Extending BPMN to Enable the Pre-Modelling of Flexibility for the Control Flow of Business Processes

Thomas Bauer
Hochschule Neu-Ulm, University of Applied Sciences, Wileystr. 1, 89231 Neu-Ulm, Germany

Keywords: Business Process, Flexibility, Build-Time, Control-Flow, BPMN.

Abstract: Pre-modelling of already known flexibility requirements of business processes (BP) already at build-time has the advantage that the resulting run-time deviations can be reviewed and approved. Furthermore, this results in less effort for the end users compared to completely dynamic changes at run-time. Corresponding concepts have been developed in previous work. In this paper we present an extension of the BPMN standard. It allows to model corresponding BP with existing BP modelling tools. Scientific literature is analysed in order to identify suitable methods for BPMN extensions. Then, they are used to develop a BPMN extension that allows the creation of BP models that contain the mentioned pre-modelled flexibility aspects.

1 MOTIVATION

At run-time, flexibility for business processes (BP) can be achieved by using dynamic changes (Reichert and Weber, 2012) to deviate from the modelled control flow. For this purpose, the end user defines, for instance, that an additional activity shall be inserted at a specific point into the BP. Flexibility by Design (Kumar and Narasipuram, 2006; Schonenberg et al., 2007) follows a different approach: Flexibility probably required at run-time, that is expectable already at build-time for a certain point in the BP, is pre-modelled in the process graph. This has the advantage that such a deviation can be reviewed and approved by BP administrators at build-time. In addition, it can be precisely defined which users are allowed to trigger it. Furthermore, this reduces their effort at run-time since it is not necessary that the end users specify a desired change in detail.

In the CoPMoF (Controlable Pre-Modeled Flexibility) project, several aspects for the pre-modelling of flexibility at build-time have been identified (Bauer, 2021, 2020, 2019). These include, optionally executed activities, optional edges, and dynamically triggered jumps to other activities. The pre-modelling of such a run-time behaviour requires additional constructs, i.e. they are not provided by current BP modelling languages. We extend the BPMN (Business Process Model and Notation) standard to show how they can be integrated. The extensions include simple aspects, such as marking an edge as optional (Bauer, 2023a), but also complex issues, such as defining multiple activities as a potential source or target of a dynamic jump, where a different behaviour can be specified for each involved activity (Bauer, 2024, 2022).

The overall goal of CoPMoF is that a BP, that includes such extensions, can be defined with a BP modelling tool and, later at run-time, executed by a BP engine. In addition to the BPMN extensions developed in this paper, this requires further topics that have been omitted due to lack of space. However, there already exist solutions for some of these topics:

- Possibilities for the graphical visualization (notation) of pre-modelled dynamic jumps and special types of sequence edges are discussed in (Bauer, 2022, 2023a), cf. Fig. 1.
- Formal execution semantics for such special types of sequence edges, for optional edges, and for dynamic jumps have been published in (Bauer, 2025, 2024, 2023b).

So far, there exists no work that examines how a BPMN extension should be designed to enable the pre-modelling of the required flexibility aspects for the control flow of BP. In addressing this task, we aim to achieve the “best possible design” of the extension. This means that it must satisfy the following conditions: (i) The extension must be compliant with the BPMN standard. (ii) Each individual adaptation must be based on the most suitable BPMN element, and new elements must only be defined if the BPMN

Bauer, T.
Extending BPMN to Enable the Pre-Modelling of Flexibility for the Control Flow of Business Processes.
DOI: 10.5220/0013134800003929

In *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS 2025) - Volume 2*, pages 804-812
ISBN: 978-989-758-749-8; ISSN: 2184-4992

standard does not contain any suitable elements for extension.

The structure of this paper corresponds to the chosen approach (research method): First, based on a literature review, suitable methodologies for BPMN extensions are selected (Section 2). Section 3 identifies requirements for the pre-modelling of flexibility. By applying the chosen methodology, these requirements are transferred into a concrete extension of the BPMN standard in Section 4. The paper concludes with a summary and an outlook to future work.

2 RELATED WORK

In the following, methodologies and concrete approaches for BPMN extensions are presented.

BPMN 2.0.2 (OMG, 2013) offers a mechanism for domain-specific extensions of this standard. Such an extension introduces new elements that extend BPMN without violating the standard itself. With this mechanism, the schema of the extension can be defined (e.g. with the BPMN element ExtensionDefinition), i.e. this extends the BP modelling capabilities. Furthermore, concrete values (with ExtensionAttributeValue) for such new elements, that are required for a specific process model, can be modelled and stored in the XML file that specifies this BP template.

