Thomas Allweyer

BPMN 2.0

Introduction to the Standard for Business Process Modeling

2nd, Updated and Extended Edition

2 BPMN by Example

2.1 A First BPMN Model

As a starting point, a simple BPMN process model is considered. The model of posting a job in figure 1 can be directly understood by most people who previously have been concerned with any kind of process modeling. The way of modeling is similar to well-known flow charts and activity diagrams.

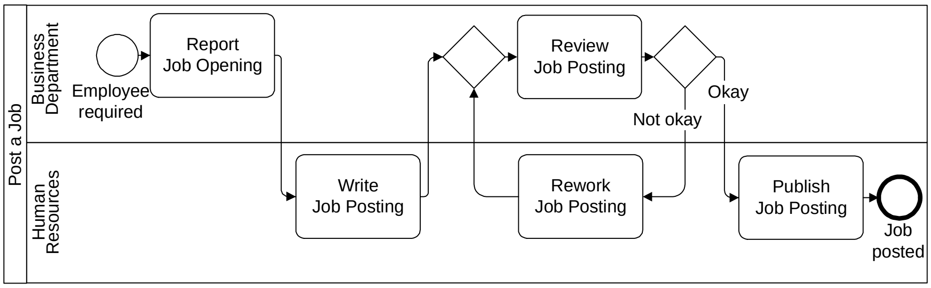

Figure 1: A simple BPMN model

A business department and the human resources department are involved in the process "Post a Job". The process starts when an employee is required. The business department reports this job opening. Then the human resources department writes a job posting. The business department reviews this job posting.

At this point, there are two possibilities: Either the job posting is okay, or it is not okay. If it is not okay, it is reworked by the human resources department. This is once more followed by the business department reviewing the job posting. Again, the result can be okay or not okay. Thus, it can happen that the job posting needs to be reviewed multiple times. If it is okay, the posting is published by the human resources department, and the end of the process is reached.

In reality, the process for creating and publishing a job posting can be much more complex and extensive. The presented example is – like all examples in this book – a simplification in order to have small and easily understandable models which can be used for explaining the different BPMN elements.

2.2 BPMN Constructs Used

Below, each element of the model in figure 1 is explained in more detail.

The entire process is contained in a pool. This is a general kind of container for a complete process. In the example above, the pool is labeled with the name of the contained process.

Every process is situated within a pool. If the pool is not important for understanding the process, it is not required to draw it in the diagram. In a process diagram which does not show a pool, the entire process is contained in an invisible, implicit pool.

Pools are especially interesting when several pools are used in order to model a collaboration, i.e. the interplay of several partners' processes. Each partner's process is then shown in a separate pool. This will be described in chapter 5.

The pool from figure 1 is partitioned into two lanes. A lane can be used for various purposes, e.g. for assigning organizational units, as in the example, or for representing different components within a technical system. In the example, the lanes show which of the process's activities are performed by the business department and which by the human resource department.

Pools and lanes are also called "swimlanes". They resemble the partitioning of swimming pools into lanes. Every participant of a competition swims only in his own lane.

The process itself begins with the start event "Employee required". Every process usually has such a start event. Its symbol is a simple circle. In most cases it makes sense to use only one start event, not several ones.

A rounded rectangle represents an activity. In an activity something gets done. This is expressed by the activities' names, such as "Report Job Opening" or "Review Job Posting".

The connecting arrows are used for modeling the sequence flow. They represent the sequence in which the different events, activities, and further elements are traversed. Often this is called control flow, but in BPMN there is a second type of flow, the message flow, which influences the control of a process as well, and is therefore some kind of control flow, too. For that reason, the term "sequence flow" is used. For distinguishing it from other kinds of flow, it is important to draw sequence flows with solid lines and filled arrowheads.

The process "Post a Job" contains a split: The activity "Review job posting" is followed by a gateway. A blank diamond shape stands for an exclusive gateway. This means that from several outgoing sequence flows, exactly one must be selected. Every time the right gateway in the job posting process is reached, a decision must be taken. Either the sequence flow to the right is followed, leading to the activity "Publish Job Posting", or the one to the left is selected, triggering the activity "Rework Job Posting". It is not possible to follow both paths simultaneously.

