Modeling and Monitoring of Collaborative Processes: A Merger of BPMN and Declare

Mihály Tass[1], Mathias Spezia[1], Jonas Lindner[2], Anti Alman[3], Giovanni Meroni[4] and Andrey Rivkin[1,*]

[1]Technical University of Denmark, Richard Petersens Plads 321, Kongens Lyngby, 2800, Denmark
[2]TU Wien, Erzherzog-Johann-Platz 1/194-3, Vienna, 1040, Austria
[3]Free University of Bozen-Bolzano, via Bruno Buozzi 1, Bolzano, 39100, Italy
[4]University of Insubria, via Ottorino Rossi 9, Varese, 21100, Italy

Abstract

In multi-model Business Process Management, relative strengths of different modeling languages can be combined to achieve a more holistic view of modeling scenarios at hand. For instance, BPMN can be used to model explicit activity flows, while Declare can provide additional overarching rules (or obligations) between flow components. Building on this idea and inspired by realistic business process scenarios, this paper advances towards increased expressiveness in collaborative processes. In particular, we present an open-source modeling and monitoring tool in which partners' processes are represented using sound BPMN models, while the overall collaboration is governed not only by BPMN message flows but also by Declare constraints representing collaborative obligations. For these constraints, we implement a simulation-time monitoring extension that displays the current status of the corresponding obligations.

Keywords

Collaborative processes, Process Modeling, Process Simulation, Hybrid Processes, Multi-Model Processes

1. Introduction

The recently introduced Multi-Model Paradigm for Business Process Management [1] draws from observations on how the business process management field, as well as process mining and business process representations in general have evolved. In particular, there is a notable trend of using multiple interdependent models to represent a single "greater whole". This idea can be mapped to collaborative processes, where each collaboration partner can be viewed as having a model of their internal process, with the complete process emerging when considering models of all partners together with their interactions, such as data exchanges and control flow dependencies.

In this paper, we demonstrate a corresponding modeling and monitoring tool (based on bpmn.io), built on the general principles of the Multi-Model paradigm, as outlined in [1, 2], and leveraging the modeling languages of BPMN [3] and Declare [4]. More specifically, BPMN is used to represent the processes of individual collaboration partners, while Declare complements BPMN message flows by providing additional constructs for modeling collaborative obligations between the control-flow elements of these processes. The resulting approach, referred to as Multi-Model Collaboration (MM-Collaboration) diagrams, is applicable to a wide range of collaborative process scenarios in which users need to model complex collaborations and explore their execution through native token-based simulation extended with monitoring capabilities allowing one to observe the state of all collaborative obligations at runtime.

Proceedings of the Best BPM Dissertation Award, Doctoral Consortium, and Demonstrations & Resources Forum co-located with 24th International Conference on Business Process Management (BPM 2026), Toronto, Canada, September 27th to October 2nd, 2026.

*Corresponding author.
mihta@dtu.dk (M. Tass); matsp@dtu.dk (M. Spezia); jonas.lindner@tuwien.ac.at (J. Lindner); anti.alman@unibz.it (A. Alman); giovanni.meroni@uninsubria.it (G. Meroni); ariv@dtu.dk (A. Rivkin)
0009-0007-5688-3681 (J. Lindner); 0000-0002-5647-6249 (A. Alman); 0000-0002-9551-1860 (G. Meroni); 0000-0001-8425-2309 (A. Rivkin)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-09-28