Stroppi Methodology: The BPMN standard does not suggest a procedure for the usage of these elements when defining an extension. To close this gap, (Stroppi et al., 2011) proposes the following model-driven methodology: In Step (1), a "Conceptual Domain Model of the Extension" (CDME) of the desired extensions is created as a UML class diagram. (2) Then, it is transformed into a "BPMN Plus Extension Model" (BPMN+X). It shows the additionally required classes, as well as their attributes and associations (between each other and with standard BPMN classes). The result of Step (3) is a "XML Schema Extension Definition Model", which defines the XML Schema of the extension unambiguously. (4) The transformation into a corresponding XML Schema document can be performed automatically, e.g. by the Eclipse plug-in provided by (Stroppi et al., 2011).

Braun Methodology: The Stroppi Methodology represents a "purely technical" approach, i.e. it does not suggest comparing the meaning of new and existing BPMN elements. Therefore, it is possible that new elements are defined in Step (1) that are redundant. (Braun et al., 2016; Braun and Esswein, 2014) propose an equivalence check that avoids such redundancy: For each new element, it is checked whether it is really required or whether an already existing BPMN element can be extended (e.g. by new attributes). This approach can be used in combination with the Stroppi Methodology to achieve better results.

Now, approaches that extend BPMN (but do not develop a new methodology) are categorized in order to choose an appropriate methodology later on.

Standard-Compliant using Stroppi: This methodology is used, among others, at (Ben Hassen et al., 2017; Betke and Seifert, 2017; Domingos et al., 2016; Mandal et al., 2017; Stroppi et al., 2015). These papers consider technical aspects (e.g. event handling) as well as specific application domains (e.g. disaster response processes). (Braun et al., 2016) uses the Braun Methodology, in addition. Only two of the papers present the Steps (3) and (4) of the Stroppi methodology. This is reasonable since these steps are unambiguous and automatable transformations.

Standard-Compliant without Stroppi: This category includes (Dörndorfer and Seel, 2017; Dukaric and Juric, 2018; Heguy et al., 2019; Jankovic et al., 2015; Yousfi et al., 2016). (Martinho et al., 2015) considers controllable flexibility: In contrast to the CoPMoF approach, only hints are modelled where and how, for example, dynamic changes should be used. (Neumann et al., 2019; Onggo et al., 2018) additionally use the Braun Methodology.

Not Compliant with the BPMN Standard: (Abouzid and Saidi, 2019; Arevalo et al., 2016; Awad et al., 2009; Großkopf, 2008) realize BPMN extensions that are not compliant with the BPMN standard.

To summarize, some papers use the Stroppi Methodology or the Braun Methodology, while others develop a BPMN extension without a rigorous approach. There also exist approaches with a result that is not compliant with the BPMN standard.

3 REQUIREMENTS

In the following, a BP for the change management of vehicle parts (cf. Fig. 1) is used to explain the requirements that concern pre-modelling of flexibility. Detailed examples from practice are given in (Bauer, 2021). In addition, (Bauer, 2021) shows that these requirements are not sufficiently covered by BPMN or can only be realized by very complex process graphs. The creation of such complex models, however, may be too difficult for normal BP designers.

The main contribution of our whole approach is that it becomes easy for BP designers to pre-model flexibility, because they can use constructs offered by the modelling tool, e.g. to specify an activity or edge as optional ("opt." in Fig. 1), to use special edge types

(from Act. C to D), to or define areas where dynamic jumps are possible (in grey). At run-time, a user can trigger this flexibility with little effort, since the corresponding properties were pre-modelled already at build-time. The BP engine realizes the required run-time behaviour, which is possible because there is a fixed set of flexibility types (Bauer, 2021, 2020) that can be implemented in a generic software product (the PMS).

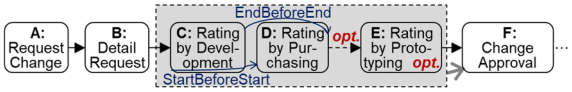

Figure 1: Simplified BP for Change Management.

3.1 Optional Activities and Edges

In the BP template of Fig. 1, Act. E is marked as optional. That means, it can be omitted if this change request is not relevant for prototyping (e.g. a change in the product documentation or the colour of a part). At run-time, this activity is offered to the potential actors in their worklists. One actor may perform it, or decide to omit this activity. Omitting an optional activity can also be useful at a lack of time. The requirements that concern optional activities are described in (Bauer, 2021), in detail. In the following, we briefly summarize the requirements that concern the BP model, i.e. the data it must contain.

Requirement 1 for Optionality Op1: It must be possible to store the information, that an activity is marked as optional, in the BP template (i.e. in the process model).

Op2: Such an activity is marked as optional in the worklists. Furthermore, at BP modelling, it shall be possible to define a text that explains when it is useful or even necessary to omit it. The end users can also see this explanation in their worklist.

Op3: Similar to actor assignments of activities (resource assignments), rights can be modelled that define who is allowed to omit this activity. As already mentioned, this can be allowed for the potential actors of this activity. However, it shall also be possible to define that only an administrator of the BP instance or the BP template is allowed to decide that the activity will be omitted.

