Business Processes Dataflow Low-Code Solution: From Modeling to Execution
Literature review

No Author Given

No Institute Given

1 Objectif

In this document, we will mention the state of the art that we could not include in the original paper due to space constraints. This report contains various BPMS and low-code platforms that utilize BPMN to execute process models. Additionally, it includes recent research that considers the integration of process and data within the same process model.

2 Requirement Analysis and Related Work

In this section, we present the state of the art that we conducted to compare various academic works and tools for modeling BPMN and to identify the limitations of these research works regarding data modeling within BPMN.

2.1 Research works

Table 2.1 identifies conceptual limitations (L1-L5) of BPMN, that hinder model understandability, usability, and quality, requiring a well-structured approach to address these issues and provide clearer guidance for developers. These limitations are intended to demonstrate executable data within BPMN using low-code solutions. L1-L2 are mentioned in [4,3,8]. We draw inspiration for L4 and L5 from [10] and the low-code solutions [6]. L6 is also discussed in [4,10].

	L1	L2	L3	L4	L5	L6
Delta-BPMN[4]	+	+	+/-	-	-	+
Activity View[3]	+	+/-	-	-	-	-
DF-BPMN[8]	+	+	+	+/-	-	-
Linking Data and BPMN[5]	+	+	-	-	-	-
Complex Data Dependencies[7]	+	+/-	-	+/-	+	-
data-aware BPMN[1]	+	+	-	-	-	+
dapSL [2]	+	+/-	+/-	-	-	-

Table 1. Requirements coverage (covered +, partially (+/-), not -)

L1. BPMN data stores are underspecified. In BPMN process models, data stores represent enduring persistent data [9], yet they lack comprehensive information regarding the conceptual architecture of a database. All of the listed approaches agree on the

need for interaction with the data stores, but there are clear differences in how this support is provided. Delta-BPMN [4] defines new properties on activities and accessing data stores. Activity View [3] proposes a new extension to bridge the gap between process and data modeling, with a focus on databases, by presenting the interaction using a tabular view. DF-BPMN [8] proposes a new graphical notation to add the data store objects and their attributes graphically within the activity. In [5], the authors define the data within class diagram that should be the database within the execution phase(+). In Complex Data Dependencies[7], The authors extends the data object to present the database object and their interaction within the BPMN. In data-aware BPMN [1], the authors define the persistent data as data store. they define catalog and repository data, for read-only relations and for read-write relations(+). In dapSL [2] the SQL queries allow us to know what are the database objects that we need to accessed and modified within a specific activity.

L2. Interaction between data instances is not clear. Data objects, as defined by the BPMN specification, symbolize volatile process data and are linked to activities through associations [9]. While the approaches mentioned above all involve interaction with the data stores, they also support interaction between the instances, except Activity View. In the latter, only half of L2 is resolved by presenting the relations and cardinalities of the data classes, but it does not present the interaction between the data store and the data object (+/-). Delta-BPMN proposes a new property called effects that can update a variable's value for the data object or using SQL queries for the data store (+). Additionally, DF-BPMN explicitly presents data relations using data flow between inputs and outputs(+). In [5] the authors are define an "Artifact" class that contains all the information's of the process variable which is describe the process variables. In complex data dependencies, the data object are used to present the life cycle of each object. However, the process variables are not used in this process model (+/-). In data-aware BPMN, the authors are define formally the relation between the violation data (data object) and persistent data (data stores). In dapSL, the authors define the relations between database table, but the data objects (process varaibles) are not used within these model.

L3. Data from/to external environment are not supported. External environments, such user interfaces and services, are frequently interacted with by information systems. For modeling and development to be effective, it is imperative that the relationship between process activities and these external resources be visually represented. In terms of the injection with user variables, Delta-BPMN introduces new variables, denoted as "var," to define user variables as properties within the activity. However, users still have to add these properties to the activity without any graphical presentation (+/-). The Activity View primarily focuses on database information and neglects user data (-). In DF-BPMN, a data type called "user" is defined, allowing for inputs graphically defined within the model to address these limitations (+). Similar to activity-view, linking data and BPMN, and complex data dependency, and data-aware BPMN, the authors focuses on the data within the process without take into account the data from the user(-). In dapSL the data from the user are the function's input that need to be filled within this framework. This data are filled after the integration of this framework to another BPMS engine. But, this data still coded and not support fully visal user data (+/-).

