Modelling of Advanced Dependencies Between the Start and the End of Activities in Business Processes

Thomas Bauer[a]
Hochschule Neu-Ulm, University of Applied Sciences, Wileystr. 1, 89231 Neu-Ulm, Germany
thomas.bauer@hnu.de

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

Abstract: The control-flow of a business process (BP) defines the allowed execution orders of its activities. Until now, only whole activities can be used to define such orders. This shall be extended: Additional execution orders are enabled by allowing to use the start and the end events of activities at control flow modelling. For example, defining that the end of Act. A must happen before the end of Act. B, increases the flexibility since Act. B can be started earlier as with a classic sequence edge. This paper presents corresponding examples from practice and derives the resulting requirements for BP modelling. Furthermore, possibilities for a BP modelling tool are discussed, to visualize such dependencies graphically.

1 INTRODUCTION

An advantage of process-aware information systems (PAIS) (Reichert and Weber 2012), compared to traditional IT systems, is that the process management system (PMS) guarantees that a business process (BP) is executed exactly as modelled at build-time (process reliability). Furthermore, end users are unburdened from non-productive actions, e.g. searching the right function of the application or the data required for the current process step (activity). At PAIS, such actions are performed automatically. However, these applications also come with disadvantages: Some users dislike their reduced freedom caused by this active process control. Furthermore, in exceptional cases, the restricted execution orders of process activities can result in situations, where orders are forbidden, which would be advantageous for the business. One solution for this problem are dynamic changes that are triggered by a user at run-time of the BP, e.g. inserting a new activity dynamically (Reichert and Weber 2012). The project CoPMoF (Controlable Pre-Modeled Flexibility) follows a different approach: Required flexibility is already modelled at build-time (Bauer 2019, Bauer 2021). This has the advantage that such predictable and pre-modelled deviations can be checked and approved during BP design. Therefore, process reliability can still be guaranteed. In addition, this results in reduced effort for the end users since all information, required for the deviation, was already pre-modelled at build-time.

At (Bauer 2020, Bauer 2021), the idea to pre-model flexibility was presented and many different aspects were described shortly as, for instance, optional activities, alternative activities, multi-instance activities, dynamic jumps (Bauer 2022), or aborting an activity. In the following, one of these aspects is examined in detail: Current meta-models for BP consider activities as (atomic) units. The control-flow defines dependencies between such (whole) activities (e.g. a sequence), that are respected by the BP engine at run-time. We extend this concept, by allowing to use the start and the end event of an activity separately at BP modelling. Then, among other things, new types of sequence edges become available, e.g. it can be defined that the start of Act. A must happen before the start of Act. B. Since this results in additional execution orders, that cannot be modelled until now, this increases flexibility for the end users at BP execution (in a pre-modelled manner).

Currently, this approach for BP modelling was hardly respected by meta-models for BP as BPMN or by scientific literature (Russell and Hofstede 2006), for details see Section 4. This applies to semantic modelling of BP (the business view) as well as for its implementation in a PMS with the goal to control workflow execution automatically (the technical

[a] https://orcid.org/0000-0001-8360-8430

view). To reduce this gap, this paper deals with the following research question: Which scenarios exist, where dependencies between the start and the end events of activities shall be modelled, and how should they be visualized in a BP modelling tool?

This paper presents the corresponding requirements and explains their necessity by concrete examples from practice.[1] The work is based on scenarios that are known by the author because of his long-term work in industry and research in the area of BP management, as well as on BP examples from literature. They were examined in order to figure out, which dependencies exist between the start and the end events of activities. Herefrom, abstract and generally applicable requirements were derived. With this research design, of course, completeness of the requirements cannot be achieved. Instead, the requirements have to be extended in future. For this purpose, it is not sufficient to search respective building blocks in BP models that already exist in an organization, since it is not possible, until now, to use such new (i.e. not supported) modelling concepts in BP models.

In Section 2, the identified requirements are presented. Section 3 examines how the corresponding building blocks can be modelled graphically. In Section 4, it is discussed to what extent current tools, standards, and scientific work cover the presented topics. The paper closes with a summary and an outlook on future work.

2 REQUIREMENTS

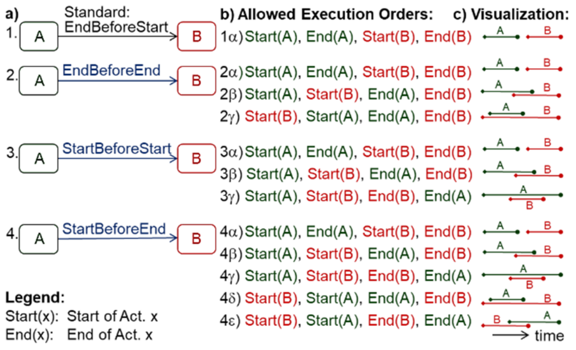
This section presents scenarios (i.e. requirements) where the order of activities has to be defined based on their start and end events. The validity of these requirements is shown by examples from practice.