Op4: It shall be possible to model a rule that defines when an optional activity shall be omitted automatically, e.g. because it is no longer necessary or useful. Assume that the sole purpose of an Act. X is to capture data that is only useful for Act. Y. Act. Y is executed in parallel to Act. X and displays (among others) information that was already captured by Act. X. After Act. Y has finished, it is no longer meaningful to execute Act. X. Therefore, it is omitted automatically by the BP engine. Since not only the potential actors of human tasks are able to omit an activity (cf. Op3 and Op4), also automatic process steps (e.g. service calls) and entire sub-processes can be marked as optional.

In Fig. 1, the edge from Act. D to Act. E is marked as optional, i.e. it defines the "normally desired" execution order. But it is also allowed to choose another order, e.g. in exceptional cases. The optional edge enables that Act. E is performed simultaneously with or before Act. D. In this example, this makes sense since the rating of the purchasing department can be helpful for prototyping, but it is not really necessary. Both, Act. D and E are offered to the potential actors in their worklists, but Act. E is labelled as "actually not to be performed yet". Then, a user can decide to execute this Act. E prematurely, e.g. because time must be saved at the execution of this specific process instance.

Op5: For each edge, it can be defined whether it shall be Optional or Mandatory (the regular case). If a sequence of activities o_1 to o_n, with optional edges between them, is followed by an Act. m with a mandatory edge between o_n and m, Act. m only becomes startable when all activities o_1 to o_n are completed. However, in some scenarios, it may also make sense that Act. m waits only for the completion of its directly preceding Act. o_n. Then, Act. m can be started before an Act. $o_i \in \{o_1, \ldots o_{n-1}\}$ has finished, i.e. its succeeding Act. o_{i+1} (that is connected with an optional edge) was started prematurely. To realize such a behaviour, an edge with the type Soft is modelled from Act. o_n to Act. m (Bauer, 2023b).

Op6: As already mentioned, an activity is marked in the worklists accordingly, if it is prematurely executable. In addition, it shall be possible to define an explanation text that informs the user in which cases it makes sense to start it earlier.

3.2 Special Types of Sequence Edges

The normal behaviour of a sequence edge from Act. X to Act. Y is that Act. Y can be started as soon as Act. X has finished. This can be extended by allowing to use the start and end events of activities arbitrarily for the definition of sequence edges. In the BP of Fig. 1, Act. D (rating by the purchasing department) can start already when Act. C has been started. Such a behaviour saves time (concurrent engineering) and can be modelled with the edge type StartBeforeStart. In addition, the end of Act. D is only allowed after the end of Act. C (type EndBeforeEnd). The reason for this additional condition is that the final output

data of the development department must be inspected before finishing Act. D. With the type StartBeforeEnd, the succeeding Act. Y can be finished as soon as its preceding Act. X has been started. (Bauer, 2025, 2023a) introduces the following requirements for advanced types of sequence edges:

Ed1: An edge has one of the 4 types (i.e. all possible combinations): EndBeforeStart (normal case), StartBeforeStart, EndBeforeEnd, and StartBeforeEnd.

Ed2: These edge types define that the specified event of the target activity Y may occur (immediately) after the specified event of its start activity X. For edges, it shall be additionally possible to define time intervals that must be observed. For example, if an adhesive (applied at Act. X) must dry for a sufficiently long time, the next Act. Y may only be started 4 hours after the end of Act. X. For this purpose, it shall be possible to mark an edge as a time edge.

Ed3: Both, minimum and maximum time intervals, may be required. Therefore, a value minTime and a value maxTime can be defined for each time edge. However, one of these values can be omitted (i.e. then there is no minimum or maximum time restriction).

Ed4: To be able to inform actors of such activities appropriately (e.g. by email), an escalation text can be modelled. This text is used if the desired time for starting or finishing this activity has been exceeded.

3.3 Critical Section

During the development of a prototype part, the activities X (harden part) and Y (customer demonstration) are processed in an arbitrary order, i.e. they are located in different parallel branches of an AND-Split. Assume that there exists only one instance of the resource "prototype part". Since it is required by both activities, only one of the activities X and Y can be performed at the same time. Such a mutual exclusion is modelled by a critical section (Russell and Hofstede, 2006). It is not necessary that the assigned activities are located in a contiguous region of the process graph, e.g. non-critical activities can be located between the AND-Split and Act. X or Act. Y (Bauer, 2023a).

CS1: Arbitrary activities can be assigned to a critical section.

CS2: An activity can be assigned to several critical sections, e.g. if several exclusively usable resources are required at the execution of this activity.

CS3: The assignment of an activity to a critical section can be marked as optional. This is useful if, for certain process instances, a resource may be exceptionally not needed at all or is no longer used by the activity (although it has not yet finished). As with optional edges (Op6), the potential actors see a warning in their worklists that starting this activity is normally not desired, but is possible in principle.

CS4: Again, an explanation text can be modelled that explains why an overlapping execution of these activities is normally not desired.