L4. Data operations are ambiguous. Achieving a milestone in an activity instance implies successful data instance attainment (if any), but BPMN's lack of explicit data instance representation can lead to ambiguity and confusion, especially when tracking data instances is critical for understanding process flow and reducing errors during implementation. Data milestones are reached when operations are well-defined, encompassing CRUD operations and complex operations like conditions and aggregations. Looking at the data operations, the research works such as Delta-BPMN, and Activity View do not support this functionality(-). In DF-BPMN, the authors mention that they define data processing operators to present complex operations, but it is still a symbol without any practical implementation (+/-). In linking data and process, all the operations are implicitly presented within the OCL operation(-). In complex data dependencies, the authors are well define the CRUD operation using an annotation added to the data object (+/-). In dapSL, all the operations are implicitly presented within the SQL queries(-).

L5. Hard-coding of data operation. Despite the implementation of the control flow of a process and the allocation of the required resources based on a given process model, data operations are still hard-coded by developers to execute the process model. This can lead to errors in execution semantics if the developer misunderstands the process requirements. Therefore, automation and code generation for data operations should be the preferred solution to reduce errors in achieving the desired process goals during execution. Research models such as Delta-BPMN, Activity View, and DF-BPMN are designed solely for the design phase, lacking the capability to execute these models (-). In linking data and process, the authors propose to convert the OCL operation to logic derivation rules and the BPMN diagram to a petri net. However, this method still need the hard code using OCL operations (-). In complex data dependencies, the patterns (data annotation) are converted to SQL code, and this model can be executable within Activiti(+). In deta-aware BPMN, the model is a formal model and does not support the execution of the process model (-). The dapSL can be integrated within a BPMS, and they define an API to do that within DAPHNE engine, so the developer need write codes to enact the data and process model (-).

L6. Model Verification. Verification corresponds to the task of determining whether a process model is compliant with a specified set of correctness criteria [10]. There are different frameworks that integrate data and processes, all detailed in the document above. delta-BPMN supports the verification of the model using MCMT (+). Also, data-aware BPMN support the verification using the SMT (+). While the other works do not support verification methods (-).

2.2 BPMS & Low-code platforms

In this section, we will present the requirement analysis of executing BPMNs and LCDPs. Since these tools are based on BPMN, they have the same limitations as BPMN in presenting persistent data (data stores) and volatile data (user data).

Table 2.2 represents the limitations that we used to analyze most of the BPMS and Low-Code platforms. These platforms utilize BPMN as a process diagram to define the business process and we examine how these tools define the data within this process.

	L1	L2	L3	L4	L5	L6
Bonita	+	+	+/-	+	-	+
Activiti	+	+	+/-	+	-	+
Camunda	+	+	+/-	+	-	+
Mendix	+	+	+/-	+	-	-
flowable	+	+	+/-	+	-	+
Bizagi	+	+	+/-	+	-	-
Appian	+	+	+/-	+	-	-
Nintex	+	+	+/-	+	-	-
Pega	+	+	+/-	+	-	-
Outsystems	+	+	+/-	+	-	-

Table 2. Requirements coverage of BPM tools (covered +, partially (+/-), not -)

L1. How do they support the connection to the database? In Bonita[1], they support the data as Business Data Model (BDM) and then define a business variable to be used in the model. Which usually used within application, and they integrate using REST APIs and JDBC connectors to the database. Mendix[2] supports integration with databases via its Database Connector module, allowing connections to various database systems. Activiti[3], Flowable[4], and Camunda[5] provide connectors for common database systems and support custom database configurations through Java APIs. These platforms enable database interaction through service tasks and script tasks within BPMN processes. Bizagi[6] supports database connectivity through its Data Layer feature, allowing integration with various database systems. It provides tools for mapping database entities to process data objects and executing SQL queries within process models. Appian[7] offers integration with databases through its Data Store feature, supporting connections to relational databases and other data sources, which allow you to insert, update, query, and delete data in the format needed by your applications without writing structured queries. Nintex[8] supports database integration through its Workflow Connectors. It offers actions for querying and updating database records within workflows, with support for custom queries. Pega[9] offers database connectivity through its Data Pages feature, allowing integration with various database systems and external data sources. Outsystems[10] supports database integration through its Integration Studio. It offers visual mapping tools for defining database entities, queries, and data relationships within applications.

[1] bonitasoft.com
[2] mendix.com
[3] activiti.org
[4] flowable.com
[5] camunda.com
[6] bizagi.com
[7] appian.com
[8] nintex.com
[9] pega.com
[10] outsystems.com

L2. How do they define the connection or relation between process variables and the database? Bonita defines process variables within its BPMN diagrams through its Studio interface. The connection between process variables and the database is established through Bonita's data modeling capabilities. Bonita allows mapping process variables to database tables using connectors or through custom Java code. In Mendix, process variables are defined within microflows. These variables can be defined as attributes of entities within the Mendix domain model. Process variables can be mapped to attributes of entities, which are then persisted in the underlying database. Activiti, Camunda, and Flowable allow process variables to be defined within their BPMN diagrams or programmatically. Interaction with databases is typically achieved through code. Bizagi, like Bonita, defines process variables within BPMN diagrams through Studio interface. Users can define entities and attributes within Bizagi's Data Modeler, which maps to tables and columns in the underlying database. In Appian, similar to Bonita and Bizagi, process variables are defined within process models or through its interface designer. The integration with databases is achieved using built-in connectors. Nintex defines process variables within the workflow. Interaction with databases in Nintex is achieved through built-in connectors. Pega defines process variables within the case types or through interface. It interacts with databases using its built-in database integration features. Process variables in OutSystems are defined within business processes through visual development environment.