CEUR-WS.org/Vol-4274/paper-28.pdf

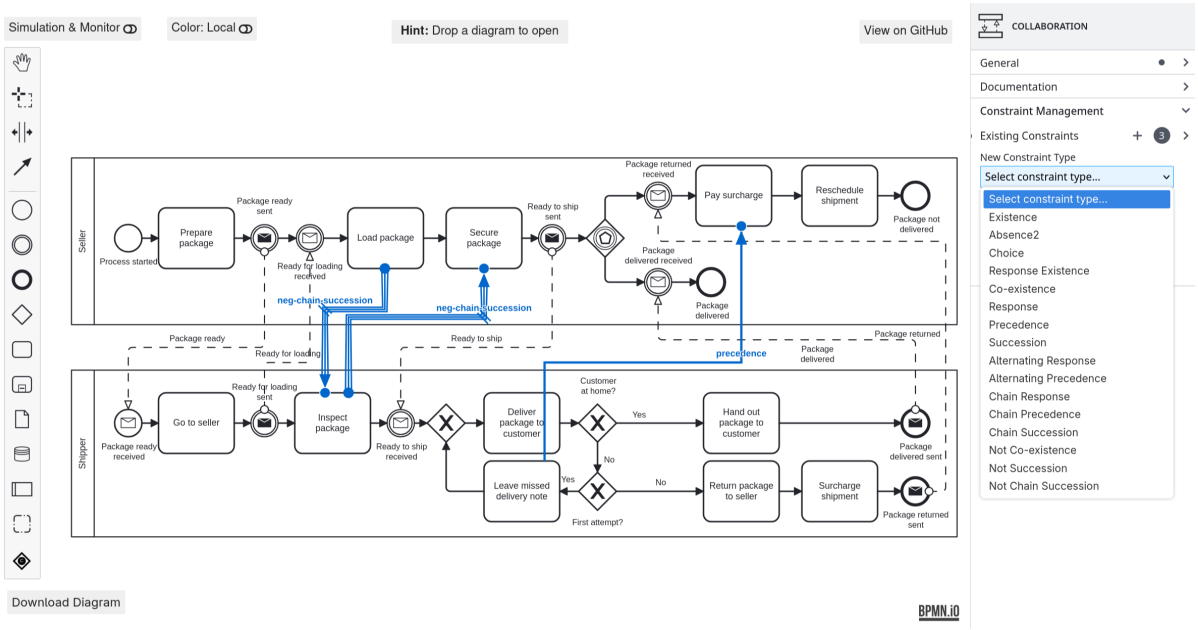


Figure 1: User interface for modeling MM-Collaborations. Currently showing an example MM-Collaboration diagram representing a logistics process.

In the following, Section 2 provides additional details on MM-Collaboration diagrams, Section 3 presents a corresponding modeling and monitoring environment called CoDecBPMN and implemented on top of bpmn.io, Section 4 showcases the availability and maturity of CoDecBPMN based on realistic collaboration examples, and finally, Section 5 concludes the paper by outlining future research directions.

2. Multi-Model Collaborations

Building on BPMN collaboration diagrams [3], an MM-Collaboration diagram consists of a set of BPMN process models, which are concurrently executed following the open-world assumption of declarative process modeling languages. That is, each individual model, when viewed in isolation, is executed according to the control flow prescribed by that model, and any interleaving of different models is allowed unless it causes a conflict with any rules on how these specifications can or must interact. Such rules are then expressed, in MM-Collaboration diagrams, using BPMN message flows and any constraint templates of the Declare language [4].

Consider the MM-Collaboration diagram given in Figure 1. It consists of two BPMN processes (Seller and Supplier) and multiple message flows and Declare constraints between these processes. As an example, the leftmost message flow (Package ready) restricts the process of the Supplier, such that it can only begin when the corresponding message is received from the Seller. Meanwhile, the leftmost constraint (neg-chain-succession) further restricts the collaboration, such that Inspect Package (Supplier) cannot be executed immediately after Load Package (Seller). However, Inspect package can still be executed either before Load Package or after Secure Package in the collaboration, as neither would conflict with the constraints nor the message flows of the overall collaboration.

The choice of BPMN for MM-Collaboration diagrams is motivated by the fact that, besides being the de facto standard for process modeling, it allows each model to be represented in a separate pool, thereby providing an additional degree of separation between the participating processes. Meanwhile, Declare is chosen for expressing additional types of interactions, beyond BPMN message flows, due its global perspective (over all possible system execution traces) and compositional nature [5]. In essence, Declare specifies temporal relations between specific activities, while all other activities may still be executed either without restrictions or according to additional constraints involving those activities.

Finally, the MM-Collaboration diagrams and their realization in CoDecBPMN build upon the ideas

introduced in [1] and are formally grounded in Petri nets [6] and automata theory [7]. CoDecBPMN currently supports only a limited subset of sound BPMN models, corresponding to a fragment of the BPMN subset studied in [6]. The set of supported Declare constraints is comparable to those commonly considered in the literature (e.g., [8]). The full list of supported modeling elements can be found at https://github.com/tassmihaly/hybrid_process_monitoring#Modeling. Further technical details on the formal foundations of MM-Collaboration diagrams are provided in the technical report accompanying the tool at https://github.com/tassmihaly/hybrid_process_monitoring/blob/main/Technical_Report.pdf.

3. Software Implementation

CoDecBPMN is open source and publicly available on https://github.com/tassmihaly/hybrid_process_monitoring. The same link also provides information about the supported Declare constraints, the monitoring features, and a technical report that formalizes the execution semantics of MM-Collaboration diagrams and describes the underlying monitoring approach. The repository is configured to publish the web version of CoDecBPMN via GitHub Pages using GitHub Actions. This version is accessible here: https://tassmihaly.github.io/hybrid_process_monitoring/modeler. A screencast showcasing the CoDecBPMN features is available at the tool repository.

A brief overview of the software architecture, implementation details, and the user interface is provided below.