2.1 Extensions of the Sequence Edge

As already mentioned, we extend the "normal" sequence edge between activities: It shall become possible to model edges arbitrarily from the start or the end of the preceding Act. A to the start or the end of the succeeding Act. B. These are similar dependencies as suggested in Allen's Interval Algebra (Allen 1983) and available for project management at network diagrams (Wysocki 2019). The result are four possible types of control flow edges (the first is the classic sequence):

[1] The development of a formal execution semantics for such dependencies and their (prototypical) realization is out of scope of this paper. This will be part of future work.

1. EndBeforeStart: The end of Act. A must happen before the start of Act. B
2. EndBeforeEnd: The end of Act. A must happen before the end of Act. B
3. StartBeforeStart: The start of Act. A must happen before the start of Act. B
4. StartBeforeEnd: The start of Act. A must happen before the end of Act. B
Fig. 1a shows how such control flow edges may be modelled (alternative visualizations are discussed in Section 3). For each edge type, Fig. 1b defines the allowed execution orders of Act. A and Act. B. Fig. 1c visualizes this behaviour graphically. The standard sequence edge (Type 1) comes with the lowest number of possible execution orders (only one) and, therefore, with the lowest flexibility. Type 4 (StartBeforeEnd) enables the largest number of execution orders (5 possibilities) of Act. A and B. Compared to a parallel execution (AND-Split, i.e. all execution orders are allowed) it is only forbidden that Act. B is completed before Act. A starts (i.e. End(B) before Start(A) is not allowed). Both, Types 2 and 3 enable three execution orders.


a)
Standard: EndBeforeStart
1. A
B
2. A
EndBeforeEnd
B
3. A
StartBeforeStart
B
4. A
StartBeforeEnd
B
b) Allowed Execution Orders:
c) Visualization:
1α) Start(A), End(A), Start(B), End(B)
2α) Start(A), End(A), Start(B), End(B)
2β) Start(A), Start(B), End(A), End(B)
2γ) Start(B), Start(A), End(A), End(B)
3α) Start(A), End(A), Start(B), End(B)
3β) Start(A), Start(B), End(A), End(B)
3γ) Start(A), Start(B), End(B), End(A)
4α) Start(A), End(A), Start(B), End(B)
4β) Start(A), Start(B), End(A), End(B)
4γ) Start(A), Start(B), End(B), End(A)
4δ) Start(B), Start(A), End(A), End(B)
4ε) Start(B), Start(A), End(B), End(A)
A
B
time
Legend:
Start(x): Start of Act. x
End(x): End of Act. x
Figure 1: Types of Sequence Edges between Act. A and B.

The example process depicted in Fig. 2 contains an edge of type EndBeforeEnd (Type 2) from Act. A to Act. B: The Act. A (vehicle cleaning: the truck driver removes rubbish from car cabin, e.g. transport documents) must be completed before the Act. B (vehicle transport) is finished. He is not able to clean the vehicle afterwards since he does not possess it anymore. It is possible, however, that the activities A and B are performed concurrently, e.g. when the driver cleans the vehicle during a transportation break.

In the following example, an edge of Type 3 (StartBeforeStart) is required (cf. Fig. 2): The Act. B (vehicle transport) must start before Act. C (inform customer about upcoming vehicle handover) can be started. Otherwise (i.e. when the information is transmitted to the customer earlier), the risk of a misinformation would be too high because the transport may be not performed in fact, e.g. since the truck is not available or broken.

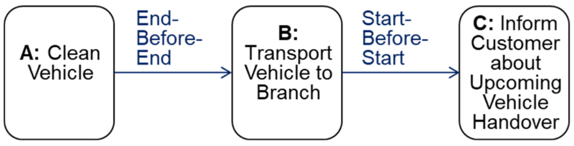

A: Clean Vehicle
End-Before-End
B: Transport Vehicle to Branch
Start-Before-Start
C: Inform Customer about Upcoming Vehicle Handover

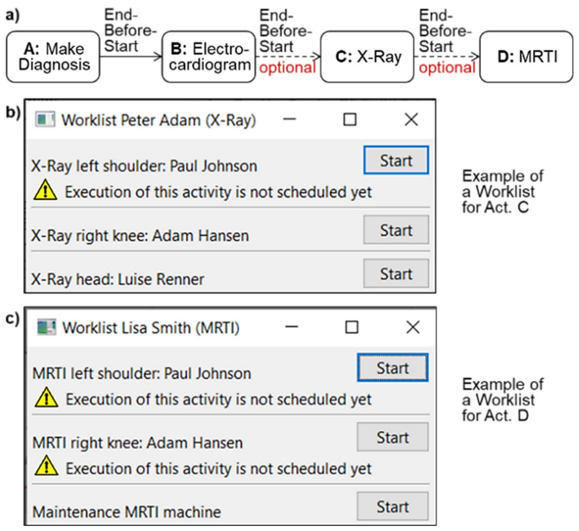
Figure 2: BP with Control Flow Edges of Type 2 and 3.