The logic of such a decision is also called "exclusive OR", abbreviated "XOR". The conditions on the outgoing paths determine which path is selected. If a modeling tool

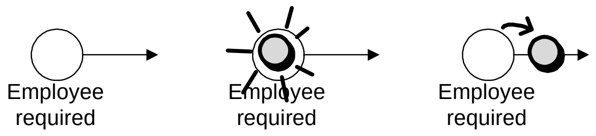

Figure 2: A start event creates a token

is used and the process has to be executed or simulated by a software program, then it is usually possible to formally define exact conditions. Such formal descriptions, which may be expressed in a programming language, can be stored in special attributes of the sequence flows.

If, on the other hand, the purpose of a model is to explain a process to other people, then it is advisable to write informal, but understandable, statements directly into the diagram, next to the sequence flows. The meaning of “okay” and “not okay” after the activity called “Review Job Posting” is clear to humans – a program could not make use of it.

Gateways are also used for merging alternative paths. In the sample process, the gateway on the left of the activity “Review Job Posting” merges the two incoming sequence flows. Again, this is an exclusive gateway. It expects that either the activity “Write Job Posting” or “Rework Job Posting” is carried out before the gateway is reached – but not both at the same time. It should be taken care to use a gateway either for splitting or for joining, but not for a combination of both.

The last element in the example process is the end event. Like the start event, it has a circle as symbol – but with a thick border.

2.3 Sequence Flow Logic

The flow logic of the job posting process above is rather easy to understand. In more complex models it is sometimes not clear how the modeled structure exactly is to be interpreted. Therefore, it is helpful if the meaning of the sequence flow’s elements is defined in an unambiguous way.

The logic of a process diagram’s sequence flow can be explained by “tokens”. Just as in a board game tokens are moved over the board according to the game’s rules, one can imagine moving tokens through a process model according to BPMN’s rules.

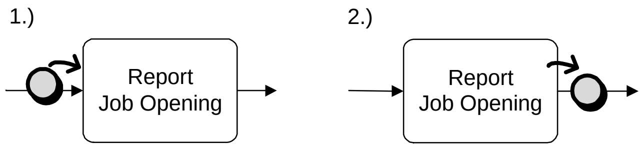

Figure 3: An activity receives a token and forwards it after completion

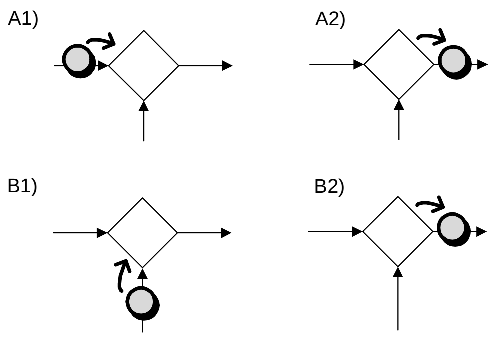

Figure 4: Routing of a token by a merging exclusive gateway

Every time the process is started, the start event creates a token (cf. figure 2). Since the job posting process is carried out more than once, many tokens can be created in the course of time. Thereby it can happen that the process for one job posting is not yet finished, when the process for posting another job starts. As it moves through the process, each token is independent from the other tokens' movements.

The token that has been created by the start event moves through the sequence flow to the first activity. This activity receives a token, performs its task (in this case it reports a job opening), and then releases it to the outgoing sequence flow (cf. figure 3).

The following activity forwards the token, too. It then arrives at the merging exclusive gateway. The task of this gateway is simple: It just takes a token that arrives via any incoming sequence flow and moves it to the outgoing sequence flow. This is shown in figure 4. In case A, a token arrives from the left, in case B from below. In both cases, the token is routed to the outgoing sequence flow to the right.

