Data Objects with Variables in BPMN

Maximilian König[1,*], Tom Lichtenstein[1], Anjo Seidel[1] and Mathias Weske[1]

[1]Hasso Plattner Institute, University of Potsdam, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

Abstract

Managing the creation and manipulation of data is critical in today's organizations. This is reflected in the emergence of object-centric business processes, whose execution is driven by data objects. Therefore, it comes as a surprise that current activity-centric modeling languages such as BPMN lack comprehensive data modeling capabilities, especially when different data objects of the same class have to be processed. This paper proposes to extend BPMN process diagrams with variable identifiers that allow to model precise referencing behavior for different objects within a single process instance.

Keywords

BPMN, Data in Processes, Object-centric Processes, Variables

1. Introduction and Motivation

An important aspect of business process management is helping organizations to maintain an overview of the complex processes that drive their value creation. For that purpose, a wide variety of methodologies is provided to support the business process lifecycle, from design and verification to implementation, monitoring, and evaluation [1]. Recently, the management of data objects in business processes received significant attention, leading to the emergence of object-centricity as a novel paradigm [2]. Instead of focussing on the order of activities and events based on control flow, processes are considered from the perspective of the involved data objects, which represent business data manipulated in process executions [3, 4].

Traditional, well-established activity-centric process modeling languages, such as BPMN process diagrams [5], struggle with the representation of complex data behavior in interaction with the control flow [6]. Hence, the question arises whether this deficiency can be overcome with concepts from the currently evolving field. While version 2.0 of the BPMN standard [5] introduced concepts that support modeling relevant data objects and their impact on process flow, they mainly handle single objects and have imprecise semantics.

In a BPMN process diagram, data object nodes, denoted by a document shape, visualize the existence of certain types of data. Each reference specifies a data class and a state denoted in square brackets. The data class defines a set of objects with the same structure or of the same type, while the state defines an abstraction for the expected data. BPMN does not provide a means to define either data classes or states in more detail. Hence, related approaches often rely on supplementary data models to explicate data classes with attributes and relations between them [6, 7, 8].

To reference several data objects of the same class in the same state, a data object node may be annotated with a multi-instance marker (**|||**), representing a *collection* of data objects. Data objects and collections can be read and written by activities. Read operations are visualized in the model through so-called data associations from data object nodes to activities, and write operations through associations from activity to data object node. An example is shown in Fig. 1. Read operations imply that at least one data object in the specified state must exist before an activity is enabled. Write operations

Proceedings of the Best BPM Dissertation Award, Doctoral Consortium, and Demonstrations & Resources Forum co-located with 22nd International Conference on Business Process Management (BPM 2024), Krakow, Poland, September 1st to 6th, 2024.

*Corresponding author.

maximilian.koenig@hpi.de (M. König); tom.lichtenstein@hpi.de (T. Lichtenstein); anjo.seidel@hpi.de (A. Seidel); mathias.weske@hpi.de (M. Weske)

0000-0002-2244-1179 (M. König); 0000-0001-5585-1003 (T. Lichtenstein); 0000-0002-9652-5340 (A. Seidel); 0000-0002-3346-2442 (M. Weske)

update the referenced object/collection or, if none exists, create a new one. An important assumption in the BPMN specification is that a data object node always refers to the same data object or data object collection per process instance [5, p. 206], which makes it impossible to access different objects of the same type.

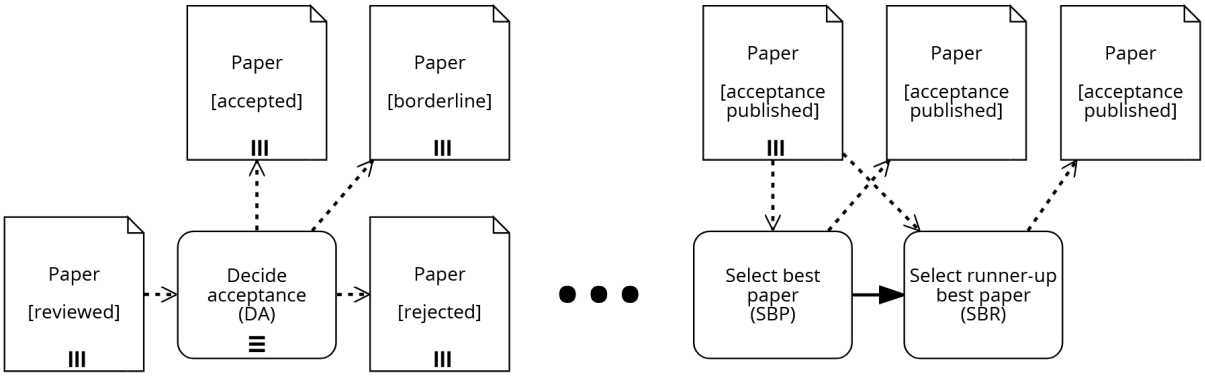

Figure 1: Two excerpts from a paper selection process for a conference.