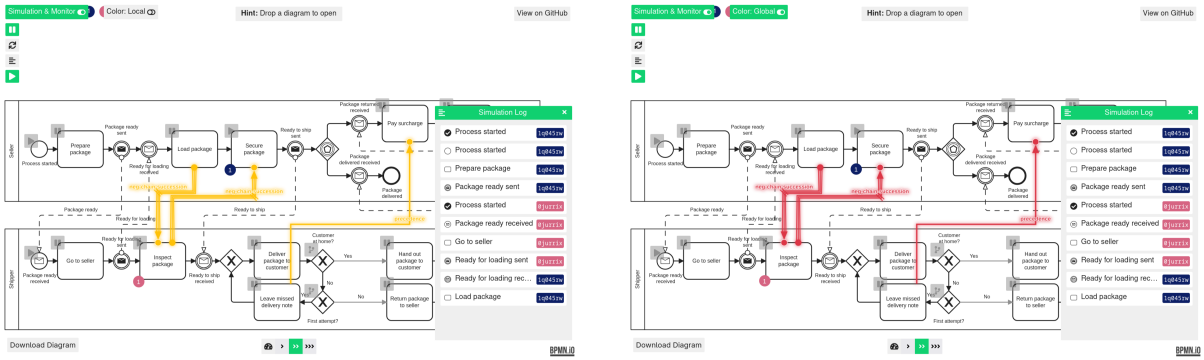
Architecture The modeling and monitoring environment is implemented as a web-based application consisting of two modules: a modeler and a simulator.

The modeler can be used to model an MM-Collaboration from scratch or from an existing BPMN collaboration diagram, which can be easily extended with Declare constraints. These constraints are represented through a custom BPMN extension and managed directly within the modeling interface. The simulator lets the user simulate the MM-Collaboration by relying on the standard BPMN token-based simulation extended with a Declare monitoring (similar to the approach from [8] and discussed in detail in the technical report), showing, in realtime, when Declare constraints are satisfied or violated.

When simulation starts, the monitoring component is responsible for generating a finite state automaton required by the simulator to evaluate Declare constraints. To do so, it takes as input a BPMN collaboration diagram extended with Declare constraints, representing the MM-Collaboration being modeled, and transforms them into a colored Deterministic Finite Automaton (DFA), following an automata-based approach as described in the technical report. This transformation is performed entirely in the browser. The resulting colored DFA is then used by the simulator to evaluate the satisfaction status of all involved Declare constraints at runtime while performing token-based simulation. This, in turn, allows to implement additional simulation and execution modalities, such as automatically terminating the simulation when one of the constraints becomes `Violated`.

Implementation Details The modeler is implemented as an extension of https://bpmn.io: a web-based modeling environment for BPMN. For the simulator, we extended the BPMN token simulator extension of bpmn.io (see https://github.com/bpmn-io/bpmn-js-token-simulation), which allows for both manual execution and (semi-)automatic simulation of BPMN models. To serialize Declare constraints together with BPMN elements, we adopted the BPMN XML format, introducing (`constraint:constraintFlow`) as a custom element to the BPMN XML schema. This also allows our tool to export and import BPMN models created with third-party tools.

The DFA construction relies on an intermediate conversion of BPMN into Petri Nets. At the start of simulation, we take as input the MM-Collaboration model serialized as a BPMN XML. Each pool within the BPMN collaboration is considered a single BPMN diagram and is converted into a Petri Net. Then, it converts the Petri Net into a set of DFAs. Finally, it combines the DFAs with the Declare constraints to produce one colored DFA, which is used as our monitor during simulation. The aforementioned Petri net and DFA transformations are implemented using the PM4JS library (see https://www.pm4js.org/); coloring of the DFA follows the algorithm in the technical report.



(a) Simulation results with local coloring

(b) Simulation results with global coloring

Figure 2: User interface for simulating and monitoring MM-Collaborations

User Interface When the user accesses the web-based user interface, the tool is in design mode. Its interface, as shown in Figure 1, is not much different from the one of bpmn.io. An example diagram (the one shown in Figure 1) is displayed on a virtual canvas, which can be edited by using the toolbar on the left side of the interface and the property panel on the right side of the interface. A model can be saved on the local device by clicking the Download Diagram button on the bottom side of the interface. An existing model can be opened by dragging its corresponding file on the virtual canvas.

The main changes with respect to the stock bpmn.io modeling environment is the presence of one additional button on the toolbar, which can be used to add DECLARE constraints to the model similarly to how message flows are added in bpmn.io. To add a DECLARE constraint, one has to click the toolbar button, click on the source task, and finally click on the destination task. When a DECLARE constraint is selected, the panel on the right displays a property named Constraint Type, which can be used to change the type of constraint (e.g., from response to choice). In addition, one can add DECLARE constraints using the *Constraint Management* panel, as shown on the right in Figure 1, by selecting the new constraint type as well as source activity and target activity. The panel also lists all Existing Constraints.