L3. How do they define the interaction with the user? Is it done using graphical presentation? All the tools mentioned in Table 2.2 employ a form of graphical representation for defining processes and user interactions. However, none define graphical integration with the user directly within the process model.

L4. How do they define the functions, data operations, and CRUD operations? Since the tools listed in Table 2.2 are based on BPMN, none of them support graphical representation of data operation definitions within the model. Each tool defines data operations using its own method. For example, in Bonita, operations are defined that can utilize Java functions to interact with BDM and process variables. Additionally, connectors or scripts can be used to define operations. Activiti, Flowable, and Camunda define functions using Java classes or scripting languages within service tasks. Bizagi's operations are implemented using Bizagi's data management features and by configuring appropriate database interactions within processes. Other systems use built-in connectors to access the database and custom code for actions and data operations.

L5. What type of verification do they use? (to avoid deadlock, ...) All the tools provide validation of the process models to ensure correct configuration and functionality. Some tools offer simulation features to test different scenarios and detect potential deadlocks or bottlenecks. However, not all of these tools support a formal mechanism for verifying the model before execution.

L6. Is it an open-source BPMS? Among the BPMSs listed, Bonita, Activiti, camunda, and Flowable are open source, providing users with flexibility and community-driven support.Bonita also offers an open-source version alongside its enterprise edition. The remaining tools, including Mendix, ProcessMaker, Bizagi, Appian, Nintex, Pega, and Outsystems, are proprietary solutions.

In our work, utilizing a BPMS or an LCDP to execute our process model is essential. Therefore, we opted for an open-source solution. Given Bonita's recognition as a leader in the 2023 SPARK Matrix for Digital Process Automation Software [11], we decided to utilize it as the platform for executing our process model.

References

1. Calvanese, D., Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Formal modeling and smt-based parameterized verification of data-aware BPMN. In: Hildebrandt, T.T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) Business Process Management - 17th International Conference, BPM 2019, Vienna, Austria, September 1-6, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11675, pp. 157–175. Springer (2019). https://doi.org/10.1007/978-3-030-26619-6_12, https://doi.org/10.1007/978-3-030-26619-6_12
2. Calvanese, D., Montali, M., Patrizi, F., Rivkin, A.: Modeling and in-database management of relational, data-aware processes. In: Giorgini, P., Weber, B. (eds.) Advanced Information Systems Engineering - 31st International Conference, CAiSE 2019, Rome, Italy, June 3-7, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11483, pp. 328–345. Springer (2019). https://doi.org/10.1007/978-3-030-21290-2_21, https://doi.org/10.1007/978-3-030-21290-2_21
3. Combi, C., Oliboni, B., Weske, M., Zerbato, F.: Conceptual modeling of processes and data: Connecting different perspectives. In: Conceptual Modeling - 37th International Conference, ER (2018)
4. Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Delta-bpmn: A concrete language and verifier for data-aware BPMN. In: Business Process Management - 19th International Conference, BPM 2021 (2021)
5. Giacomo, G.D., Oriol, X., Estañol, M., Teniente, E.: Linking data and BPMN processes to achieve executable models. In: Advanced Information Systems Engineering - 29th International Conference, CAiSE (2017)
6. Hirzel, M.: Low-code programming models. Commun. ACM **66**(10), 76–85 (2023). https://doi.org/10.1145/3587691, https://doi.org/10.1145/3587691
7. Meyer, A., Pufahl, L., Fahland, D., Weske, M.: Modeling and enacting complex data dependencies in business processes. In: Business Process Management - 11th International Conference, BPM (2013)
8. Nour Eldin, A., Baudot, J., Gaaloul, W.: Zooming in for clarity: Towards low-code modeling for activity data flow. Business Process Management Forum (2023)
9. OMG: Business Process Model and Notation (BPMN), Version 2.0 (2011), http://www.omg.org/spec/BPMN/2.0
10. Steinau, S., Marrella, A., Andrews, K., Leotta, F., Mecella, M., Reichert, M.: DALEC: a framework for the systematic evaluation of data-centric approaches to process management software. Softw. Syst. Model. (2019)

[11] https://www.bonitasoft.com/awards/bonitasoft-leader-in-2023-spark-matrix-for-dpa-software