It shall be possible to model multiple of these edges between the same activities. If they are combined with a logical AND-Operation, all their conditions must be fulfilled, i.e. the possible execution orders result as the intersection of the orders allowed for the single edge types. It is not meaningful, however, to combine edges of the Types 1 or 4 with other types, since Type 1 only allows one single execution order (1α in Fig. 1b) and Type 4 allows all execution orders that are possible at the other types. Therefore, only a combination of edges with the Types 2 and 3 makes sense. This results in the allowed execution orders 2α=3α and 2β=3β (cf. Fig. 1). An OR-combination is meaningful only for these Types 2 and 3 as well. The allowed execution orders result as the union of both cases: 2α (=3α), 2β (=3β), 2γ, and 3γ. These combinations may be modelled either as two edges or as one edge with two types. In addition, it has to be defined, which kind of combination (AND/OR) shall be used.

It shall be even possible to model a second edge between two activities, that has the opposite direction. For instance, assume as extension of the example of Fig. 2, that Act. C (notify customer) must be finished before Act. B (transport vehicle) can be completed (in order to avoid a delayed or forgotten notification). This requires an additional edge from Act. C to Act. B of Type 2 (EndBeforeEnd). The result of these two edges is that Act. B must start before Act. C and it must end later than Act. C (cf. 3γ in Fig. 1).

2.2 Optional Control Flow Edges

As extension of the Types 1 to 4, it shall be possible to define that such an edge represents an optional execution order (cf. edges to Act. C and D in Fig. 3a). Their meaning is that this order is desired, but it is not absolutely necessary. After completion of Act. A, the Act. B (that shall be executed next in normal cases) is offered to the users in their worklists. Its optional successors Act. C and Act. D are also visible in the worklists. However, these worklist entries possess a warning sign (the triangles in Fig. 3b and c) and a text with an explanation. Therefore, the end users are able to detect that these activities shall not be started yet, but that this is only an option for exceptional cases. That means, a user has the possibility to decide explicitly, that he wants to perform one of these activities earlier than Act. B.



Worklist Peter Adam (X-Ray)

X-Ray left shoulder: Paul Johnson ⚠ Execution of this activity is not scheduled yet	Start
X-Ray right knee: Adam Hansen	Start
X-Ray head: Luise Renner	Start

Worklist Lisa Smith (MRTI)

MRTI left shoulder: Paul Johnson ⚠ Execution of this activity is not scheduled yet	Start
MRTI right knee: Adam Hansen ⚠ Execution of this activity is not scheduled yet	Start
Maintenance MRTI machine	Start

Figure 3: BP with Optional Edges and Resulting Worklists.

The hospital process depicted in Fig. 3a is used to explain that optional edges are required in practice: Directly after the diagnosis in Act. A, an electrocardiogram (ECG) is made (Act. B) normally, i.e. in regular cases that match to the standard of this hospital. Afterwards, an X-ray (Act. C) and a magnetic resonance tomography imaging (MRTI, Act. D) are made. For the case that one of the examination facilities is currently not available or overburdened, it is possible to deviate from this standard order: After completion of Act. A, all remaining activities B, C, and D are offered to users in their worklists. However, as already mentioned, Act. C and Act. D are marked as not scheduled for regular execution yet (cf. Fig. 3b and c). Assume the exceptional case that, after completion of Act. A for the patient Paul Johnson, the ECG machine (required for Act. B) is currently not available. Then, the patient can be sent directly to the X-ray in order to perform Act. C earlier. Since the worklists of the employees of the X-Ray department contain this Act. C (Fig. 3b), they can start it without any difficulties. The Act. B and D are started later on.

For this patient Paul Johnson, it is also possible to start Act. D before Act. B and C (cf. Fig. 3c), due to the optional edge from Act. C to Act. D. This also applies to Adam Hansen. But for the latter patient, Act. B was already finished. Thus, his worklist entry of Act. C contains no warning (cf. Fig. 3b) since it can be started regularly.

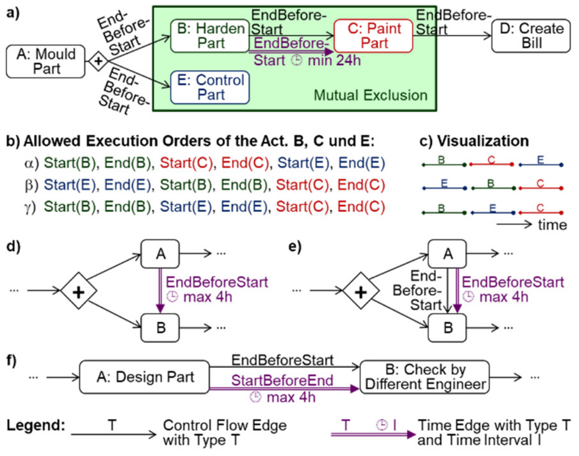
At the BP depicted in Fig. 3a, it is possible to perform the Act. B, C, and D concurrently. This is hardly meaningful for an examination process, since a patient can be in only one examination facility at the same time. Such an overlapping execution of activities can be prevented by using an area of mutual exclusion (cf. the next Subsection 2.3).