3.4 Dynamic Jumps

In exceptional situations, the user shall be able to trigger a jump forward and backward in the process graph. This is necessary, for example, if process regions have to be skipped due to lack of time, or have to be repeated to correct data entered or actions performed during the original execution of activities. In Fig. 1, starting at one of the activities C, D, or E, a dynamic jump to Act. F (approval) is possible at any time. For instance, if a change is unavoidable (e.g. due to new legal requirements), it is possible to jump directly to the approval (Act. F) as soon as the request has been detailed in Act. B. However, the rating by the purchasing department (Act. D) must be caught up, because the received offers from suppliers are required later in the BP. Regions of activities, where dynamic jumps are allowed, are pre-modelled at build-time. Thereby, the BP designer can specify the exact behaviour of a specific jump with configuration options (Bauer, 2024, 2022). Such sophisticated jumps are not offered by classical BP metamodels (Russell and Hofstede, 2006). Furthermore, commercial BP engines offer only very limited possibilities for jumps. For example, IBM Business Administration Workflow (IBM, 2022) does not allow jumps into or out of regions with concurrently executed activities. A reason for this may be that, in such cases, the execution semantics (i.e. the intended behaviour) is not clear.

Jp1: For each jump, it shall be possible to model multiple potential source and target activities. At runtime, when a dynamic jump is triggered, the set of potential target activities is used by an actor to select its concrete target activities (one per parallel branch). It is not necessary that the source activities form a contiguous region, because there may exist activities between them, during those execution a jump does not make sense. Similarly, there may exist activities between the target activities where it does not make sense to continue the BP after the jump.

Jp2: An activity can be defined as a default target activity for a particular jump. This can reduce the effort for the end users for the selection of the concrete

target activities. Of course, the users can also select other activities as jump targets. If the potential jump targets are located in parallel branches, one target activity can be marked as the default for each branch.

Jp3: An activity can be modelled as a source or a target for several different jumps.

Jp4: A jump can have different directions (regarding the control flow). It is possible to model a forward, a backward, or a jump into another XOR/OR branch (sideward). Thereby, even jumps into and out of regions with concurrently executed activities are allowed.

Jp5: For each activity, the desired behaviour at the jump can be selected using configuration options. They can be defined for the source and target activities of the jump, as well as for activities in between and after them. The following configuration options are offered:

- **CatchUpMode** determines whether an activity, that was skipped during a forward jump, shall be caught up later. This can be useful, because the execution of its associated action or its output data will be required later in the BP. Catching up can concern activities that belong to the source and target regions of the jump, as well as to activities in between.
- **RepeatMode** defines for an Act. X, that is to be executed again after a backward jump, whether it shall be repeated in fact. This behaviour can be useful, e.g. because it will produce different output data. Therefore, with RepeatMode(X)=Discard, the output data of its first execution are discarded. In case of RepeatMode(X)=Control, the original data are kept, and the actor must check whether they must be adapted. For this purpose, at the forward execution after the jump, Act. X is started again with its data fields already pre-filled with original data. With RepeatMode(X)=Keep, the output data of its first execution is always kept and Akt. X is not executed again.
- **ContinueMode** defines for a backward jump whether activities must be aborted or shall be continued. The latter case is useful, for example, for an Act. X where the repetition of the other activities will never produce a different result. Thus, by completing Act. X and keeping its output data, time can be saved during the subsequent forward process execution. This configuration option can be used for the source activities of the backward jump as well as for their successors. ContinueMode(X)=Abort results in the cancellation of Act. X when the jump is triggered. With Complete, a running Act. X can be finished, but not started newly. The latter is additionally allowed with Start&Complete, so that further activities can be executed.

Jp6: For each jump, it can be defined who is allowed to trigger it. Again, the definition of these rights is similar to an actor assignment. If no special rights are defined for a jump, it can be triggered by any current actor of one of its source activities.

4 EXTENSION OF BPMN

The requirements are realized as an extension that is compliant with the BPMN standard. As described in (Zarour et al., 2019), it is easy to integrate such an extension into existing modelling tools. Furthermore, the Stroppi Methodology is used to develop the extension, because this is a well-defined and rigorous procedure. Additionally, the Braun Methodology is used to determine, which elements are newly required in fact, and for which it is sufficient to add attributes or associations to existing BPMN elements.

4.1 Development of the CDME

Following the Stroppi Methodology, the first step is to create the UML class diagram CDME shown in Fig. 2. It describes all the classes, associations, and attributes needed to implement the requirements described in Section 3. The additionally used Braun Methodology ensures that a new class is only introduced if it is really necessary. Furthermore, always the most suitable class for the extension is identified. Therefore, in each case, we discuss which class from the BPMN standard should be used for this purpose.

4.1.1 Optional Activities

The BPMN element Activity is extended, because (as explained in Requirement Op4), in addition to user tasks, among others, entire sub-processes may be optional. Therefore, the BPMN element Task is not suitable. FlowNode is too general, since it includes gateways, which cannot be omitted. According to the requirements Op1 to Op4 (cf. Section 3.1), Activity is extended by the following attributes and associations (see also Fig. 2):

Op1: optional The Boolean value true indicates that an activity is optional.

