Semantic Preserving, Notational and Transformational Challenges in Transfiguring BPMN models into Petri Nets

Karnika Shivhare[1], Rushikesh K. Joshi[1]

[1]*Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Mumbai - 400 076, India.*

Abstract

The paper identifies several issues and problems in converting BPMN models into formal Petri Net models. The conversion of BPMN models into Petri nets has been attempted for about a decade, but the richness and diversity in the BPMN's toolset continues to demand research attention towards completeness and accuracy in conversion of BPMN elements into equivalent and modular and hence comprehensible Petri net constructs. The paper discusses two classes of modeling gaps, and presents hidden and overlooked hurdles that cause inaccurate transformation of BPMN models. These hurdles encompass *challenges* encountered by modelers during actual transformation, *impediments* arising due to notations, and *pitfalls* faced while designing transformation strategies in order to accurately accommodate semantics.

Keywords

Business Processes, BPMN, Petri Nets, Processes Modeling, Model Transformation

1. Introduction

Business Process Model and Notation (BPMN), a standard specification for high level business process modeling, is generally coupled with Petri Nets (PNs) for purposes of formal analysis, verification, etc. of BPMN process models. BPMN can handle the visual modeling segment for the processes, and Petri nets can be utilised for their formal analysis and verification. However, we observe incompleteness, inconsistencies and ambiguities in transformation rules in the current literature. We elucidate them as *Transformational Challenges*. They include *Notational Impediments*, such as ambiguous descriptions of a few BPMN elements, transformational inadequacy for redundant BPMN notations, etc. Furthermore, we present *Semantics Preservation Pitfalls*, that occur on account of semantics, and need to be addressed for preserving features of process-oriented visual modeling notations. This paper brings out these challenges that need to be addressed for precise transformation of BPMN models.

2. Behavioral Semantic Gaps

Semantic gaps are the discrepancies or inconsistencies in the meaning or interpretations. They are presented in the context of process modeling in [1] [2]. We define these gaps at two edges of mapping levels in lifespan of processes, specifically at modeling and analysis phase.

Intended and Modeled Behavior Gaps. Semantic gap between intended process behavior and modeled behavior refers to the difference between the desired behavior of a process and its actual representation in a high level visual model. These gaps have also been elaborated, analysed for root cause and attempted for reduction by Karnika and Joshi [2]. Such class of semantic gaps exist due to notational defects of high level visual modeling languages such as non-compactness [2] [3] [4], notational complexity [5] [4], redundancy [1] [6] [5] [2], ambiguousness [7] [8]. These defects form notational impediments during development of transformation rules for translating visual models into formal models.

Visualised and Analysed-Model Gaps. Sometimes, modeled process behavior is not accurately transformed into the analysis models. This inaccuracy leads to a state where some different process behavior is analysed instead of the modeled process behavior. This difference refers to *Visualised and Analysed* behavior gap between the two models. In order to *precisely* transform the behavior of visual models into formal models, and ensure that the correct model is being analyzed, the need is to have accuracy in transformation rules.

3. Semantic Preservation Pitfalls

BPMN is a standard specification for high-level visual modeling notation, and offers a variety of semantically diverse options in terms of syntax. Existing transformational rules translate the syntactic definitions of high level models but miss out the semantic translations because the semantics remain abstracted in the visual models as part of language schemas etc.

International Workshop on Petri Nets and Software Engineering 2023, PNSE'23

karnika@cse.iitb.ac.in (Karnika); rkj@cse.iitb.ac.in (Rushikesh K.)
https://www.cse.iitb.ac.in/~karnika/ (Karnika); https://www.cse.iitb.ac.in/~rkj/ (Rushikesh K.)
0000-0001-6490-0380 (Karnika); 0000-0002-2712-1406 (Rushikesh K.)