2.3 Mutual Exclusions

At a mutual exclusion (also named interleaved routing, critical section (Russell and Hofstede 2006)) the following applies to all activities of the corresponding area: Any Act. X, that was already started, must be finished before any other Act. Y can be started. That means, at any point in time, only one of these activities can be executed. Since this concerns the start and the end events of activities, mutual exclusions match to the topics presented in this article.

A mutual exclusion is only meaningful in combination with a parallel execution of activities (cf. the AND-Split in Fig. 4a, followed by the area of mutual exclusion visualized as green rectangle) or combined with the new edge types already presented in this section, since otherwise only one activity is executed at the same time. After completion of Act. A in Fig. 4a, all startable activities are offered to the end users in their worklists (Act. B and Act. E). When one of these activities is started by a user, all other activities are removed from the worklists. Therefore, only one activity can be executed at the same time. After completion of this activity, the others appear in the user worklists, again.

The manufacturing process depicted in Fig. 4a is used to explain the necessity of mutual exclusions: It is only allowed to execute one of the activities contained in the green rectangle (Act. B, C, and E) at the same time, i.e. the others are performed completely before or completely after this activity. In this example, first a part is moulded (Act. A), then it is hardened (Act. B), and painted in Act. C. Afterwards, a bill for this part is created (Act. D). In parallel to Act. B, C, and D, the part is shown to and controlled by the customer (Act. E). It is necessary for the execution of Act. B, C, and E that the part is present. Since these activities are performed at different locations, their execution must not overlap (in time). This is modelled by the area of mutual exclusion, that prevents that one activity is started while another one is currently running. Therefore, only the execution orders depicted in Fig. 4b and c are allowed.


a)
A: Mould Part
End-Before-Start
End-Before-Start
B: Harden Part
EndBefore-Start
EndBefore-Start ◷ min 24h
C: Paint Part
E: Control Part
Mutual Exclusion
EndBefore-Start
D: Create Bill
b) Allowed Execution Orders of the Act. B, C und E:
α) Start(B), End(B), Start(C), End(C), Start(E), End(E)
β) Start(E), End(E), Start(B), End(B), Start(C), End(C)
γ) Start(B), End(B), Start(E), End(E), Start(C), End(C)
c) Visualization
B C E
E B C
B E C
time
d)
A
EndBeforeStart ◷ max 4h
B
e)
A
End-Before-Start
EndBeforeStart ◷ max 4h
B
f)
A: Design Part
EndBeforeStart
StartBeforeEnd ◷ max 4h
B: Check by Different Engineer
Legend:
T
Control Flow Edge with Type T
T ◷ I
Time Edge with Type T and Time Interval I
Figure 4: Mutual Exclusions and Time Dependencies.

Since Act. D is not part of the area of mutual exclusion, it only has the restriction that it must be executed after Act. C (because of the sequence edge). Therefore, in Case α, Act. D can be executed in parallel to Act. E (i.e. at the same time) or earlier than Act. E.

In Case γ, Act. E is executed between Act. B and Act. C. If this behaviour is undesired, the process must be modelled differently: Instead of Act. B and Act. C, the upper branch of the mutual exclusion contains only one Act. BC. This Act. BC is composed of the elementary Act. B followed by Act. C. Since the composed Act. BC is part of the mutual exclusion, an overlapping execution with Act. E is forbidden, i.e. Act. E cannot be executed between Act. B and Act. C.

All variants of edges presented in this paper can be used to connect the activities of a mutual exclusion, e.g. optional edges. The assignment of an activity to an area of mutual exclusion can be optional itself, as well. Assume that this applies to Act. E of Fig. 4a. Then it can be started while Act. B or C are currently executed. But, similar to the behaviour of optional edges, there is a warning in the worklist that this will result in a not desired overlapping execution. Even multiple optionally assigned activities of a mutual exclusion can be executed simultaneously. But the PMS ensures that always only one activity, that is mandatorily assigned to a mutual exclusion, is executed at the same time. Of course, it shall be allowed that an activity belongs to several areas of mutual exclusion since multiple resources, that can only be used exclusively, may be required by the same activity. Each assignment to such an area may be defined as optional or mandatory.

An explicit building block for mutual exclusions is necessary in the BP meta-model because it is not meaningful to realize a mutual exclusion with sequence edges and gateways (i.e. Split- and Join-Nodes). Since several different execution orders are allowed (cf. Fig. 4b and c), with this workaround, it would be necessary to model many different execution orders separately. Furthermore, XOR-Splits must be used to select the path required at this process instance. The result would be a very confusing process model that contains redundant activities. But even worse, the desired behaviour cannot be realized in fact: The branching condition of a normal XOR-Split is evaluated before the succeeding activities are inserted into the user worklists. Therefore, the activity, that shall be started next, cannot be determined by an end user who selects a worklist entry. Instead, this decision is already made at the XOR-Split, i.e. too early.