To switch from design mode to simulation mode, one has to click the *Simulation & Monitor* button on the top side of the interface. When doing so, the colored DFA is generated. When in simulation mode, as shown in Figure 2, the user interface is almost identical to the one of the BPMN token simulator extension of bpmn.io. To start the simulation, one can click the play button on the left side of the interface. Then, by clicking on the process elements, it can control how the process runs (e.g., clicking on an exclusive gateway changes the branch taken).

The main changes with respect to the stock bpmn.io simulation environment are the presence of DECLARE constraints and how their validity is represented. During simulation, each constraint receives a color. The colors change as the process progresses, based on the color assigned to each constraint at a certain state in the automaton. The user can choose between two different color settings, that can be selected over the *Color* button on the top left of the UI. *Color: Local* (default) colors each constraint based on local evaluation, following the coloring as presented in [9], as shown in Figure 2a. *Color: Global* colors all constraints in the same color, as shown in Figure 2b, corresponding to a global evaluation of satisfaction or violation of the process run, following the algorithm in the technical report.

4. Maturity and Availability

The software implementation is under active development at the time of writing. Nevertheless, all core functionalities described in this paper are fully implemented and have been exercised extensively on various MM-Collaboration examples, including the one shown in Figure 1. The modeling environment builds on established BPMN-based tooling, and, thus, the extensions for support for DECLARE constraints and additional UI features can be considered stable. Similarly, the simulation capabilities rely on the

same underlying foundations. Furthermore, the correctness of the automata construction for each supported Declare constraint template has been manually verified, and the implementation relies on the formal execution semantics available in the technical report. The underlying DFA construction leverages PM4JS for an intermediate Petri net construction, while the remaining functionality is provided by the presented implementation. In addition, the theoretical foundations of the approach are supported by detailed formalizations of both the automata construction and the execution semantics, accompanied by proofs, providing a high degree of confidence in the conceptual correctness of the framework (more details on that are in the technical report). Although no systematic user testing has been carried out yet, the tool has been applied to a range of representative modeling and simulation scenarios, providing confidence in its practical applicability within the scope considered in this work.

5. Conclusion

In this paper, we introduced a modeling and enactment environment for a novel type of collaborative processes, referred to a s MM-Collaborations. In doing so, we combined BPMN and Declare, with the former representing internal processes of collaboration partners and the latter enabling additional expressiveness for modeling the interactions between these processes. This implementation will serve as a testbed for future extensions of MM-Collaborations, including the exploration of different execution modalities (e.g., optional constraints) and recovery strategies (e.g., resetting violated constraints).

Declaration on Generative AI

During the preparation of this work, the authors used generative AI tools in order to check grammar and spelling and improve phrasing. After using this service, the authors reviewed the content and take full responsibility for the publication's content.

References

[1] A. Alman, F. M. Maggi, S. Rinderle-Ma, A. Rivkin, K. Winter, Towards a multi-model paradigm for business process management, in: CAiSE, volume 14663 of *LNCS*, Springer, 2024, pp. 178–194.

[2] F. M. Maggi, A. Alman, P. H. Wittlinger, From constraint-based process modeling to framed autonomy: A historical excursus, in: Mining a Scientist's Process, volume 16480 of *LNCS*, Springer, 2026, pp. 199–211.

[3] OMG, Business Process Model and Notation (BPMN), Technical Report, Object Management Group, 2014. URL: https://www.omg.org/spec/BPMN/2.0.2/PDF.

[4] M. Pesic, H. Schonenberg, W. M. P. van der Aalst, DECLARE: full support for loosely-structured processes, in: EDOC, IEEE CS, 2007, pp. 287–300.

[5] A. Alman, C. Di Ciccio, F. M. Maggi, M. Montali, H. van der Aa, RuM: Declarative process mining, distilled, in: BPM, volume 12875 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 23–29.

[6] R. M. Dijkman, M. Dumas, C. Ouyang, Semantics and analysis of business process models in BPMN, Inf. Softw. Technol. 50 (2008) 1281–1294.

[7] G. De Giacomo, R. De Masellis, M. Montali, Reasoning on LTL on finite traces: Insensitivity to infiniteness, in: AAAI, AAAI Press, 2014, pp. 1027–1033.

[8] G. De Giacomo, R. De Masellis, F. Maria Maggi, M. Montali, Monitoring constraints and meta-constraints with temporal logics on finite traces, ACM Trans. Softw. Eng. Methodol. 31 (2022) 68:1–68:44.

[9] G. De Giacomo, R. De Masellis, F. M. Maggi, M. Montali, Monitoring constraints and metaconstraints with temporal logics on finite traces, ACM Transactions on Software Engineering and Methodology (TOSEM) 31 (2022) 1–44.