Op2: explanation This string contains the explanation text for the end users.

Op3: authorizedUsers This is a new association of the element Activity. Its target element has the type

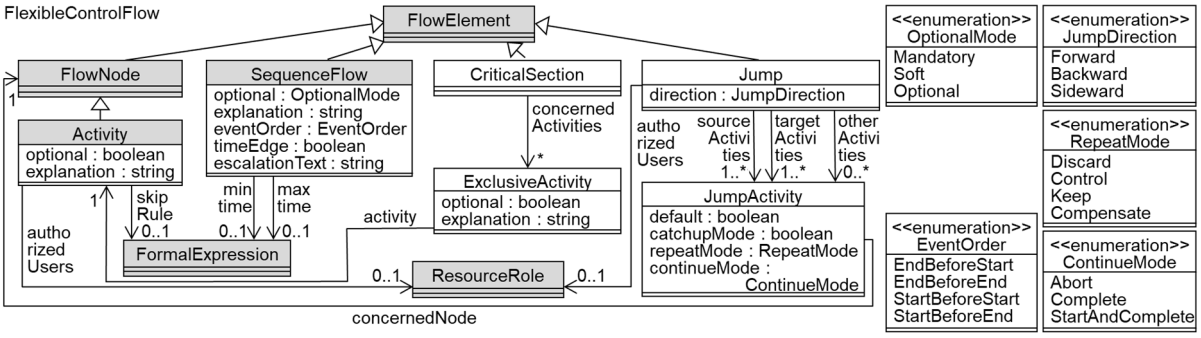

Figure 2: Conceptual Domain Model of the Extension (CDME) for the Extensions Required to Pre-Model Flexibility.

ResourceRole (the same as actor assignments in the BPMN standard). It can be omitted at process modelling. Therefore, the cardinality 0 is allowed. In this case, only the potential actors of the activity are allowed to omit it (cf. Section 3.1).

Op4: skipRule This association can be used to assign a FormalExpression. As soon as it evaluates to true, the activity is omitted automatically. This association can be missing. Then, omitting is always triggered manually by a user.

Similar as in (Stroppi et al., 2011), elements from the BPMN standard have a grey colour in Fig. 2. Attributes already defined by the BPMN standard are not repeated. Therefore, for example, no attributes are shown for the class FlowNode. New classes have a white background. The same applies to new attributes of BPMN standard elements.

4.1.2 Optional Edges

Sequence edges are realized in BPMN by the element type Sequence-Flow. Therefore, this type is extended by the following attributes:

Op5: optional This attribute describes whether an edge has the type Mandatory, Optional, or Soft (cf. Section 3.1). These types are defined by the enumeration OptionalMode.

Op6: explanation It contains the text that explains to the user, when a premature execution of the target activity of this optional edge makes sense.

4.1.3 Special Types of Sequence Edges

Since these are also edges between activities, the required attributes and associations are added to the element SequenceFlow, as well.

Ed1: eventOrder This attribute describes the type of the edge, i.e. whether a start or end event of an activity is referenced. For this purpose, the enumeration EventOrder is introduced, which contains the four possible values explained in Section 3.2.

Ed2: timeEdge It is a time edge if the Boolean attribute is true.

Ed3: minTime, maxTime These are associations to the BPMN element FormalExpression. It is used to calculate the corresponding point in time. Such an expression can contain a pure time length, that, for example, is added to the end time of the preceding activity to calculate the earliest or latest start time. However, more complex expressions are possible as well, e.g. “11:00 a.m. on the workday that starts at least 2 days later”. At both associations, the cardinality 0 is also allowed, because the expression can be missing. This means that the earliest or latest time is arbitrary.

Ed4: escalationText The escalation text that is used to inform users.

4.1.4 Critical Section

It is realized by the new element CriticalSection, since BPMN does not contain a suitable standard element as a basis for an extension. CriticalSection is a specialization of FlowElement, because it affects the execution order of activities. Several attribute values are modelled for each concerned activity (e.g. assignment is optional). Since these values can be different for each activity of a CriticalSection, we introduce the element type ExclusiveActivity. This allows to create a separate element of this type for each activity and, therefore, to store different attribute values.

CS1: concernedActivities This association assigns multiple elements of the type ExclusiveActivity to a CriticalSection. For each, the concerned activity is identified by the association activity, which refers to an element of type Activity. This element type was chosen because any kind of elementary task (e.g. UserTask, ServiceTask) can belong to a critical section, as well as a whole sub-process that must use a resource exclusively.

CS2: activity This association even allows that an activity is assigned to multiple critical sections (by assigning it to multiple elements of type ExclusiveActivity).

CS3: optional The value true indicates that the activity is assigned optionally.

CS4: explanation The explanation text for the end users in case of an optional activity assignment.

