VBPMN: Automated Verification of BPMN Processes

Ajay Krishna[1], Pascal Poizat[2,3], and Gwen Salaün[1]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, Inria, LIG, F-38000 Grenoble, France
[2] Université Paris Lumières, Univ Paris Nanterre, Nanterre, France
[3] Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR7606, Paris, France

Abstract. Business process modeling is an important concern in enterprise. Formal analysis techniques are crucial to detect semantic issues in the corresponding models, or to help with their refactoring and evolution. However, business process development frameworks often fall short when it comes to go beyond simulation or syntactic checking of the models. In this paper, we present our VBPMN verification framework. It features several techniques for the automatic analysis of business processes modeled using BPMN, the *de facto* standard for business process modeling. As such, it supports a more robust development of business processes.

Keywords: business processes, BPMN, verification, evolution, tool, process algebra, LNT, labelled transition system, model transformation.

1 Introduction

Mastering business processes has become a central concern in companies and organizations. The modeling of these processes is the first step in order to refine, optimize, or make them evolve while reducing costs and increasing incomes. BPMN is a workflow-based notation that has been published as an ISO standard [11,9], and thus is used widely for business process modeling. Several frameworks have been developed in order to support the development of BPMN processes. They mostly provide modeling, simulation, or execution features. However, but for syntactic checking, these frameworks do not provide any advanced, *i.e.*, behavioral semantics-related, support for analyzing the process models.

In this paper, we present a verification framework, VBPMN, that is freely available for download [1]. It enables one to verify several properties of interest on BPMN processes. VBPMN relies on an intermediate process meta-model called PIF (Process Intermediate Format). This pivot meta-model, and its XML representation, open the way to the use of different process modeling notations as front-end. They also enable us to develop back-end connections to the input languages of several verification tools, and as a consequence to several kinds of verification. For now, we have focused on BPMN as a front-end, and on connection to the CADP verification toolbox [7] using one of the input languages it supports, LNT [3], and the SVL verification scripting language [6]. It is worth

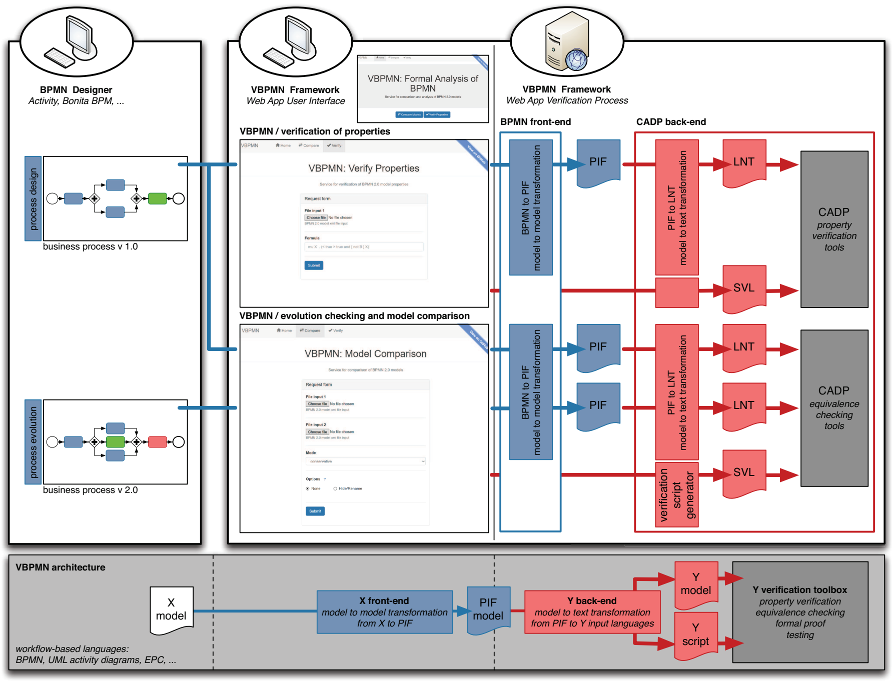
BPMN Designer
Activity, Bonita BPM, ...
process design
business process v 1.0
process evolution
business process v 2.0
VBPMN Framework
Web App User Interface
VBPMN: Formal Analysis of BPMN
VBPMN / verification of properties
VBPMN: Verify Properties
VBPMN / evolution checking and model comparison
VBPMN: Model Comparison
VBPMN Framework
Web App Verification Process
BPMN front-end
CADP back-end
BPMN to PIF model to model transformation
PIF
PIF to LNT model to text transformation
LNT
SVL
CADP property verification tools
CADP equivalence checking tools
verification script generator
VBPMN architecture
X model
workflow-based languages: BPMN, UML activity diagrams, EPC, ...
X front-end model to model transformation from X to PIF
PIF model
Y back-end model to text transformation from PIF to Y input languages
Y model
Y script
Y verification toolbox property verification equivalence checking formal proof testing