2.4 Time Dependencies

In some scenarios, it is necessary that minimal and maximal time intervals are respected at the execution of activities. Some commercial PMS already allow defining a maximum time for the execution of a specific (single) activity. Exceeding this execution time results in an escalation, e.g. a warning of the actor or a message to his supervisor (Aalst *et al.* 2007). This mechanism is extended by allowing to define minimal and maximal time intervals between different activities, as well. The start and the end of such a time interval correspond to the start of the activity execution (i.e. the selection of the work item in the worklist by an actor) and its end (i.e. the completion of this activity). These are the same events that are relevant for the edge types presented in Section 2.1. Therefore, it is possible to model time restrictions as labels assigned to such edges (cf. Fig. 4). Defining a time interval, however, in some cases, does not require that also a corresponding control flow (sequence) edge exists: Assume, as depicted in Fig. 4d, that an Act. A is modelled in parallel to Act. B (AND-Split) and Act. B must be started not later than 4 hours after the end of Act. A to prevent delays. Then, a time edge of the type EndBeforeStart has to be used. In this scenario, it is allowed to perform Act. B earlier than or overlapping with Act. A. This is no longer possible if a control dependency of type EndBeforeStart exists from Act. A to Act. B, additionally. In Fig. 4e, this dependency is visualized as a control flow edge, explicitly. In order to enable also execution orders as depicted in Fig. 4d, we need "pure" time edges, i.e. a time edge that does not result in a control flow dependency (of Type 1 to 4, cf. Section 2.1).

The necessity of time edges shall be explained by examples from practice: At the manufacturing process depicted in Fig. 4a, after hardened (Act. B), the part has to cool for at least 24 hours before it can be painted (Act. C). The latter activity, therefore, only appears in the worklists of the end users after these 24 hours have elapsed. This is an example for a time dependency between the end of the preceding Act. B and the start of the succeeding Act. C, i.e. Type 1 (EndBeforeStart).

As an extension of the example of Fig. 4a, it may be meaningful to define a maximum time interval (e.g. max. 72 hours) for this edge. Then, Act. C can be started between 24 and 72 hours after completion of Act. B. Therefore, sufficient cooling is still guaranteed, but undesired storage costs and delays are prevented.

It can be necessary, in addition, that several edges with different types are modelled between the same activities. Some of these edges may only define time restrictions. At the manufacturing process of Fig. 4a, it may be defined additionally, that painting (Act. C) must be completed not later than 80 hours after the end of Act. B (again: to prevent delays), i.e. an additional time edge of type EndBefore<u>End</u> is required with the restriction "max. 80h".

For time edges, even the Type 4 (StartBeforeEnd) is meaningful: As mentioned, such an edge only defines a time restriction (i.e. no execution order). The order may be defined by another edge type, e.g. Type 1 (EndBeforeStart). This applies to the excerpt of a development process depicted in Fig. 4f: Both activities have to be performed within 4 hours. That means, from the start of the part design by an engineer (Act. A) till the completion of the check by his colleague (Act. B), at most 4 hours must elapse. This prevents delays, but the engineers have the possibility to use (i.e. distribute) the whole available processing time more flexibly than at a process model that defines a maximum time of 2 hours for Act. A and also 2 hours for Act. B (after completion of Act. A).

The process designer defines time intervals at BP modelling (build-time). The PMS must guarantee these restrictions at BP execution (run-time). It is not possible for the PMS to perform the start or completion of an activity directly, since these actions are executed by end users manually. The following measures, however, can be used to enforce or support time restrictions:

- **Minimum Times:** If a minimal time interval is defined for the start of a succeeding Act. X (i.e. StartBeforeStart or EndBeforeStart), the PMS does not insert the corresponding entry into the worklists till the defined minimal time is elapsed. Therefore, the users are not able to start Act. X too early. A minimal time for the end of an Act. Y may be enforced by preventing the completion of Act. Y. For this purpose, the completion function can be deactivated (e.g. the "Complete Button" is inactive and visualized in grey).

- **Maximum Times:** If the start or the end of an activity does not occur in time, the PMS performs an escalation. For this purpose, the same techniques can be used at current PMS when guaranteeing the execution time of single activities (e.g. notifications, delegations to different users, etc.). In order to be able to meet the defined maximum times in fact, the escalation should be performed in good time before the deadline elapses (Aalst *et al.* 2007). At BP modelling, it shall be possible to define this "warning time" for each activity individually.

Also time edges can be optional (cf. Section 2.2). In case of an optional minimal time distance, it is possible to start or complete the activity too early, but an appropriate warning is displayed at this worklist entry (cf. Fig. 3b and c) or the "Complete Button". When an optional maximal time dependency is missed, the escalation message may contain a different text (e.g. "it is desired – but not absolutely necessary – to start / complete the Act. … at least at …") or a different type of escalation can be defined at BP design (e.g. a message to the end user instead to his supervisor).