4.1.5 Dynamic Jumps

The new element Jump defines a pre-modelled jump. It is a specialization of FlowElement, because jumps affect the execution order.[1] Several associations are used to assign elements of type JumpActivity. Such an element defines properties (e.g. configuration options) of an activity that is affected by the jump. JumpActivity refers to an element of type FlowNode since jumps may concern any type of activity as well as gateways. The latter are relevant, because a jump can start directly after a gateway, and a gateway may be its target, e.g. since the condition of an XOR-Split must be re-evaluated.

Jp1: sourceActivities, targetActivities, otherActivities These associations refer to multiple elements of the type JumpActivity. Each JumpActivity assigns properties to a referenced activity. The associations sourceActivities and targetActivities must contain at least one activity, since otherwise no source or target of this jump is defined. The association otherActivities is used, for example, to define the configuration options for activities that are located between the source and target activities of the jump (cf. Requirement Jp5).

Jp2: default true indicates that an activity is a default target of the jump.

Jp3: concernedNode An activity can be a source or a target activity for an arbitrary number of jumps. In such cases, it is assigned to multiple JumpActivity elements with this association.

Jp4: direction Specifies the direction of the jump. This value is contained in the element Jump because it is valid for all activities of this jump. Its data type is the enumeration JumpDirection that offers the values Forward, Backward, and Sideward.

Jp5: catchupMode, repeatMode, continueMode They realize the configuration options and are attributes of JumpActivity since the values may vary for different activities of the same jump. The data type of catchupMode is Boolean, because only two values are possible. For the other attributes, enumerations are introduced that define appropriate values.

Jp6: authorizedUsers The BPMN element ResourceRole is used to assign user rights for this jump (as at Op3 for optional activities).

4.2 BPMN+X Model

Using the Stroppi Methodology, the CDME is transformed into the BPMN+X Model FlexibleControlFlow shown in Fig. 3. Its stereotype is <<ExtensionModel>>. As described in (Stroppi et al., 2011), the following transformation procedure was used:

- Classes from the BPMN standard are adopted as stereotype <<BPMNElement>> (e.g. FlowElement, Activity).
- If additional attributes are needed for such a class (e.g. for Activity), an additional class with the stereotype <<ExtensionDefinition>> is created. It contains these attributes (e.g. optional and explanation for OptionalActivity).
- Classes newly defined in CDME that are direct subtypes of BPMN classes are treated analogously, i.e. they have the stereotype <<ExtensionDefinition>> and the attributes are adopted (e.g. class Jump with the attribute direction).
- If a new class only has relations to BPMN standard classes, it gets the stereotype <<ExtensionElement>>. Attributes are adopted as well (e.g. JumpActivity with the attributes default, etc.).
- Enumerations result in the stereotype <<ExtensionEnum>> (e.g. OptionalMode).
- Generalizations between original elements of the BPMN standard are preserved (e.g. between FlowNode and FlowElement).
- Generalizations with a newly created class as subtype and a BPMN standard class as supertype result in an association with stereotype <<ExtensionRelationship>> (e.g. from Activity to OptionalActivity and from FlowElement to Jump).
- All associations of newly created classes are adopted. This applies to associations between new classes (e.g. sourceActivities for the class Jump) and to associations to BPMN standard classes (e.g. authorizedUsers for Jump).

The corresponding XML Schema Extension Model and the XML Schema document (cf. (Bauer, 2023c)) were created as described in (Stroppi et al., 2011).

[1] To realize a "jump edge" by extending the BPMN element SequenceFlow is not appropriate because SequenceFlow connects two FlowNodes. However, to model jumps, elements of type JumpActivity are required to enable the assignment of additional attributes. In addition, SequenceFlow would contain the unnecessary attributes condition and isImmediate.