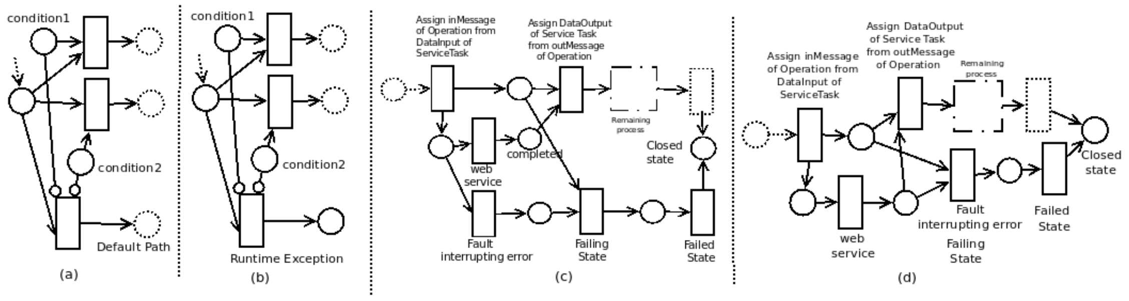

Figure 1: Transformations to preserve semantics of (a) Default Flow for Diverging Exclusive Gateway, (b) Control Flow Behavior Including *Runtime Exception* in a Gateway, (c) and (d) *Faults* in Invoked Services by *Service Task*

Handling the Defaults. BPMN 2.0 [9] provisions *Default flow* as separate notation in visual classification of sequence flows and as options in gateway constructs. However, former is semantically defined as *Sequence-Flows* rather than a separate visual notation, and has selfsame XML schema definition similar to that of a conventional sequence flow, whereas, latter aids modelers to optionally define *default* for gateways. This provision is semantically ingrained as part of XML definition of the gateways, and syntactically available as an elemental attribute in the syntaxes of BPMN gateways. Existing translation strategies are inaccurate in sense of handling these defaults and their semantics. Figure 1(a) illustrates a mapping to incorporate the semantic details for default path from diverging exclusive gateway.

Runtime Exceptions refer to unexpected events or conditions that occur during the execution of a process instance. BPMN defines runtime exceptions at semantic levels, rather than defining them as modeling elements. These runtime exceptions are different from the event exceptions of BPMN 2.0 toolset. For example, a runtime exception is indicated if there is no default path defined in a gateway and none of the conditions are satisfied. BPMN identifies runtime exceptions at *semantic level*. Other examples of semantically defined runtime exceptions are unavailability of *OutputSets*, non-compliance of *InputSet* and *OutputSet* with the associated *IORule*, etc. Existing translation strategies overlook semantics and, ergo, lead to incorrect representation of a process during analysis. Figure 1(b) illustrates an example PN that showcases generation of *runtime exception* when none of the conditions are met in absence of default path definition in a gateway.

***Faults* in Invoked Services.** BPMN specification [9] defines *Service task* that invokes services, and mentions that these services may fail and return *fault* to the invoking process. Figures 1(c) and 1(d) represent transformations of *Service task* such that they include a control flow trace that defines possible faults returned by service. The two transformations differ in consideration of position at which fault generation is expected in control flow trace.

Multiple Instantiation. BPMN permits multiple instantiation through multiple start events, *multiple-start* event, etc., and manages these contexts using mechanism of correlations. However, existing transformation rules miss out these details.

Asynchronous Tasks and Rendezvous. Models need careful transformation to preserve the asynchrony in the model, and also to ensure that new one is not introduced. Likewise, there are synchronization issues, such as in joins. Moreover, while designing for asynchrony and synchronization, instances need to be isolated from each other, and a token from one instance should not be visible in another instance.

Initialization. BPMN 2.0 [9] allows initialization to be managed by various means such as timers, conditions, etc. Management of initialization for control flow of the process is abstracted at the semantic level, irrespective of BPMN element(s) responsible for initialization. Nonetheless, these semantic abstractions are not abstracted at the analysis stage, and transformation rules should incorporate these semantic level initialization abstractions to tokenize the PN for formal analysis and to avoid possible mismatches and semantic gaps defined in Section 2.

Handling Encapsulation. Processes in BPMN are visually encapsulated as swim-pools and swimlanes, and use variations to represent types of encapsulation, such as black boxes for private processes. Preserving the notion of encapsulation while defining transformations is a hidden challenge because, at an outset, it appears to be ignorable. The encapsulation detail is either commonly overpassed or translated conveniently as a swim-pool and swimlane in PNs too [10]. However, encapsulation pertains to implementation, and traceability of encapsulation mapping up to the implementation phase is needed.

Buffer Semantics. Process-oriented languages require mechanisms to manage and deal with physical and informational items, their (data) structures, states, lifecycles, instantiations, associations, etc. BPMN models utilise semantic buffering mechanisms for management of these items and for bringing the modeling phase closer to the implementation phase where amount of data involved is huge. Transformations need to be meticulously defined in order to precisely capture these aspects.