The impact of that assumption is exemplified in Fig. 1. On the right, the best paper and the runner-up are singled out from a collection of accepted papers to individually notify the authors later on. However, due to the assumption that data object nodes always reference the same objects at runtime, the second of these two subsequent write operations overwrites the reference to the first object. Hence, only the runner-up paper can be accessed in the process model later on, while the reference to the best paper is lost.

Furthermore, when processing collections of data objects under this assumption, activities can only transition all objects that are in the same state into the same target state. For example, when deciding on the acceptance of a collection of submitted papers as shown in Fig. 1, the intended behavior is that only some papers are accepted while others are rejected or require further discussion. However, this behavior cannot be captured with traditional BPMN. Essentially, splitting and merging collections of objects is not possible.

In the following, this paper aims to overcome the presented deficiencies by extending BPMN with variables to model dynamic references to different objects of the same class.

2. Related Work

There exists a large research corpus on the integration of business processes and data that cannot be covered holistically in this position paper. In the following, we will therefore highlight works on BPMN extensions improving its data modeling capabilities and a selection of process modeling approaches with an emphasis on data.

Meyer et al. present an extension to BPMN to capture object relations by denoting foreign key relations between objects of different classes on data object references [6]. Another work introduces a translational semantics for dealing with multiple object collections in BPMN, using the state as an additional identifier next to the data class [9]. Combi et al. [7] employ an SQL-based approach to link BPMN activities to the data classes they operate on. However, in all three approaches, the identification of different objects of the same class was either not tackled or limited to objects' states. Ghilardi et al. introduce a supplementary SQL-based language to explicate data access in BPMN models [10] with a focus on the verification of the resulting compound models.

To improve the integration of processes and data, object- and data-centric process modeling notations have been proposed in the literature. Steinau et al. provide a comprehensive overview [11]. After the publication of their paper, additional approaches have been introduced, including object-centric Petri nets [3], object-centric behavioral constraints [12], synchronous proclets [4], and object-centric

Petri nets with identifiers [13], all covering different subsets of the object-centric modeling features as introduced by Gianola et al. [13]. Compared to these formal approaches, BPMN has the advantage of being well-understandable and widely adopted in industry [14]. Therefore, we decided to extend BPMN's data modeling capabilities to better capture data behavior.

3. Introducing Variables to BPMN

To address the limitations of the current data semantics of BPMN as outlined above, we extend data object nodes with *variables*. In essence, a variable represents an identifier denoted on a data object node that is bound to one specific data object at runtime. Therewith, the variable can be used to access the assigned data object again in later steps of the process. Furthermore, variables can be reassigned to a different data object of the same class during process execution. Hence, by using variables, we can precisely model the referencing behavior for multiple data objects of the same class, eliminating the need for data object nodes of a class to reference the same data object throughout process execution.

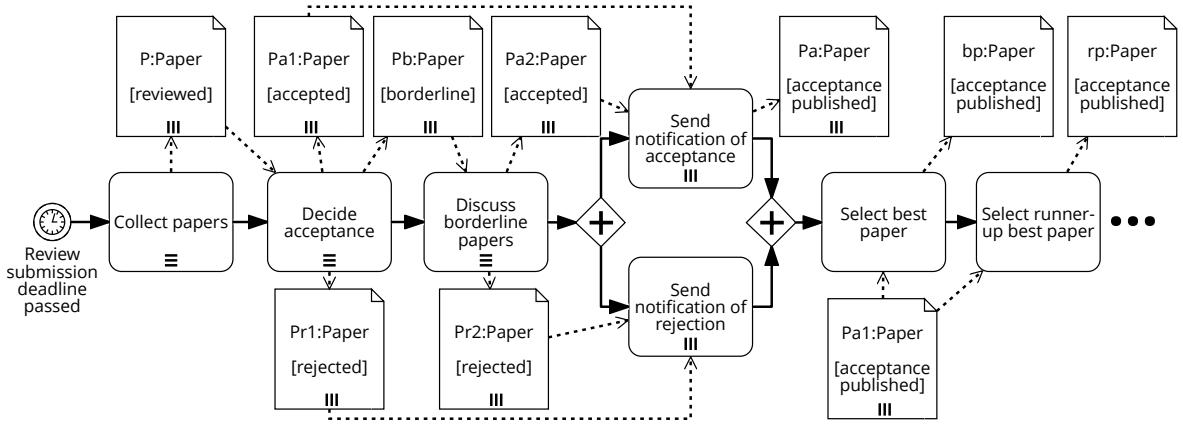

Figure 2: Paper review process enriched with variables.