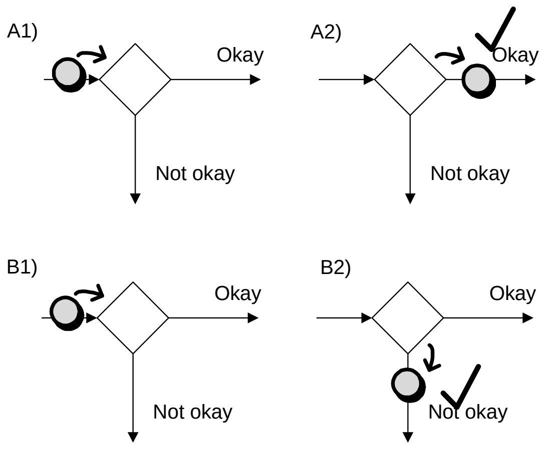

Figure 5: Routing of a token by a splitting exclusive gateway

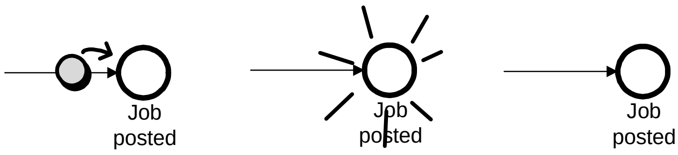

Figure 6: An end event removes an arriving token

The task of the splitting exclusive gateway is more interesting. It takes one arriving token and decides according to the conditions, to which sequence flow it should be moved. In case A in figure 5, the condition "okay" is true, i.e. the preceding review activity has produced a positive result. In this case, the token is moved to the right. Otherwise, if the condition "not okay" is true, the token is moved to the downwards sequence flow (case B).

The modeler must define the conditions in such a way that always exactly one of the conditions is true. The BPMN specification does not state how to define conditions and how to check which conditions are true. Since the considered process is not executed by software, the rather simple statements used here are sufficient. Otherwise, it would be necessary to define the conditions according to the requirements and rules of the software tool.

The token may travel several times through the loop for reworking the job posting. Finally, it arrives at the end event. This simply removes any arriving token and thus finishes the entire process (figure 6).

The sequence flow of every process diagram can be simulated in this way with the help of tokens. This allows for analyzing whether the flow logic of a process has been modeled correctly.

It should be noted that a token does not represent such a thing as a data object or a document. In the case of the job posting process, it could be imagined to have a document "job posting" flowing through the process. This document could contain all required data, such as the result of the activity "Review Job Posting". At the splitting gateway, the decision could then be based on this attribute value. However, the BPMN sequence flow only represents the order of execution. The tokens therefore do not carry any information, other than a unique identifier for distinguishing the tokens from each other. For data objects, there are separate BPMN constructs which will be presented in chapter 10.

2.4 Presentation Options

Usually, pools are drawn horizontally. The preferred direction of sequence flow is then from left to right. On the other hand, it is also possible to use vertical pools and to draw the sequence flow from top to bottom, as in the example in figure 7.

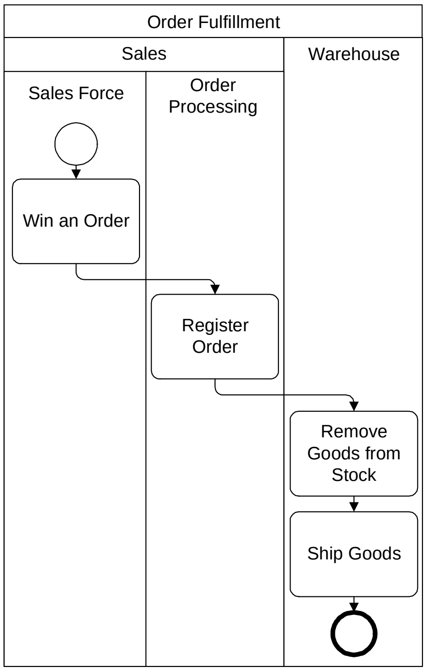

Figure 7: Vertical swimlanes and nested lanes

It makes sense to decide for only one of these possibilities – horizontal or vertical. Nevertheless, there are modeling tools which only support horizontal modeling.

Figure 7 also shows an example of nested lanes. The lane labeled "Sales" is partitioned into the two lanes "Sales Force" and "Order Processing". In principle, it is possible to partition these lanes again, and so forth, although this only makes sense up to a certain level of depth.

It is not prescribed where to put the names of pools and lanes. Typical are the variants selected for figure 1 and figure 7. Here the names are placed on the left of the pools or lanes, or at the top for the vertical style, respectively. The name of a pool is separated by a line. The names of the lanes, however, are placed directly in the lanes. A separation line is only used for a lane that is partitioned into further sub-lanes.