3 VISUALIZATION CONCEPTS

The fact that an edge is optional or represents a time dependency is solely a single property of this edge. Therefore, it is an appropriate graphical visualization method to place a label next to such an edge and, perhaps in addition, to use different colours or line styles (e.g. a dotted or double line, cf. Fig. 3a and 4). Therefore, in the following, only visualization of the other types of dependencies are discussed. We present different visualization styles and explain their advantages and disadvantages. However, a final rating or even an empiric examination of the suitability of these styles are out of scope

3.1 Types of Control Flow Edges

The new types of control flow edges (cf. Section 2.1) may be visualized in a BP modelling tool with one of the following styles:

1. In Fig. 1 to 4, a label next to the edge (e.g. StartBeforeStart) is used to distinguish these types. This is unambiguous, but the meaning is not visible graphically. Therefore, a disadvantage is that the BP designer has to read and understand each label, i.e. the type of the edge cannot be recognized "at first glance".
2. An alternative to a label is to use different edge styles (sidled/dotted/double lines, different colours, or arrowheads) for the different edge types. This is the same technique as used (additionally to a label) in Fig. 3a and 4 for optional and time edges. This visualization style is place-saving, but the meaning of an edge is even more difficult to recognize as with Style 1. This becomes even worse by the fact, that 4 different edge styles are required for the 4 types of control flow edges.
3. Fig. 5a depicts a special graphical visualization style for the new types of control flow edges (i.e. Types 2-4): The start and the end of an edge concern the start or the end event of an activity. This is visualized by an edge that starts and ends at the left or right side of the rectangle representing the activity. In addition, this "point of contact" of an activity may be marked with special symbols: In Fig. 5a, small circles, that are similar to the start and end events of BPMN, are used for this purpose. The advantage is that the type of the edge is easy to recognize. However, such edges cross the lines of the activity rectangles. Furthermore, this style is only appropriate for modelling solely from left to right. It may be adapted for modelling from top to down. But if place shall be saved by modelling a sequence sidled (i.e. from left to right and underneath back to left) this style becomes confusing, since then the points of contact are on the wrong side. Finally, it may be effort to realize such edges in a modelling tool because no straight edges can be used (cf. Fig. 5a).
4. The styles depicted in Fig. 5b and c distinguish the edge types by symbols that represent iconized visualizations of Style 3 (cf. Fig. 5a). At Fig. 5b, a symbol is attached to each start and each end point of an edge, that depicts a "BPMN-like" start or end event and an outgoing or incoming edge. At Fig. 5c, only one symbol is attached to the edge that depicts the concerned event types and their connecting edge. An advantage of these styles is that straight edges can be used that can be connected with the activity rectangles at arbitrary points. This results in a neatly arranged process

graph. Again, the edge type is visualized graphically. The BP diagram, however, contains additional symbols that consume place. Since the symbols can be omitted for the most commonly used Type 1 (EndBeforeStart), this is almost of no consequence.

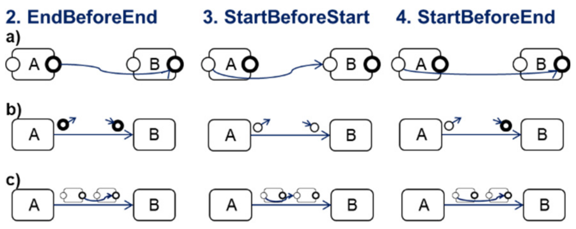


Figure 5: Visualizations of new Types of Sequence Edges.

3.2 Mutual Exclusions

As already depicted in Fig. 4a, a mutual exclusion can be visualized by locating the concerned activities in an area that is symbolized by a rectangle and may be coloured additionally. Therefore, these activities are easy to identify. In the example of Fig. 6a, the mutual exclusion concerns activities that are not connected. In such cases, this style may be difficult to realize by a modelling tool. This becomes even worse if a third branch is located between the branches depicted in Fig. 6a, and the activities of this additional branch do not belong to the area of mutual exclusion (cf. Act. I to L in Fig. 6b).

The style depicted in Fig. 6b avoids such confusing visualizations and is easy to realize in a modelling tool, in addition: All activities that belong to the same area of mutual exclusion are marked with a symbol. In the example of Fig. 6b, the symbol X (mutual eX-clusion) is used for this purpose. If multiple areas of mutual exclusions exist, this symbol can be extended by a number (e.g. X-1). In the depicted example, the activities of the same mutual exclusion are filled with the same colour to simplify their identification. This style has the advantage that it is easy to realize, even for not connected activities, multiple areas of mutual exclusions, and for activities that belong to multiple of these areas (by attaching multiple symbols and filling activities with multiple colours, e.g. striped). An optional assignment of an activity to a mutual exclusion can be visualized easily, as well, e.g. by a label X-1_optional, perhaps in addition with a different text style or a brighter filling colour. The disadvantage of this approach is that the area itself is not directly visible (e.g. as rectangle).

