither task A nor task B will be executed, and the process will terminate without any action being taken. The solution to this situation, as already explained, is to define a default flow that is triggered when all divergent flows from the inclusive OR gateway are invalid or false.

In other words, the default flow must be constructed with the combination of all conditions of the diverging flows from the OR gateway. In this example, the condition for this default flow would be as follows:

$$\neg(\text{conditionA} \text{ OR } \text{conditionB})$$

The solution presented to address the undesirable situation described in Section 3.1 is materialized in the jBPM XML code shown in the figure above through the defaultFlow. This flow aggregates the conditions of flow 1 and flow 2:

```

<process id="myProcess" name="My Process">
  <startEvent id="start" />
  <sequenceFlow id="flow1"
    sourceRef="start"
    targetRef="gateway" />
  <inclusiveGateway id="gateway"
    name="OR Gateway" />
  <sequenceFlow id="flow2"
    sourceRef="gateway"
    targetRef="taskA">
    <conditionExpression
      xsi:type="tFormalExpression">
        #{conditionA}
      </conditionExpression>
    </sequenceFlow>
  <sequenceFlow id="flow3"
    sourceRef="gateway"
    targetRef="taskB">
    <conditionExpression
      xsi:type="tFormalExpression">
        #{conditionA}
      </conditionExpression>
    </sequenceFlow>
  <sequenceFlow id="flowDefault"
    sourceRef="gateway"
    targetRef="taskDefault">
    <conditionExpression
      xsi:type="tFormalExpression">
        #{conditionDefault}
      </conditionExpression>
    </sequenceFlow>
  </process>

```

```

<sequenceFlow id="flow4"
  sourceRef="gateway"
  targetRef="join" />
<userTask id="taskA"
  name="Task A" />
<userTask id="taskB"
  name="Task B" />
<userTask id="taskDefault"
  name="Task Default" />
<joinGateway id="join"
  name="Join" />
<sequenceFlow id="flow5"
  sourceRef="join"
  targetRef="end" />
<endEvent id="end" />
</process>

```

Figure 5: Example - OR gateway with default.

To detail the XML elements that represent the example of using the OR Gateway in the jBPM tool, see the list below.

Inclusive Gateway. Each outgoing sequenceFlow from the gateway has an associated condition. If conditionA is evaluated as true, the flow proceeds to taskA. If conditionB is true, the flow progresses to taskB. In particular, both conditions can be true simultaneously, resulting in the execution of both tasks.

Join Gateway. This gateway serves to synchronize the flow after the inclusive gateway. The process continues only after all tasks activated by the inclusive gateway have been completed.

Conditions. The conditions can be any valid Boolean expression within the context of jBPMN.

An initial step involves populating an ontology with general data from a Business Process Conceptual Model, utilizing an ontology instantiation mechanism. This transforms the process representation into a formal ontology structure, facilitating reasoning and inference.

Instantiation entails populating the ontology with real-world data, thereby creating instances of the classes defined within the ontology. This involves creating individuals and assigning properties and relationships based on the structure defined in the ontology, which also encompasses the axioms formalized in Description Logic, as previously discussed. This process comprises a sequence of steps:

Identify the Individuals Determine which real-world entities will be represented as instances in the ontology.

Create Instances Use the chosen tool to create instances of the relevant classes for each individual.

Populate Properties Assign values to the properties of each instance, describing its characteristics.

Define Relationships Establish relationships between instances that represent the connections between them.

The data and relationships of the actual processes would be extracted through analysis using optimized mechanisms from XML files that record process elements.

To perform the task of reading the jBPM XML file, instantiating the ontology, and subsequently performing inference, it is necessary to utilize a set of tools. Initially, for demonstration purposes, the following tools appear suitable:

- SAX API (Simple API for XML): For sequential processing of the file and preparation of the OWL file for consumption in the instantiation and inference phase.
- Ontology instantiation: Employ an ontology instantiation tool (Protégé) for analysis of instances in the ontology and subsequent inferences.