Lanes can also be arranged as a matrix. The procurement process in figure 8 runs through a business department and the procurement department, both of which span a branch office and the headquarters. When a demand occurs in a branch's business department, this department reports the demand. In the next step, the procurement is approved by the same department in the headquarters. The central part of the procurement department then closes a contract with a supplier, followed by the branch's purchasing department carrying out the purchase locally.

12 Conversations

12.1 Conversation Diagrams

A conversation diagram provides an overview of which partners of a certain domain co-operate on which tasks. In figure 168, three conversations can be seen. When processing an order for an advertisement, one customer works together with one advertising agency and several designers. On the other hand, a customer and an advertising agency can jointly run an advertising campaign. For this, they co-operate with several media. A designer can also be part of another inter-company activity: Together with a publisher, he handles orders for illustrations.

In the end, such a conversation is realized by a series of message flows. The details can be modeled e.g. in a choreography diagram or a collaboration diagram. As an example, the message flow of the conversation "Process Order for Advertisement" is described by the collaboration diagram in figure 159, as well as by the choreography diagram in figure 160. However, it is not required for a collaboration or choreography diagram to specify exactly one conversation. It is also possible to combine the message flows from two or more conversations in one diagram.

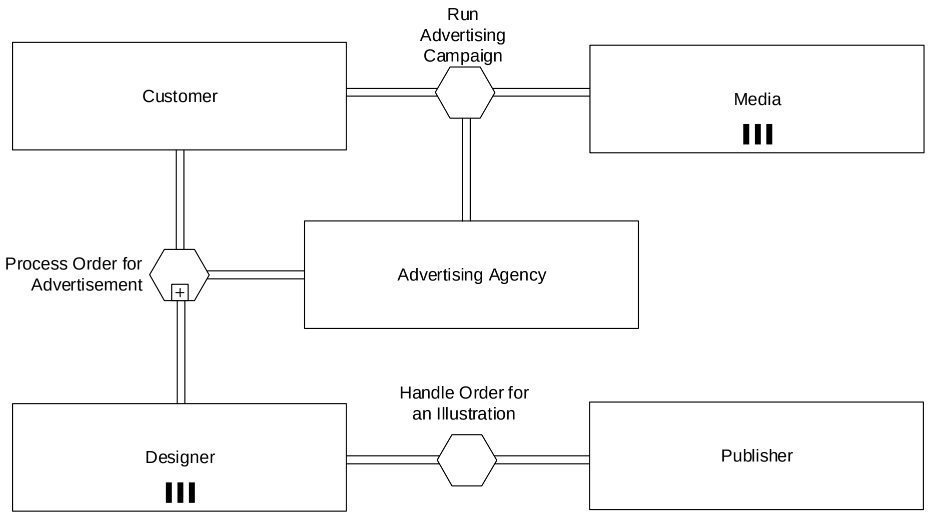

Figure 168: Conversation diagram

12.2 Message Correlations

The contents of the message flows within one conversation are always related to each other. For example, all messages that are exchanged within one instance of the conversation “Process Order for Advertisement” relate to the same advertisement order. It is, therefore, possible to use the order ID for the correlation, i.e. the assignment of messages to a process instance. If a customer receives an advertisement for approval, he can determine the corresponding order – and thus the process instance – based on the order ID. All messages of a conversation have a common correlation.

The connection between a conversation and a participant is called conversation link. A conversation is always connected to two or more participants.

It is possible that there are several partners of the same type involved in a conversation. “Process Order for Advertisement” has exactly one customer and one advertising agency as participants, but multiple designers. Therefore, the designer’s pool contains a multiple marker. However, it is not clear which conversations have several partners of the same type. For example, the participant “Designer” is also connected with the conversation “Handle Order for an Illustration”. Maybe in this conversation, there is only one designer involved. If such information is important, more detailed collaboration or choreography diagrams are required.