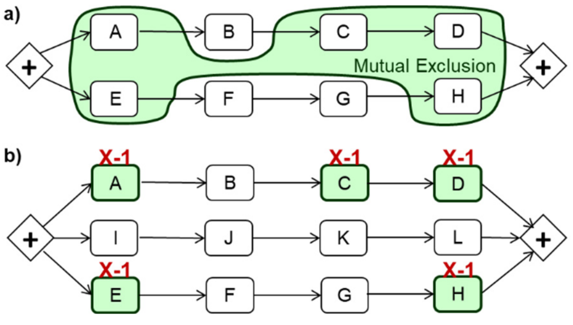


Figure 6: Graphical Visualizations for Mutual Exclusions.

4 STATE OF THE ART

This section describes the degree of realizability of the presented concepts in current PMS (e.g. with workarounds) and relevant scientific literature.

4.1 Standards and IT Systems

Many commercial PMS are based on standardized BP modelling languages as BPEL and BPMN. IBM Business Automation Workflow (IBM 2022), for instance, uses BPMN diagrams for BP modelling. Thus, such products offer the building blocks defined by these standards. With respect to the topics presented in this paper, BPEL and BPMN offer the same relevant building blocks. They are very similar to the possibilities offered by semantic BP modelling languages, i.e. when modelling the business view (e.g. as eEPC in ARIS).

The sequence edges of all these modelling languages solely describe the pure sequential execution of activities (Type 1). Overlapping execution can be realized with AND-Splits. For the activities of the resulting parallel branches, however, arbitrary execution orders are allowed. There does exist no building blocks that enable to define restrictions introduced by the Types 2 to 4 of Section 2.1. That means, it is not possible to define, for instance, that (only) the start of Act. A must happen before the start of Act. B.

Optional edges are not part of the mentioned modelling languages and, therefore, cannot be realized in commercial PMS that are based on these standards.

BPEL and BPMN do not offer a building block for mutual exclusions, directly. For instance with BPMN, as a workaround, this functionality can be modelled by realizing all allowed execution orders and using XOR-Splits of the type “Deferred Choice”. This (advanced) building block, however, is not offered by most commercial PMS (Havey 2009).

Time edges are not part of the standards as well. At BPMN, however, a maximal time interval can be realized with an intermediate timer event. An additional path has to be modelled that is executed in case of a missed deadline. It consists of the timer event, followed by an activity that performs the escalation, e.g. sends a message. It is executed when the event occurs (i.e. the deadline is missed). With this workaround, (only) maximal time intervals can be realized in a PMS that is based on BPMN. The result, however, are complex process graphs that may be too confusing for "normal" BP designers.

For the visualization of time dependencies, BPMN only offers the "clock symbol" of timer events. Therefore, time edges and the time distances are not directly visible. Instead, the already mentioned complex process graphs result. Since there exist no building blocks for the other dependencies presented in Section 2, BPMN does not propose a notation (i.e. visualization).

4.2 Scientific Literature

The control flow patterns (Russell and Hofstede 2006) describe many building blocks for BP design and, therefore, enable many different execution orders. These patterns, however, only respect the order of whole activities, e.g. Act. A must be finished before Act. B can be started. That means, control flow edges of Types 2 to 4 are not respected. Optional edges are not mentioned at all. This work describes mutual exclusions. However, the corresponding control flow patterns Critical Section and Interleaved Routing are presented without a discussion of more advanced requirements (as introduced in Section 2.3).

(Heinlein 2001) enables to define arbitrary dependencies between the start and the end events of activities (even the Types 2 to 4). Mutual exclusions can be realized as well. The goal of this approach, however, is not to define dependencies between activities of the same process instance. Instead, it considers dependencies between activities of different process instances or even process templates (i.e. process types). Since it is not possible to model such dependencies within a single process graph, regular expressions and special interaction graphs are proposed.

Case Handling (Aalst *et al.* 2005, Hewelt and Weske 2016) does not use control flow edges to define the execution order of activities. Instead, an activity becomes executable when all required input data are available. This enables the realization of edges with Type 3 (StartBeforeStart): For this purpose, the preceding Act. X must write a data object, required by the succeeding Act. Y, directly when Act. X starts. This workaround is only meaningful for scenarios, where such a data-flow exists from Act. X to Act. Y. Furthermore, at case handling, data-dependencies concern only the start of an activity, i.e. the Types 2 and 4 (..Before<u>End</u>) cannot be realized.

CrossFlow (Grefen *et al.* 2000, Klingemann 2000) proposes optional edges as optional execution order. Similar as described in Section 2.2, the concerned activities shall be executed in the modelled order. Their parallel execution, however, is allowed in exceptional cases, as well.

At constraint-based approaches (see (Reichert and Weber 2012) for an overview), no control flow graph is modelled at all. Instead, constraints were defined, which restrict the set of possible execution orders. These constraints refer to whole activities, and not to their start and end events separately. Therefore, dependencies of the Types 2 to 4 cannot be realized. Optional constraints allow to realize optional execution orders. A constraint of the type respondedExistence(A, B) (Reichert and Weber 2012) allows defining a mutual exclusion for these activities.

The necessity of an explicit building block for mutual exclusions is explained in (Laue and Kirchner 2017) at examples from practice, without discussing detailed requirements.