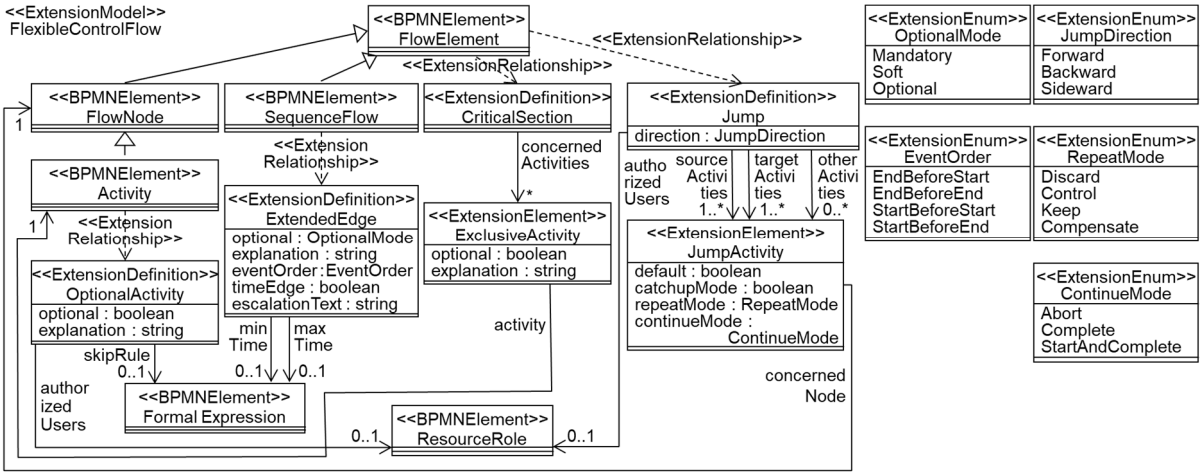

Figure 3: BPMN Plus Extension (BPMN+X) Model

5 SUMMARY AND OUTLOOK

Based on scenarios for the pre-modelling of flexibility (which have already been explained in preliminary work of the CoPMoF project in detail), this paper derives concrete requirements for the extension of the BP metamodel. Scientific literature was used to identify suitable methods for extending the BPMN standard. The chosen methodology results in an extension that is compliant with the BPMN standard. Furthermore, it does not contain any elements that are redundant to elements of this standard. The developed extension is a step towards being able to extend an existing BP modelling tool for the pre-modelling of flexibility. Since several extensions of BPMN are described in literature, a technical evaluation probably will not result in any relevant insights. However, the resulting tool can be used to explore the suitability of different notations for constructs that allow pre-modelling flexibility for the control-flow of BP (some graphical visualizations have already been proposed in preliminary work of CoPMoF).

So far, only for some flexibility aspects a formal execution semantics has been developed. This semantics is realized by execution rules, which extend and modify the rules used by classical BP engines. However, such rules are necessary (i.e. they partially still have to be developed) to extend a BP engine appropriately. Then, it is able to control workflows that include pre-modelled flexibility (e.g. optional sequence edges). Based on such a BP engine, it can be analysed, whether and to what extent, the CoPMoF approach actually covers real-world requirements and how often these extensions are needed.

The approach on which this work is based, i.e. to pre-model flexibility, is obviously not suitable for all scenarios and flexibility requirements. It can only be used for changes that can be anticipated already at build-time of the BP. As shown in previous work, in practice, there are several such BP with predictable flexibility requirements (e.g. in the change management process of Fig. 1). However, even for such BP, unexpected flexibility may be required, in addition. Therefore, it must be possible to perform (completely) dynamic changes at run-time of the BP, as well. Furthermore, there exist BP that require even more flexibility for the end users. In this case, completely different approaches are required, such as case handling.

REFERENCES

Abouzid, I., Saidi, R., 2019. Proposal of BPMN Extensions for Modelling Manufacturing Processes. Proc. Int. Conf. on Optimization and Applications.

Arevalo, C., Escalona, M.J., Ramos, I., Domínguez-Muñoz, M., 2016. A Metamodel to Integrate Business Processes Time Perspective in BPMN 2.0. Information and Software Technology 77, 17–33.

Awad, A., Grosskopf, A., Meyer, A., Weske, M., 2009. Enabling Resource Assignment Constraints in BPMN. Hasso Plattner Institute, Potsdam.

Bauer, T., 2025. Behaviour and Execution Semantics of Extended Sequence Edges in Business Processes, in: Proc. 27th Int. Conf. on Enterprise Information Systems. Porto.

Bauer, T., 2024. A Formal Execution Semantics for Sophisticated Dynamic Jumps within Business Processes, in: Proc. 26th Int. Conf. on Enterprise Information Systems. Angers, pp. 634–642.

Bauer, T., 2023a. Modelling of Advanced Dependencies Between the Start and the End of Activities in Business Processes, in: Proc. 25th Int. Conf. on Enterprise Information Systems. Prague, pp. 457–465.

Bauer, T., 2023b. Behaviour and Execution Semantics of Optional Edges in Business Processes in: Proc. Informatik 2023, Workshop ZuGPM, Berlin (in German).

Bauer, T., 2023c. BPMN-Erweiterungen zur Vormodellierung von Flexibilität für den Kontrollfluss von Geschäftsprozessen, in: Proc. 18th Int. Conf. on Wirtschaftsinformatik. Paderborn.

Bauer, T., 2022. Requirements for Dynamic Jumps at the Execution of Business Processes, in: Proc. 12th Int. Symposium on Business Modeling and Software Design. Fribourg, pp. 35–53.

Bauer, T., 2021. Pre-modelled Flexibility for the Control-Flow of Business Processes, in: Enterprise Information Systems. Springer, pp. 833–857.

Bauer, T., 2020. Business Processes with Pre-designed Flexibility for the Control-Flow, in: Proc. 22nd Int. Conf. on Enterprise Information Systems. pp. 631–642.

Bauer, T., 2019. Pre-modelled Flexibility for Business Processes, in: Proc. 21th Int. Conf. on Enterprise Information Systems. Heraklion, pp. 547–555.