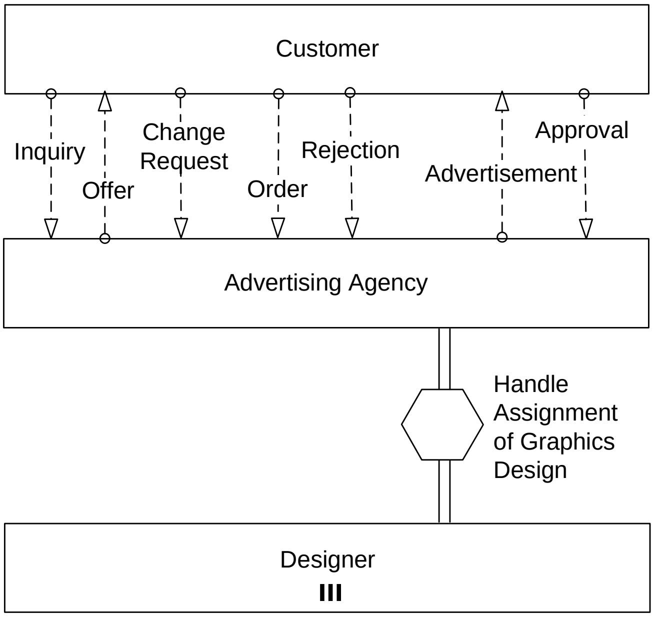

Figure 169: Conversation diagram for sub-conversation “Process Order for Advertisement”

12.3 Hierarchies of Conversations

Conversations can be further detailed using sub-conversations. Similar to sub-processes they are marked with a '+'-sign. The details of a sub-conversation can be described in another conversation diagram. The diagram of a sub-conversation can only contain those participants who are linked to the sub-conversation within the parent diagram.

Figure 169 shows the detailed conversation diagram for the sub-conversation "Process Order for Advertisement" As can be seen from this diagram, it is also possible to draw message flows directly into the conversation diagram. Other than collaboration diagrams, conversation diagrams are not allowed to show processes in the pools or choreographies between the pools.

The diagram contains those message flows that are related to the same order. To be more precise, they relate to the same inquiry. In the beginning, an order has not been placed yet, and not every inquiry turns into an order. Therefore, the common reference point is the inquiry.

Besides the explicitly displayed message flows between customer and advertising agency, the diagram also contains the conversation "Handle Assignment of Graphics Design". All message flows of this conversation are also related to the same inquiry, but this information is not sufficient for the advertising agency in order to assign all incoming messages correctly. This is because availability requests are sent to several designers. The advertising agency has to assign each incoming availability notice to the correct availability request. Thus, additional information is required for correlating these messages, e.g. the IDs of the availability requests. Therefore, it is possible to define a separate Conversation for the message flows between the advertising agency and the designers.

The message exchanges of this conversation can also be modeled in a collaboration diagram (figure 170) or a choreography diagram (figure 171). Of course, it is also possi-

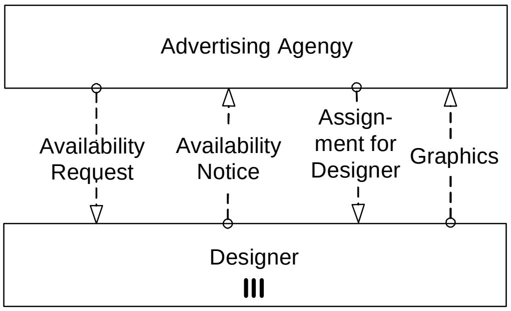

Figure 170: Collaboration diagram for conversation "Handle Assignment of Graphics Design"

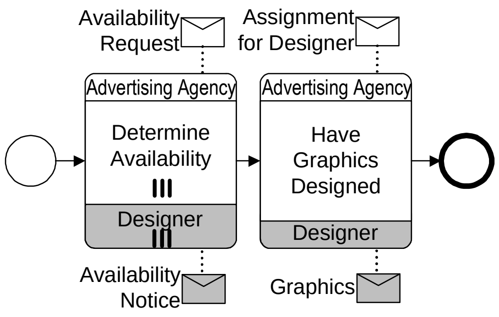

Figure 171: Choreography diagram for conversation "Handle Assignment of Graphics Design"

ble to show the message flows of the entire sub-conversation within a single diagram (figures 159 and 160 in the previous chapter).