(Lanz *et al.* 2010) presents time patterns that enable the definition of minimal and maximal time intervals between activities. They can arbitrarily refer to the start and the end points of the preceding and succeeding activities. Therefore, all four edge types of Section 2.1 are covered. But these time constraints are not discussed in the context of the other concepts presented in Section 2.

The graphical visualization of mutual exclusions in (Russell and Hofstede 2006) is similar to Fig. 4a. This work, however, does not focus on visualizations. Instead it explains the meaning of control flow patterns. Constraint-based approaches do not use a process graph for BP definition. Therefore, they do not present suggestions for appropriate visualization. The interaction graphs of (Heinlein 2001) are not proposed as modelling technique for BP designers, but are a formal (mathematical) representation of the regular expressions.

To summarize: We propose edges of the Types 2 to 4 for the modelling of dependencies between activities of the same BP for the first time. The concepts presented in Section 2.2 to 2.4 were already mentioned in literature, but partially in a different context (e.g. not for graph-based BP modelling) and not in combination with the other presented aspects that concern start and end events of activities. Therefore, for instance, optional time edges and areas of mutual exclusion with optionally assigned activities also represent original work. Current standards for BP modelling and the corresponding commercial PMS do not offer the presented concepts as well. However, some of them can be realized with workarounds.

5 SUMMARY AND OUTLOOK

In this paper, we propose new modelling concepts with the goal to extend the possibilities of current BP modelling languages. Their necessity is explained by examples from practice. The new Types 2 to 4 of control flow edges enable additional execution orders. Without these (i.e. at classic BP models), the flexibility of the end users at BP execution is restricted. Optional edges increase this flexibility by enabling additional execution orders as well. Areas of mutual exclusions and the definition of time intervals prevent that activities are executed at the wrong points in time. This enables a more detailed modelling of the desired behaviour. The control flow edges of Types 2 to 4 are a completely new aspect for BP. For the other topics, we have presented requirements that were not mentioned in BP literature before, e.g. optional time edges, the optional assignment of activities to an area of mutual exclusion, or to multiple such areas.

This paper is a first step in a new direction. The identified requirements are based on a limited number of BP. In order to complete these requirements, additional BP and application domains have to be analysed. This may also allow to figure out whether relevant examples exist, where control flow edges of Type 4 are required.

Additional possibilities for the visualization of the presented types of dependencies have to be identified in future (e.g. more comprehensible symbols). Experiments with BP designers can be used to figure out which visualization is best suited for BP modelling.

A formal execution semantics has to be developed for the presented building blocks, as well. It is required by a BP engine to control such processes, e.g. to identify the activities that can be started. Based on a (perhaps prototypical) implementation of such an engine, case studies have to be performed to rate the usability of the presented concepts for the end users.

REFERENCES

Aalst, W. van der, Rosemann, M., and Dumas, M., 2007. Deadline-based Escalation in Process-Aware Information Systems. *Decision Support Systems* (43(2)), 492–511.

Aalst, W. van der, Weske, M., and Grünbauer, D., 2005. Case Handling: A New Paradigm for Business Process Support. *Data & Knowledge Engineering*, 53 (2), 129–162.

Allen, J.F., 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26 (11), 832–843.

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

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

Bauer, T., 2021. Pre-modelled Flexibility for the Control-Flow of Business Processes: Requirements and Interaction with Users. *Enterprise Information Systems*, 833–857.

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

Grefen, P., *et al.*, 2000. CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises. *Computer Systems Science & Engineering*, 15 (5), 277–290.

Havey, M., 2009. *Essential Business Process Modeling*: O'Reilly.

Heinlein, C., 2001. Workflow and Process Synchronization with Interaction Expressions and Graphs. *Proc. 17th Int. Conf. on Data Engineering*, 243–252.

Hewelt, M. and Weske, M., 2016. A Hybrid Approach for Flexible Case Modeling and Execution. *Proc. 14th Int. Conf. on Business Process Management, Business Process Management Forum*, 38–54.

IBM, 2022. *IBM Business Automation Workflow 20.x and 21.x* [online]. Available from: https://www.ibm.com/docs/en/baw/20.x [Accessed 18 Oct 2022].

Klingemann, J., 2000. Controlled Flexibility in Workflow Management. *Proc. Int. Conf. on Advanced Information Systems Engineering*, Stockholm, 126–141.

Lanz, A., Weber, B., and Reichert, M., 2010. Workflow Time Patterns for Process-Aware Information Systems. *Proc. Enterprise, Business-Process, and Information*, 94–107.

Laue, R. and Kirchner, K., 2017. Using Patterns for Communicating About Flexible Processes. *Proc. 18th Int. Conf. on Business Process Modeling, Development and Support*, Essen, 12–19.

Reichert, M. and Weber, B., 2012. *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*: Springer.

Russell, N. and Hofstede, A. ter, 2006. *Workflow Control-Flow Patterns: A Revised View*. BPM Center Report BPM-06-22.

Wysocki, R.K., 2019. *Effective Project Management*: Wiley.