Fig. 1. Overview of VBPMN.

emphasizing that other modeling notations can be connected to PIF, *e.g.*, Event-driven Process Chains (EPC) or UML activity diagrams, since they share an important subset of concepts and associated semantics. Complementary back-ends are already under development, concretely, a transformation from PIF to the input language of an SMT solver to support data-flows and data contraints in processes, and a transformation from PIF to Maude to support the quantitative analysis of timed processes.

Figure 1 gives an overview of VBPMN. It comes with a Web application that takes as input BPMN-2.0-compliant business processes. The processes are first transformed into PIF. Then, from the PIF descriptions, models in LNT and model-specific verification scripts in SVL are generated. In the end, CADP is used to check either for functional properties of a given business process or for the correctness of the evolution of a business process into another one. This later kind of verification being supported by VBPMN is particularly helpful in order to improve a process *wrt.* certain optimization criterion.

The rest of this paper is organized as follows. Section 2 introduces the models and languages supported by VBPMN. Section 3 gives an overview of the VBPMN Web application. In Section 4, we focus on the CADP back-end, on the properties it allows one to check, and we present experimental results. Section 5 concludes the presentation and sketches some perspectives for our work.

2 Models and Languages

VBPMN relies on an intermediate format, PIF. It allows one to support several modeling languages, *e.g.*, BPMN, and to target several verification tools, *e.g.*, the CADP toolbox using LNT and LTS formal descriptions. We present here the main models and languages currently supported in the framework.

BPMN. BPMN is an ISO/IEC standard since 2013 [11,9]. It is a workflow-based graphical notation (and an XML-based language) for modeling business processes whose development is supported by many designers and execution frameworks, *e.g.*, Activiti, Bonita BPM, or jBPM. In our work, we focus on the behavioral subset of BPMN which consists of start/end events, tasks, and gateways (exclusive, inclusive, and parallel). We support looping behaviors and unbalanced workflows, that is, gateways without an exact correspondence between split and merge gateways. We only require that BPMN processes are syntactically correct, which is enforced by the aforementioned BPMN designers.

PIF. PIF stands for Process Intermediate Format. We use it as a pivot metamodel and language in order to make our approach generic and extensible. PIF is based on the common constructs one finds in a workflow-based modeling language. The interest of such a pivot language is that several modeling languages can be used as input, *e.g.*, BPMN (that is supported by now), UML, or Event-driven Process Chains. Moreover, several verification techniques and tools can be connected to it as a back-end, *e.g.*, to deal with behavioral properties of models, or with extensions of such properties to time and data-related aspects.

LNT and LTS. We have focused so far on purely behavioral properties. Verification operates on Labelled Transition Systems (LTSs). This low-level model is especially convenient because there are many verification tools accepting this format as input, in particular in the model checking area. A translational semantics from PIF to LTSs was obtained indirectly using a model transformation from PIF to the LNT process algebra. LNT [3] is expressive enough to encode the expressiveness of the PIF constructs and LNT operational semantics maps to LTSs. Further, LNT is the input formalism of the CADP toolbox [7], which provides various kinds of analysis we reuse for formally analyzing the PIF descriptions resulting from our BPMN to PIF model transformation.

Transformations. VBPMN works thanks to several model transformations, as depicted in the generic architecture on the bottom of Figure 1. We first use a model-to-model transformation in order to transform BPMN processes into PIF models. Then, we use a model-to-text transformation for generating from PIF models corresponding LNT specifications as well as CADP verification scripts in the SVL language [6]. These scripts automate the verification selected in the VBPMN Web interface (see Sect. 3 below for more details). Note that when one of the verification steps described in the SVL scripts fails, one gets a witness (*i.e.*, a counter-example) that is presented back in the Web interface so that the designer can use it to modify the erroneous process model.

3 Web Application

Business processes are usually designed by business analysts that may not be familiar with formal verification techniques and tools. Our goal is to enable one to take benefit from formal verification without having to deal with a steep learning curve. The VBPMN Web Application has been developed in this direction. It hides the underlying transformation and verification process, it provides the users with simple interaction mechanisms, and it generates analysis results that are easily relatable to the input process model(s). There are numerous tools supporting the modeling of business processes. Extending a specific one, *e.g.* the Eclipse BPMN designer, would limit the community that could use VBPMN. Therefore, we have decided to architecture it as a Web application.