Like sub-processes, sub-conversations can also be expanded, i.e. the hexagon is enlarged, and the detailed conversation is shown in its interior. However, it is graphically not easy to include, for example, the contents of figure 169 into an expanded sub-conversation in figure 168. Unfortunately, the BPMN specification draft does not contain any examples for expanded sub-conversations either.

12.4 Calling Global Conversations and Collaborations

Similar to processes and choreographies it is possible within conversation diagrams to call global conversations. A global conversation is defined independently from the conversation diagram from which it is called. A global conversation is specified by its own, independent conversation diagram. The border of a calling conversation is drawn with a thick line (figure 172).

Since a called conversation is defined elsewhere, it may be necessary to map its participants and correlation information to the participants and correlation information of the conversation diagram from which it is called. This, however, is mainly a topic for automating inter-company processes. In business-oriented diagrams, such mappings can be deduced from the context, or they are explained by annotations.

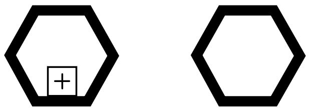

Figure 172: Calls of a collaboration (left) and a global conversation (right)

14 BPMN Modeling Patterns

Process modelers repeatedly encounter similar situations and problems. Modeling patterns are proposals how to model such recurring cases. Instead of developing a specific solution every time, it is often possible to re-use existing and proven solutions. A common patterns catalog in an enterprise helps the modelers to represent the same things always in the same way. This makes the models better understandable.

It is therefore recommended to develop such collections of patterns and to continuously add new patterns that are detected in daily modeling. Depending on the application domain and the modeling purpose, there may be very different kinds of patterns.

In the following paragraphs, some general modeling patterns are presented. These patterns address problems that are relevant to many companies. Some of these patterns have been developed together with BPMN trainers from AXON IVY AG, Switzerland.

14.1 Four Eyes Principle

The four eyes principle is used for important documents, letters, proposals, etc. They must not be created and released by the same person. Instead, the item needs to be reviewed by someone else. This policy helps to ensure that company guidelines are followed, that errors are detected at an early stage, and that fraud is prevented.

The application of this principle in a process is easy to model (figure 175). After a document has been written by the author, it is reviewed by another person. If this person approves the content, the document is released. Otherwise, the author reworks the document, before it is reviewed again.

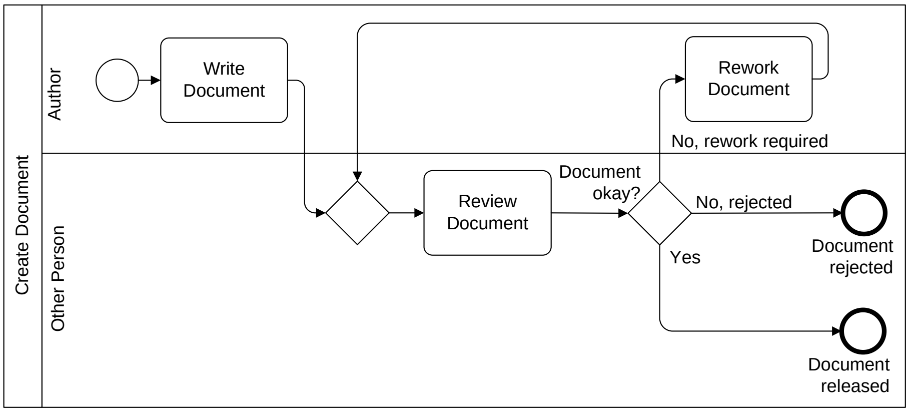

Figure 175: Four eyes principle with cancelation

Instead of a document, the created object can also be a proposal, a contract, a calculation, or something similar.

In this pattern, it is important that the two roles represented by the lanes actually need to be performed by different persons. In many other processes it is perfectly acceptable that one single person performs two or more roles, but here this must be prevented. Therefore, the lower lane explicitly has been labeled with "Other Person". When this pattern is used in a specific process in which the lanes need to have other labels (such as "Developer" and "Quality Inspector"), an annotation can be used for documenting that these roles must be performed by different persons.

At a closer look, the model in figure 175 has a shortcoming. If the author and the reviewer do not eventually agree that the document can be released, the two tasks "Review Document" and "Rework Document" are repeatedly carried out in an endless loop. In practice, the participants will quit this loop after a while – although this is not explicitly defined in the model.