As noted above, jBPM utilizes variables to store process information. These variables persist in a database. This facilitates the automation process, enabling direct reading of process data and conversion into a format suitable for the ontology and reasoning tool employed in this work, namely Protégé

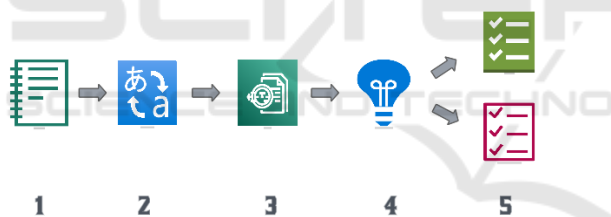


Figure 6: Proposal steps.

Figure 8 illustrates the sequence of steps in the proposed approach for analyzing BPMN elements to identify the mapped errors and indicate the corresponding solutions. The five main steps are as follows:

1. Read the XML file from the jBPM process execution engine.
2. Convert / treat XML (Nika et al., 2022).
3. Read the OWL file generated in step 2 using the Protégé tool.
4. Execute inference using reasoners within Protégé.
5. Present the results, including undesirable situations and those with solutions already mapped in the ontology.

The results display will present, as an outcome of the inference, the signaling classes. For example,

identifying the use of an OR gateway (inclusive) will signal the possibility of an ErrorORGateway. If the implementation of the standard flow is identified, it will signal SolverORGateway, indicating that a corresponding treatment measure has been taken to account for the possibility of error.

Accordingly, two lists are presented below: cases of possible errors followed by titles indicating the implemented solution measures.

Errors	Solvers
ErrorORGateway	SolverORGateway
ErrorANDGateway	SolverANDGateway
ErrorXORGateway	SolverXORGateway

7 FUTURE WORK

This study has presented a formalization for specifying business processes using Description Logic, with a focus on identifying and correcting common errors in BPMN models. However, there are several opportunities to deepen this research.

Although the errors in the BPMN models presented here represent only a small sample, there is a vast field of study to be explored. This includes the potential to map and consolidate antipatterns in business process modeling and their corresponding solutions, similar to the work done on antipatterns in object-oriented software engineering. (Gamma, 2009) This research would focus specifically on identifying and addressing recurring problematic patterns in business process models, ultimately contributing to improved modeling practices and process quality.

One promising direction is to expand the formalization language to capture more complex aspects of processes, such as time, resources, and uncertainty. In addition, integration with machine learning techniques can enable automatic error detection and the generation of BPMN models from historical data.

Another area of interest is the development of decision support tools that use the proposed formalization to assist in process analysis and optimization. These tools can be used to identify bottlenecks, simulate different scenarios, and evaluate the impact of changes in processes.

Furthermore, the ontological segments constructed for simulations can be extended to encompass all elements of the BPMN notation, based on a review of the literature concerning undesirable situations. Furthermore, the conceptual conformity of a foundational ontology such as UFO (Guizzardi, 2005) can be assessed to improve the conceptual validity of the ontology and the semantic rules mapped from the literature.

8 CONCLUSION

This study delved into the significance of Business Process Management (BPM) and the role of BPMN in business process modeling. While BPMN is a powerful tool, it can be prone to errors, particularly when dealing with complex processes, especially when the process flow becomes lost in the ramifications of flows at their convergences and divergences in the paths taken by the process. At some point, some conditional may be poorly evaluated, generating unwanted situations and impacting the performance of business process management.

Thus, this work highlighted common errors in BPMN modeling, specifically focusing on issues related to gateways. Errors that can lead to process inefficiencies and, ultimately, implementation failures. To address these challenges, we proposed a formalization approach using Description Logic.

By representing BPMN concepts and constraints in a formal language, we can increase the accuracy and consistency of process models. This formalization can be used to automate various tasks, such as model validation, simulation, and analysis, leading to more reliable and efficient processes.