Technology stack. The VBPMN Web application is hosted on a Tomcat application server. Its responsive UI invokes a RESTful API to trigger the transformation from BPMN to PIF and the verification of the process models. The use of such an API makes the platform more extensible – other people could build custom UIs using them. Internally, the API is built using the Jersey JAX-RS implementation. The model-to-model transformation from BPMN to PIF is realized at the XML level (both BPMN and PIF have XML representations) using a combination of JAXB and of the Woodstox Streaming XML API (StAX), which implements a pull parsing technique and offers better performance for large XML models. The model-to-text transformation from PIF to LNT and SVL is achieved using a Python script that can also be used independently from the Web application as a command-line interface tool.

User interface. One can choose either to verify some property or to check process evolution correctness. In the first case (Fig. 2, left), one has to upload the BPMN process model and specify the temporal logic formula for the property. In the later case (Fig. 2, right), one has to upload two BPMN processes, specify the evolution relation, and optionally give tasks to hide or to rename in the comparison (see [12] for the formal definition of the evolution relations). As a result one can visualize the LTS models that have been generated for the BPMN processes. Further, in case the verification fails, *i.e.*, either the property does not yield or the evolution is not correct, one gets a counter-example model.

4 CADP Back-End

The CADP back-end addresses business process verification using available model-checking and equivalence checking techniques in the CADP toolbox. This is achieved by transforming PIF models into LNT process algebraic descriptions, and by generating specific SVL verification scripts from UI inputs.

From PIF to LNT and SVL. The principle of the PIF to LNT transformation is to encode into LNT processes all the BPMN elements involved in a process model behavioral semantics, that is, all nodes (initial/end events, tasks, and gateways) and all sequences flows between nodes. This gives us a set of

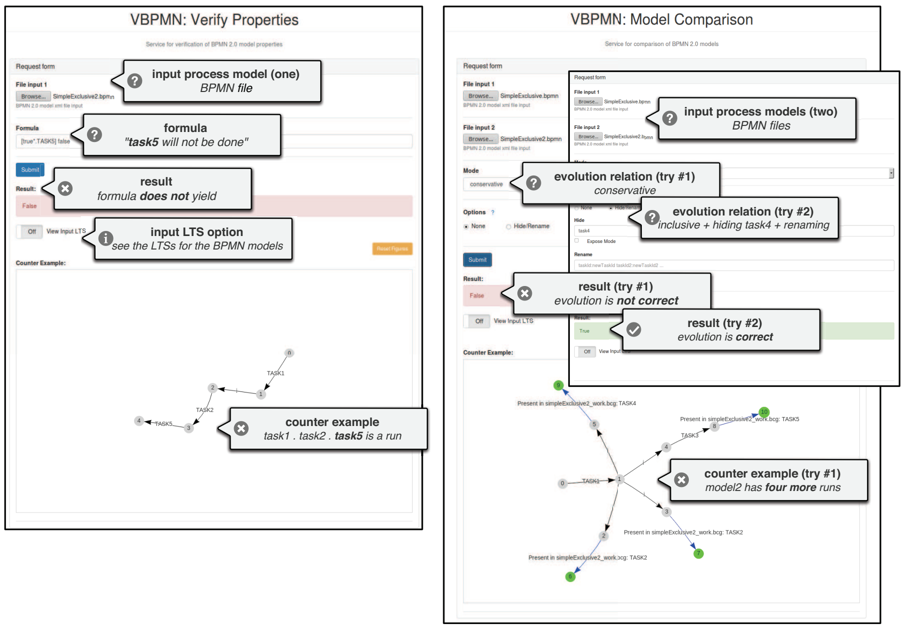

Fig. 2. VBPMN Web application in use.

LNT processes that are then composed in parallel and synchronized accordingly to the BPMN execution semantics. For instance, after execution of a node, the corresponding LNT process synchronizes with the process encoding the outgoing sequence flow, which then synchronizes with the process encoding the node appearing at the end of this flow, and so on. More details on this encoding can be found in [12], which however applied only to balanced process workflows, *i.e.*, workflows where every split gateway of some kind (exclusive, parallel, inclusive) has a corresponding merge gateway of the same kind. This limitation is no longer present in VBPMN that now supports also unbalanced process workflows. This has been achieved by implementing in LNT a scheduler that runs in parallel with all other processes, keeps track of active flows, and interacts with some specific node processes, *e.g.*, those for inclusive merge gateways, in order to indicate them whether they have to expect synchronization with more processes or not.