To model this more precisely, a third exit can be added to the splitting gateway. The sequence flow from this exit leads to a second end event that marks the unsuccessful outcome. This is shown in figure 176. Here, the other person can decide that the document is entirely rejected. It would also be possible to let the author decide whether he wants to withdraw the document. This could be modeled with another gateway after "Rework Document" with one exit leading to an end event.

This pattern can easily be extended. For example, the four eyes principle could be extended to a six eyes principle by adding another review by a third person. This second review can be carried out in parallel to the first review, as it is modeled in the pattern "Parallel Checks" (chapter 14.4). In another variation, a possible disagreement between the author and the reviewer could be solved by routing these cases to a second check that is carried out by a third person.

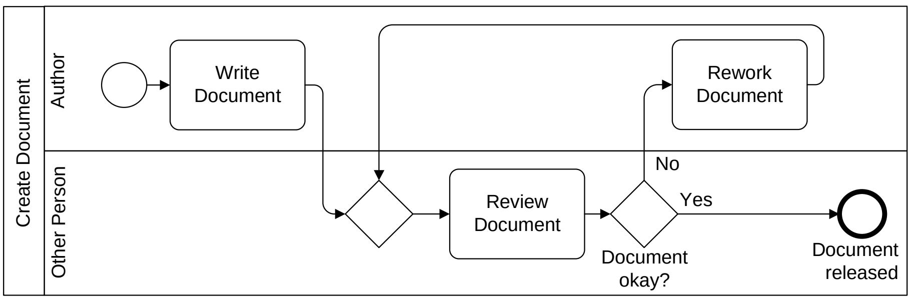

Figure 176: Four eyes principle

14.2 Decisions in Sub-Processes

Sub-processes often have several alternative end events which lead to different paths in the parent process. The pattern "Decision in Sub-Process" makes clear how the selected path is related to the result of the decision that has been made in the sub-process. This pattern has already been used in the discussion of sub-processes in chapter 7.1 (figure 107).

The sub-process can contain any arbitrary flow. The sub-process "Evaluate Proposal" in figure 177 is just an example. For the pattern it is important that each possible end result of the sub-process is represented by a separate end event. If the same results can be created in different parts of the sub-processes, they are combined into one end event. For example, in figure 177 the two "No" branches end in the common end event "Proposal rejected". All end events are positioned near the right border of the sub-process.

In the parent process, the sub-process is followed by a splitting exclusive gateway. This gateway has one exit for each of the sub-process's end events. The labels at the gateway exits indicate which path relates to which end event. In figure 177, the gateway is labeled with a question. Each answer to this question makes clear which result is being referred to. As an alternative, the question at the gateway can be omitted, and the outgoing branches can be labeled with the names of the end events (figure 178).

In both cases, it is also useful to sort the branches from top to bottom in the same order as the end events in the sub-process. Thus, the top branch refers to the top end event, etc.

Instead of a gateway, it is also possible to use conditional sequence flows. In this case, the sequence flows that are related to the sub process's end events start directly at the sub-process border (figure 179). If the expanded view of the sub-process is shown in

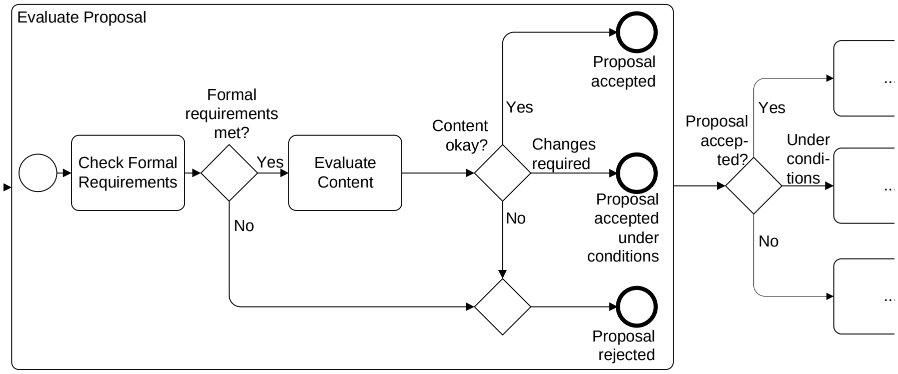

Figure 177: For each of the sub process's end events, there is one corresponding exit at the exclusive gateway