Ben Hassen, M., Keskes, M., Turki, M., Gargouri, F., 2017. BPMN4KM: Design and Implementation of a BPMN Extension for Modeling the Knowledge Perspective of Sensitive Business Processes. Procedia Computer Science 121, 1119–1134.

Betke, H., Seifert, M., 2017. BPMN for Disaster Response Processes: A Methodical Extension. Proc. 47. Jahrestagung der Gesellschaft für Informatik 1311–1324.

Braun, R., Esswein, W., 2014. Classification of Domain-Specific BPMN Extensions. Proc. 7th IFIP Working Conf. on the Practice of Enterprise Modeling 42–57.

Braun, R., Schlieter, H., Burwitz, M., Esswein, W., 2016. BPMN4CP Revised - Extending BPMN for Multi-perspective Modeling of Clinical Pathways. Proc. 49th Hawaii Int. Conf. on System Sciences 3249–3258.

Domingos, D., Respício, A., Martinho, R., 2016. Reliability of IoT-Aware BPMN Healthcare Processes, in: Reis, Maximiano (Eds.): Internet of Things and Advanced Application in Healthcare. IGI Global, pp. 793–821.

Dörndorfer, J., Seel, C., 2017. A Meta Model Based Extension of BPMN 2.0 for Mobile Context Sensitive Business Processes and Applications. Proc. 13th Int. Conf. Wirtschaftsinformatik 301–315.

Dukaric, R., Juric, M.B., 2018. BPMN Extensions for Automating Cloud Environments using a Two-layer Orchestration Approach. Journal of Visual Languages & Computing 47, 31–43.

Großkopf, A., 2008. An Extended Resource Information Layer for BPMN. Hasso-Plattner-Institute for IT Systems Engineering, Potsdam.

Heguy, X., Zacharewicz, G., Ducq, Y., Tazi, S., Vallespir, B., 2019. A Performance Measurement Extension for BPMN. Enterprise Interoperability VIII, Proc. I-ESA Conferences 333–345.

IBM, 2022. Business Automation Workflow 22.x. URL https://www.ibm.com/docs/en/baw/22.x (accessed: 2025/01/17).

Jankovic, M., Ljubicic, M., Anicic, N., Marjanovic, Z., 2015. Enhancing BPMN 2.0 Informational Perspective to Support Interoperability for Cross-Organizational Business Processes. Computer Science and Information Systems 12, 1101–1120.

Kumar, K., Narasipuram, M.M., 2006. Defining Requirements for Business Process Flexibility. Workshop on Business Process Modeling, Design and Support, Proc. of CAiSE06 Workshops Luxemburg, 137–148.

Mandal, S., Weidlich, M., Weske, M., 2017. Events in Business Process Implementation. Proc. Int. Conf. on Business Process Management 141–159.

Martinho, R., Domingos, D., Varajão, J., 2015. CF4BPMN: A BPMN Extension for Controlled Flexibility in Business Processes. Procedia Computer Science 64, 1232–1239.

Neumann, J., Franke, S., Rockstroh, M., Kasparick, M., Neumuth, T., 2019. Extending BPMN 2.0 for Intraoperative Workflow Modeling with IEEE 11073 SDC for Description and Orchestration of Interoperable, Networked Medical Devices. Int. Journal of Computer Assisted Radiology and Surgery 1403–1413.

OMG, 2013. Object Management Group - Business Process Model and Notation (BPMN) 2.0.2.

Onggo, B.S.S., Proudlove, N.C., D'Ambrogio, S.A., Calabrese, A., Bisogno, S., Levialdi Ghiron, N., 2018. A BPMN Extension to Support Discrete-event Simulation for Healthcare Applications. Journal of the Operational Research Society 69, 788–802.

Reichert, M., Weber, B., 2012. Enabling Flexibility in Process-Aware Information Systems. Springer.

Russell, N., Hofstede, A.H.M., 2006. Workflow Control-Flow Patterns: A Revised View. BPM Center Report.

Schonenberg, M.H., Mans, R.S., Russell, N.C., Mulyar, N.A., Aalst, W.M.P. van der, 2007. Towards a Taxonomy of Process Flexibility (Extended Version). University of Technology, Eindhoven.

Stroppi, L.J.R., Chiotti, O., Villarreal, P.D., 2015. Defining the Resource Perspective in the Development of Processes-aware Information Systems. Information and Software Technology 59, 86–108.

Stroppi, L.J.R., Chiotti, O., Villarreal, P.D., 2011. Extending BPMN 2.0: Method and Tool Support. Proc. Int. Workshop on Business Process Model and Notation 59–73.

Yousfi, A., Bauer, C., Saidi, R., Dey, A.K., 2016. uBPMN: A BPMN Extension for Modeling Ubiquitous Business Processes. Information and Software Technology 74, 55–68.

Zarour, K., Benmerzoug, D., Guermouche, N., Drira, K., 2019. A Systematic Literature Review on BPMN Extensions. Business Process Management Journal. 26, 1473–1503.