Future research directions include exploring the application of advanced reasoning techniques, such as ontology-based reasoning, to further refine the formalization of BPMN. In addition, investigating the integration of machine learning techniques to automate error detection and correction is a promising avenue.

In conclusion, the proposed formalization of undesirable situations with the application of reasoning tasks has proven to be an important element in enhancing the quality and effectiveness of BPMN modeling, ultimately contributing to improved organizational performance and decision-making. This, in turn, can help organizations gain a competitive advantage in their niche.

REFERENCES

- Annane, A., Aussenac-Gilles, N., and Kamel, M. (2019). Bbo: Bpmn 2.0 based ontology for business process representation. In *20th European Conference on Knowledge Management (ECKM 2019)*, volume 1, pages 49–59.
- Cherfi, S. S.-S., Ayad, S., and Comyn-Wattiau, I. (2013). Aligning business process models and domain knowledge: a meta-modeling approach. In *Advances in Databases and Information Systems*, pages 45–56. Springer.
- Chinosi, M. and Trombetta, A. (2012). Bpmn: An introduction to the standard. *Computer Standards & Interfaces*, 34(1):124–134.
- Christiansen, D. R., Carbone, M., and Hildebrandt, T. (2010). Formal semantics and implementation of bpmn 2.0 inclusive gateways. In *International Workshop on Web Services and Formal Methods*, pages 146–160. Springer.
- Chungoora, N., Young, R. I., Gunendran, G., Palmer, C., Usman, Z., Anjum, N. A., Cutting-Decelle, A.-F., Harding, J. A., and Case, K. (2013). A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. *Computers in industry*, 64(4):392–401.
- Coşkunçay, A. and Demirörs, O. (2022). A method for integrated business process modeling and ontology development. *Business Process Management Journal*, 28(3):606–629.
- Falbo, R. D. A. and Bertollo, G. (2009). A software process ontology as a common vocabulary about software processes. *International Journal of Business Process Integration and Management*, 4(4):239–250.
- Fengel, J. (2014). Semantic technologies for aligning heterogeneous business process models. *Business Process Management Journal*, 20(4):549–570.
- Fraga, A. L., Vegetti, M., and Leone, H. P. (2018). Semantic interoperability among industrial product data standards using an ontology network. In *ICEIS (2)*, pages 328–335.
- Gamma, E. (2009). *Padrões de projetos: soluções reutilizáveis*. Bookman editora.
- Guizzardi, G. (2005). Ontological foundations for structural conceptual models. *University of Twente*.
- Karray, M. H., Chebel-Morello, B., and Zerhouni, N. (2012). A formal ontology for industrial maintenance. *Applied ontology*, 7(3):269–310.
- Marin-Castro, H. M. and Tello-Leal, E. (2021). An end-to-end approach and tool for bpmn process discovery. *Expert Systems with Applications*, 174:114662.
- Mejri, S. and Ghannouchi, S. A. (2023). A proposed guidance approach for bp performance improvement. *Procedia Computer Science*, 225:1425–1437.
- Nika, Z. G., Khadivar, A., Dadkhah, C., and Rahimiand, S. (2022). Developing an ontology for business process management techniques and tools. *University of Twente*.
- Object Management Group (OMG) (2013). Business Process Model and Notation (BPMN) Version 2.0.
- Pham, T. A. and Thanh, N. L. (2016). An ontology-based approach for business process compliance checking. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication*, pages 1–6.
- Ternai, K., Khobreh, M., and Ansari, F. (2015). *An ontology matching approach for improvement of business process management*. Springer International Publishing. Cited by: 2.
- Weske, Mathias Weske, M. (2019). *Business Process Management - Concepts, Languages, Architectures*. Springer, Potsdam, Germany, 4 edition.
- Yingbo, D. and Xia, H. (2019). Business modeling and reasoning based on process ontology. In *Proceedings of the 5th International Conference on Frontiers of Educational Technologies*, pages 143–147.