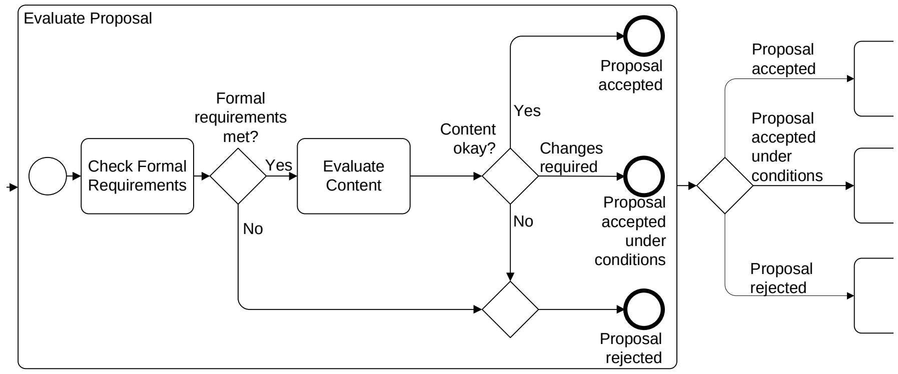

Figure 178: The gateway's exits are labelled with the names of the sub-process's end events

the diagram, the start of each sequence flow can be drawn directly next to an end event, so that it looks like the continuation of the respective sequence flow. Although the tokens in the sub-process and those in the parent process are different ones, there is a tight relation between them. The graphical layout reflects this relation.

14.3 Tasks with Multiple Actors

Typically, every task is performed by one actor. This is modeled by placing the task in the actor's lane. Sometimes, however, a task is jointly performed by several actors. This is difficult to model in BPMN since an activity can only be placed in one lane. It is not allowed to draw an activity symbol in a way that it spans several lanes.

There are several possible solutions to this problem (cf. [Chinosi 2012]). Two of these approaches are discussed in this section.

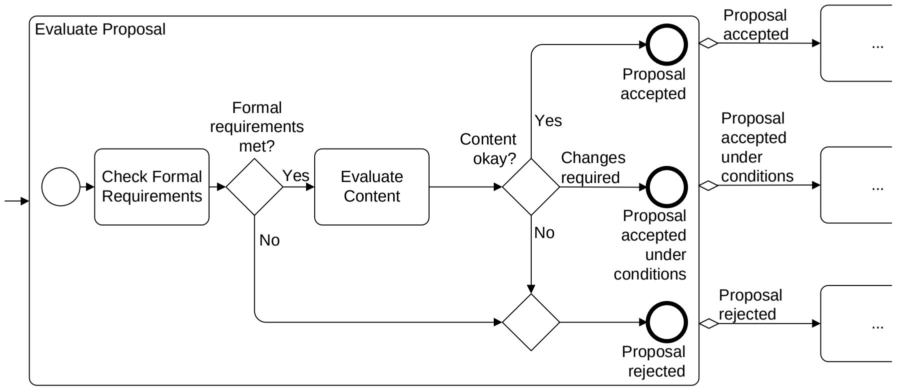

Figure 179: Decision in a sub-process, followed by conditional sequence flows

The Author

Thomas Allweyer studied Engineering at Stuttgart University and Brunel University (West London). He earned his doctoral degree at the Institue for Information Systems at the University of Saarland (Saarbrücken, Germany).

At IDS Scheer AG (now a division of Software AG) he was a product manager for the ARIS modeling tools and a consultant. After that, he became process manager at emaro AG, a joint venture of Deutsche Bank and SAP. Currently, he is a professor for enterprise modeling at Hochschule Kaiserslautern (University of Applied Sciences).

Thomas Allweyer is the author of several papers and books on business process management and process automation. Besides his university activities he is also a consultant, and he frequently holds seminars for well-known companies, especially on business process management – and BPMN, of course.

In his weblog he regularly blogs about current developments in business process management (*www.kurze-prozesse.de*, in German).