Verification using CADP. The operational semantics of the LNT process algebra enables us to generate LTSs corresponding to the BPMN process model given in the VBPMN UI. These LTSs may then be analyzed using CADP. VBPMN currently provides two kinds of formal analysis: functional verification using model checking and process comparison using equivalence checking. As far as functional verification is concerned, one can for example use reachability analysis to search, *e.g.*, for deadlock or livelock states. Another option is to use the CADP model checker for verifying the satisfaction of safety and liveness properties.

In these cases, since the properties depend on the input process, they have to be provided in the UI by the analyst, who can reuse well-known patterns for properties such as those presented in [4].

Process evolution takes as input two process models, an evolution relation and possibly additional parameters for the relation. Several evolution relations are proposed. Conservative evolution ensures that the observational behavior is strictly preserved. Inclusive evolution ensures that a subset of a process behavior is preserved in a new version of it. Selective evolution (that is compatible with both conservative and inclusive evolution) allows one to focus on a subset of the process tasks. It is also possible to have VBPMN work up-to a renaming relation over tasks. If the two input process models do not fullfil the constraints of the chosen evolution relation, a counter-example that indicates the source of the violation is returned by VBPMN in the UI. This helps the process analyst in understanding the impact of evolution and supports the refinement into a correct evolved version of a process model. All the evolution relations are checked using the CADP equivalence checker and SVL scripts for hiding and renaming.

Experiments. We used a Mac OS laptop running on a 2.3 GHz Intel Core i7 processor with 16 GB of memory. We carried out experiments on many examples taken from the literature or hand-crafted, and we present in Table 1 some of these results. For each process, the table gives the number of tasks (T), flows (F), gateways (exclusive, parallel, and inclusive, respectively), and two booleans indicating the presence of loops (L) and unbalanced workflow structure (U). The table finally presents the size of the generated LTS in terms of states and transitions (before and after minimization modulo branching equivalence [13]) as well as the computation time for obtaining the LTS model. We recall that one can use this LTS for analysis purposes using model checking available for instance in the CADP toolbox. All the examples presented in the table are compiled into LTS within a few seconds. The main factor of state space increase is the presence in the input process of parallel or inclusive gateways. Those gateways exhibit a high degree of parallelism and the enumeration of all possible executions result in larger LTSs. As far as the computation time is concerned, the number of parallel and inclusive gateways is again the main factor of explosion as shown in the last example of the table, which consists of several nested gateways. The presence of loops can also increase the size of the resulting LTS and of the computation time because this may induce additional executions to be explored.

5 Concluding Remarks

In this paper, we have presented VBPMN, our tool for the analysis of business processes. VBPMN has a particular focus on BPMN since it is a standard, but it may indeed support as input any workflow-based language that can be transformed into the PIF meta-model and language. PIF is used as an intermediate between workflow notations and back-end formal frameworks, *i.e.*, formal models, equipped with associated verification techniques and tools. We have here focused on a transformation from PIF to LTS, which is, in practice, achieved via

Table 1. Experimental results.

Process description	Constructs							LTS (states/transitions)		Gen. time
	T	F	⊗	⊕	◯	L	U	Raw	Min.	
Booking sys.	6	11	2	0	0	√	×	29/29	8/9	6s
Retry sys.	2	8	3	0	0	√	√	21/21	5/6	6s
Leave man.	6	13	3	0	0	√	√	36/36	9/11	6s
Acc. open. (1)	15	29	5	2	2	×	×	469/1,002	24/34	6s
Acc. open. (2)	16	33	5	2	2	√	×	479/1,013	26/37	7s
Software dev.	6	19	7	0	0	√	√	40/42	12/16	6s
Publishing sys.	12	31	7	2	2	√	√	3,038/9,785	32/63	7s
Incident sys.	7	16	5	0	0	×	√	39/41	11/13	6s
Travel org.	6	14	0	0	4	×	√	4,546/6,155	51/77	9s
Lunch pay.	6	24	8	0	0	√	√	54/59	11/16	6s
Hand-craft. (1)	20	38	0	8	0	×	×	577,756/3,388,390	334/1,174	26s
Hand-craft. (2)	20	43	0	6	6	×	×	4,488,843/26,533,828	347/1,450	224s

a transformation to the LNT process algebra and reusing the LTS semantics of LNT. These LTSs can then be analyzed using model and equivalence checking techniques thanks to the CADP toolbox. The overall analysis process provided by VBPMN is fully automated and freely available for download [1].