Other Concerns. Conditions can be mapped into places. However, these condition places may not be mapped one-to-one into implementation elements, and they may get scattered. Furthermore, inaccurate transformation rules may introduce race conditions across PNs. Transformation rules need to accurately map the control flow of the process to ensure that neither extra racing is introduced nor required racing is eliminated.

4. Transformational Challenges

This section defines the problems faced by an analyst or a modeler while transforming the visual process models from BPMN into Petri nets for purposes of analysis etc. These are high level challenges encountered by the end users during actual transformation phase.

Incompleteness. BPMN [9] consists of more than eighty five elements in its toolset [4] [3]. Currently, transformational exhaustiveness is still a need. For example, spectrum of tasks, gateways, events, activities, task markers, etc. have not been sufficiently dealt with. Furthermore, these elements show different behaviors as their configurational attributes are varied during modeling stage of processes. These dynamic configurations yield multifold increment in the count of configurations that need to be transformed into PNs.

Inconsistency. There are several formal languages used for transformation of BPMN models. Also, translators follow different transformation conventions which are not standardized, which pose challenges while combining ideas from different transformation schemes. Moreover, multiple transformation rules are provided for same element which may result in ambiguous choices.

Inaccuracy. Modelers face obstacles while mapping the modeled process behaviors because of inaccuracies in the usage of transformation rules. For example, inaccuracies may get typically incorporated upon varying relative positions of the elements in the model.

Reusability. Transformation rules become non-reusable as the use of the BPMN element being transformed changes. This is caused due to incompleteness in exhaustiveness of transformation rules. For example, BPMN's *event to task* transformation is easily assumed as place-transition in PNs, but as the number of outgoing arcs increase, the mapping becomes inaccurate as it produces XOR relationship rather than AND relationship.

Notational Impediments. BPMN 2.0 [9] defines elements, but the definitions contain ambiguities such as in the definition of inclusive OR gateway [7] [8]. Moreover, the richness of notations with heavy rule based interpretations in terms of attributes, schema, operational semantics as part of the BPMN specification pose challenges to accuracy in transformation rules.

5. Conclusions

The paper uncovers the hidden hurdles that exist as absence of preservation of semantics in existing transformation rules for BPMN elements. It also presents challenges faced by modellers during actual transformations for analysis of BPMN models. By shedding light on hidden and overlooked hurdles with an exemplar view and two classes of behavioral semantic gaps, the paper intents to ensure more accurate transformations for analysis of BPMN models in practice.

References

[1] H. Leopold, J. Mendling, O. Günther, Learning from quality issues of bpmn models from industry, IEEE Software 33 (2016) 26–33.

[2] K. Shivhare, R. K. Joshi, Process line diagrams (plds): An approach for modular process modeling, in: Proc. of the 16th Innovations in Software Engineering Conference, ACM, 2023.

[3] I. Compagnucci, F. Corradini, F. Fornari, B. Re, Trends on the usage of BPMN 2.0 from publicly available repositories, LNBIP, Springer, 2021.

[4] M. z. Muehlen, J. Recker, How much language is enough? theoretical and practical use of the business process modeling notation, in: Advanced Information Systems Engineering, Springer, 2008.

[5] S. Anna, Towards a taxonomy of business process and its anomalies, Int. Journal of Computer Science and Network Security 21 (2021) 230–240.

[6] A. Suchenia, T. Potempa, A. Ligęza, K. Jobczyk, K. Kluza, Selected Approaches Towards Taxonomy of Business Process Anomalies, Springer International Publishing, 2017.

[7] F. Corradini, C. Muzi, B. Re, L. Rossi, F. Tiezzi, Global vs. Local Semantics of BPMN 2.0 OR-Join, 2018, pp. 321–336.

[8] F. Corradini, C. Muzi, B. Re, L. Rossi, F. Tiezzi, Bpmn 2.0 or-join semantics: Global and local characterisation, Information Systems 105 (2022).

[9] O. M. Group, Business process model and notation (bpmn) (2014).

[10] A. G. Lazaropoulos, Business education and training during the enterprises' digital transformation: Notation alignment and equivalence rules among the enterprises' business process models, Regional Economic Development Research (2021) 51–70.