Fig. 2 depicts the paper review process extended with variables. We specify variables in the labels of data object nodes as a prefix to the associated class separated by a colon, e.g., 'P:Paper', where 'P' is the variable for data objects of the class 'Paper'. As a convention, we use uppercase letters, e.g., 'P:Paper', to refer to collections of data objects, and lowercase letters, e.g., 'bp:Paper', to refer to individual data objects. Given the example, using variables allows distinguishing between the data objects written by the activities 'Select best paper' and 'Select runner-up best paper', even though both are of class 'Paper'. Furthermore, variables enable activities to return multiple data objects of the same class in different states, since each output can still be distinguished. Considering the example in Fig. 2, 'Decide acceptance' can create three collections of the class 'Paper', each containing data objects in a different state which can later be identified via the respective variable: 'Pa1', 'Pr1', and 'Pb'. As a consequence, variables effectively address the shortcomings outlined in Section 1.

In the following, we elaborate on the inclusion of variables in process diagrams by discussing their impact on create, read, and update operations.

Create. In the original semantics of BPMN, if an activity writes to a data object node without reading from a node of the same class, a data object is either created or blindly overwritten. Since variables support distinguishing multiple data objects of the same class, we modify the semantics to always create a new data object in this case. Consequently, blind writes are no longer supported. The created data object is assigned to the variable associated with the data object node. This may include the reassignment of an already assigned variable. Similarly, the creation of a data object collection assigns all created data objects to the corresponding variable. Considering Fig. 2, 'Collect papers' creates a collection of papers in 'reviewed' that all are assigned to the variable 'P' for later reference.

Read. By associating data object nodes with variables, we limit the scope of read operations to the data object assigned to the corresponding variable. This ensures that multiple successive reads of the same variable will access the same data object. However, reading a data object from a variable requires (1) an existing assignment and (2) that the assigned data object satisfies the class and state constraints as defined in the process model. Otherwise, the read cannot be performed, thus blocking the execution of the activity.

When reading data object collections, all data objects are accessed that are assigned to the variable and satisfy the state constraints of the node. If an activity reads multiple collections of the same class, the collections are merged before being processed by the activity. For example, in Fig. 2, 'Send notification of acceptance' reads the union of the collections 'Pa1' and 'Pa2'. Similar to single data object reads, an assignment to a collection must exist for all read variables, and the union of all collections of the same class and state must contain at least one suitable data object.

Update. Updates to data objects can only be achieved by reading and writing a data object of the same class, as blind updates are no longer supported. An updated data object is assigned to the variable of the reference that is written to. If different variables were used to read and write the data object, both variables will refer to the same data object after the update, even if the state of the data object has changed. For example, 'Send notification of acceptance', updates the state of all data objects associated to 'Pa1' and 'Pa2' to 'acceptance published' and assigns the results to 'Pa'. Nonetheless, 'Pa1' and 'Pa2' keep their references to the data objects. Therewith, we can specify that the best paper and the runner-up best paper can only be selected from the papers that were accepted in the first round, which were stored in collection 'Pa1'.

Inspired by [9], when updating collections, we do not require that every data object in the collection receives the same update: The activity 'Discuss borderline papers' updates the state of each data object in 'Pb' to either 'accepted' or 'rejected', effectively splitting the initial collection into 'Pa2' and 'Pr2'. The decision on how collections are split is delayed to runtime, providing flexibility in execution. It should be noted that a split may result in an empty collection. For example, 'Pb' might be empty if no paper is considered borderline. Empty collections are still assigned to the corresponding variable.

Finally, reading and writing to data objects of the same class without changing the state allows for copying references to different variables. Given the example in Fig. 2, 'Select best paper' copies a reference of the collection 'Pa' to one data object 'bp' for future use. In this case, the data object itself is not changed.