Related work. To the best of our knowledge, the existing industrial development frameworks for BPMN, such as Activiti or Bonita BPM, do not provide formal techniques for verifying business processes. If we broaden the scope, we can compare to LoLA, ProM, and VerChor.

LoLA can be used to check whether a Petri net satisfies some property, using reduction techniques and state space explicit exploration. It has been applied in various application domains and more specifically to the verification of the BPEL orchestration language, of Web service choreographies, and of business process models, see, *e.g.*, [5]. In comparison to LoLA-based works, VBPMN proposes specific analysis techniques for the verification of business process evolution.

ProM [2] is a platform for the development of state-of-the-art process mining techniques and tools. Process mining can be used to extract knowledge, *e.g.*, under the form of process models, from execution logs. It can also be used to monitor processes and detect deviations. VBPMN does not address mining from logs, and assumes models are given. The techniques we propose for evolution checking are somehow complementary to ProM where evolution can be tackled from a deviation point of view. BPMNDiffViz [10] combines process mining and the concept of edit distance for providing a similarity measure between two processes. On the contrary, VBPMN has a more qualitative vision of evolution using bi-simulations and pre-orders. The extension to quantitative evolution is definitely an interesting perspective for VBPMN.

The VerChor platform [8] aims at analyzing choreographies possibly described using BPMN choreography diagrams. An intermediate format and a transformation to LNT was used there too. However, the focus is complementary: process diagrams and the verification of properties and of evolution in VBPMN, versus choreography diagrams and the verification of choreography-specific properties (synchronizability and realizability) in VerChor. Further, VBPMN supports unbalanced workflows, while VerChor does not.

Future work. Our main perspective is to go beyond control-flow and behavioral analysis of BPMN, and to take into account data-flow and quantitative aspects. We are studying extensibility features for the PIF meta-model and language. Further, we are developing new back-ends from PIF to SMT solvers for data-flow aspects, and to statistical model-checkers for quantitative aspects.

References

1. VBPMN Framework. `https://pascalpoizat.github.io/vbpmn/`.
2. R. P. Jagadeesh Chandra Bose, H. M. W. (Eric) Verbeek, and Wil M. P. van der Aalst. Discovering Hierarchical Process Models Using ProM. In *Proc. of CAISE'11*, volume 734 of *CEUR Workshop Proceedings*, pages 33–40. CEUR-WS.org, 2011.
3. D. Champelovier, X. Clerc, H. Garavel, Y. Guerte, F. Lang, C. McKinty, V. Powazny, W. Serwe, and G. Smeding. Reference Manual of the LNT to LOTOS Translator, Version 6.1. INRIA/VASY, 2014.
4. M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in Property Specifications for Finite-State Verification. In *Proc. of ICSE'99*, pages 411–420. ACM, 1999.
5. D. Fahland, C. Favre, J. Koehler, N. Lohmann, H. Völzer, and K. Wolf. Analysis On Demand: Instantaneous Soundness Checking of Industrial Business Process Models. *Data Knowl. Eng.*, 70(5):448–466, 2011.
6. H. Garavel and F. Lang. SVL: A Scripting Language for Compositional Verification. In *Proc. of FORTE'01*, pages 377–394, 2001.
7. H. Garavel, F. Lang, R. Mateescu, and W. Serwe. CADP 2011: A Toolbox for the Construction and Analysis of Distributed Processes. *STTT*, 2(15):89–107, 2013.
8. M. Güdemann, P. Poizat, G. Salaün, and L. Ye. VerChor: A Framework for the Design and Verification of Choreographies. *IEEE Trans. Services Computing*, 9(4):647–660, 2016.
9. ISO/IEC. International Standard 19510, Information technology – Business Process Model and Notation. 2013.
10. S. Ivanov, A. A. Kalenkova, and W. M. P. van der Aalst. BPMNDiffViz: A Tool for BPMN Models Comparison. In *Proc. of BPMN'15 Demo Session*, volume 1418 of *CEUR Workshop Proceedings*, pages 35–39. CEUR-WS.org, 2015.
11. OMG. *Business Process Model and Notation (BPMN) – Version 2.0.* January 2011.
12. P. Poizat, G. Salaün, and A. Krishna. Checking Business Process Evolution. In *Proc. of FACS'16*, LNCS. Springer. To appear.
13. R. J. van Glabbeek and W. P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *J. ACM*, 43(3):555–600, 1996.