4. Implications and Challenges

With the introduction of variables to BPMN data object references, we improve BPMN's expressiveness regarding the representation of multiple data objects of the same class. Such data objects can now be explicitly referenced and independently processed in activity instances. Applying the concept to data object collection references eliminates the limitation that state transitions always apply to all objects in a collection. Instead, each element can individually be updated to any of the referenced output states. Together, this overcomes the deficiencies depicted in Fig. 1. The collection of submitted papers can be split into 'accepted', 'rejected', and 'borderline' papers, and the best paper and runner-up can be selected and further processed individually, as illustrated in Fig. 2.

Besides the advantages, the extension entails a number of challenges. So far, it is possible to create variables referencing collections that contain data objects in different states, as well as collections partly referring to the same data objects. While that behavior is supported, it may lead to unintuitive behavior which requires further investigation. Similarly, empty output collections can lead to deadlocks: Given the example in Fig. 2, if all papers are accepted, 'Send notification of rejection' cannot be executed, resulting in a deadlock. Furthermore, variables must be assigned before being read, which is an additional aspect model designers have to consider. To overcome this, it is essential to develop guidelines for the use of variables and to explore methods for the automatic detection of potentially erroneous behavior

through model checking.

Another aspect that may be addressed by using variables in BPMN is object correlation, i.e., the association of related objects within a process instance. For example, if the decision on a paper's acceptance used a collection of reviews as additional input, it should be possible to identify which reviews were created for which paper in the scope of the variables. While some object-centric approaches already cover this aspect [13], support in BPMN is still lacking. A starting point for investigation might be the foreign key relations introduced in [6].

So far, the description of the semantics for the novel object variables remains on a conceptual level. Future studies should investigate suitable formalisms to concisely define the intended behavior.

5. Conclusion

BPMN process models currently are insufficiently specified to capture complex data behavior involving multiple objects of the same class. Therefore, this position paper motivates the introduction of variables for data object nodes to enable individual access to and processing of different objects of the same class. In addition, the implications of such an extension as well as entailed challenges are outlined, providing a starting point for future research endeavors.

References

[1] M. Weske, Business Process Management - Concepts, Languages, Architectures, Fourth Edition, Springer, 2024.

[2] W. M. P. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: Software Engineering and Formal Methods - 17th International Conference, SEFM 2019, Proceedings, volume 11724 of *LNCS*, Springer, 2019.

[3] W. M. P. van der Aalst, A. Berti, Discovering object-centric petri nets, Fundam. Informaticae 175 (2020) 1–40.

[4] D. Fahland, Describing behavior of processes with many-to-many interactions, in: PETRI NETS 2019, volume 11522 of *LNCS*, Springer, 2019, pp. 3–24.

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

[6] A. Meyer, L. Pufahl, D. Fahland, M. Weske, Modeling and enacting complex data dependencies in business processes, in: BPM 2013, volume 8094 of *LNCS*, Springer, 2013, pp. 171–186.

[7] C. Combi, B. Oliboni, M. Weske, F. Zerbato, Conceptual modeling of inter-dependencies between processes and data, in: SAC 2018, ACM, 2018, pp. 110–119.

[8] M. Hewelt, M. Weske, A Hybrid Approach for Flexible Case Modeling and Execution, in: M. L. Rosa, P. Loos, O. Pastor (Eds.), BPM Forum 2016, volume 260 of *LNBIP*, Springer, 2016, pp. 38–54.

[9] M. König, M. Weske, Multi-instance data behavior in BPMN, in: ER Forum 2023, volume 3618 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023.

[10] S. Ghilardi, A. Gianola, M. Montali, A. Rivkin, Delta-bpmn: A concrete language and verifier for data-aware BPMN, in: BPM 2021, volume 12875 of *LNCS*, Springer, 2021, pp. 179–196.

[11] S. Steinau, A. Marrella, K. Andrews, F. Leotta, M. Mecella, M. Reichert, DALEC: a framework for the systematic evaluation of data-centric approaches to process management software, Softw. Syst. Model. 18 (2019).

[12] W. M. P. van der Aalst, A. Artale, M. Montali, S. Tritini, Object-centric behavioral constraints: Integrating data and declarative process modelling, in: Workshop on Description Logics 2017, volume 1879 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017.

[13] A. Gianola, M. Montali, S. Winkler, Object-centric conformance alignments with synchronization, in: CAiSE 2024, volume 14663 of *LNCS*, Springer, 2024, pp. 3–19.

[14] M. Dumas, D. Pfahl, Modeling software processes using BPMN: When and when not?, in: Managing Software Process Evolution, Springer, 2016.