POLITECNICO MILANO 1863

BUSINESS PROCESS MODELING NOTATION (BPMN): MODELING EXERCIZES

A cura di:
Cinzia Cappiello
Pierluigi Plebani
Monica Vitali

v.1.1 (10 Gennaio 2016)

Cinzia Cappiello is a researcher at the Department of Electronics and Information at Politecnico di Milano. Her research focuses on information systems, primarily on topics related to data quality management and service quality evaluation. She has published several articles on these topics in international journals and conferences. In recent years, her research interests have expanded to other areas such as Green IT and the design of adaptive applications. The full curriculum vitae is available at `http://home.deib.polimi.it/cappiell/`

Pierluigi Plebani is a Tenured Researcher at the Department of Electronics, Information, and Bioengineering at Politecnico di Milano, where he earned his PhD in Information Engineering. He is an adjunct professor for the courses on Information Systems for Management Engineering and Process and Service Design for Computer Engineering. His research interests focus on the design of service-oriented and business-process-driven information systems. Recently, he has studied the role of the Internet of Things in business processes and energy-saving solutions in the Cloud. He is the author of numerous publications in international journals and conferences and co-author of several academic textbooks and EUCIP certification books. The full curriculum vitae is available at `http://plebani.faculty.polimi.it`

Monica Vitali is a Research Fellow at the Department of Electronics, Information, and Bioengineering at Politecnico di Milano, where she earned her PhD in Information Engineering. Since 2012, she has served as teaching assistant for Information Systems courses for the Information Sector (Computer Engineering and Telecommunications) and for Management Engineering. Since 2015, she has been adjunct professor for the Information Systems course for the Information Sector. Her research interests focus on energy efficiency in information systems within distributed cloud environments and on adaptive systems capable of automatically detecting and resolving operational issues. The full curriculum vitae is available at `http://vitali.faculty.polimi.it`

Introduction

Business Process Modeling Notation (BPMN) is increasingly being established as the standard tool for business process modeling. This booklet aims to offer a compendium for quick and easy consultation to learn and understand the fundamental constructs of BPMN. The version referred to in this text is 2.0, whose complete specification is available at `http://www.omg.org/spec/BPMN/2.0`.

Since the BPMN specification is very broad, this first edition of the booklet focuses mainly on process modeling and collaboration diagrams, leaving aside aspects related to process choreography. The text is divided into the following chapters:
Chapter 1: how to model the activities of a process.
Chapter 2: how to define the flow control of a process through gateways.
Chapter 3: how to manage and generate events in a process.
Chapter 4: how processes can interact with each other.
Chapter 5: how to assign a role to data within a process.
Chapter 6: how to manage transactional and compensation aspects.
Chapter 7: self-assessment exercises to verify learning.

The booklet is mainly addressed to students of the *Information Systems* courses for Computer and Management Engineering at the Politecnico di Milano. However, we believe it may also be useful to anyone who wishes to approach the BPMN language and understand the semantics behind its many graphical constructs.

Like Donald Knuth, we too would like to reward all readers with a dollar for every error found. However, having decided to make this booklet freely available, we scale down the reward to a heartfelt thank you.

Milan, December 22, 2015

Cinzia Cappiello
Pierluigi Plebani
Monica Vitali

Acknowledgements

The authors would like to thank in particular Leonardo Bruni for his help in drafting the final exercises. Special thanks also go to Barbara Pernici and Maria Grazia Fugini for their support.

Contents

1 Modeling the activities of a process: tasks and subprocesses

A process can be defined as a set of activities that transform an input into an output. Inputs and outputs can be either tangible or intangible goods. In BPMN, an activity is defined as a unit of work that requires time to be executed. According to BPMN notation, an activity is represented by a rectangle containing the name of the activity itself. This name must be descriptive and must make clear what the activity performs. To express the order in which the activities are executed during the process, arrows connecting the activities to each other are used.

Example 1.1 - The example shown in Figure 1.1 illustrates a first example of a process. Every BPMN process must start with a start event (a circle with a thin line) and end with an end event (a circle with a thick line). For more details on events, see Chapter 3. The relationship that links two activities through an arrow directed from activity A to activity B is the sequential flow of the process and indicates that activity B can only be executed after activity A has finished.

The modeling of a process therefore makes it possible to define which activities compose a process and the order in which they must be executed.

Example 1.2 - A process can describe any set of goal-oriented activities. In Fig. 1.2, for example, a simple process is shown that describes the order in which the activities necessary for preparing a coffee with a moka pot must be carried out.

1.1 Repetition of an activity: loop and multi-instance

BPMN notation makes it possible to define particular types of activities that can be useful to model activities that need to execute their body multiple times before being completed. Two types can be defined, whose symbols are illustrated in Fig. 1.3:

- *Loop activity*: an activity of loop type is executed until a Boolean termination condition is verified or until a maximum number of iterations has been exceeded. In BPMN notation, a loop-type activity is characterized by a circular arrow at the bottom.

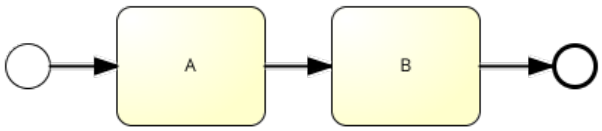


Figure 1.1: *Sequential flow between two activities in a process*

Figure 1.2: *Coffee preparation process*

- *Parallel or sequential multi-instance activity*: an activity of multi-instance type generates a number of copy instances of the activity itself. The number of instances to be created is determined by an expression or is calculated based on process data, but it is fixed at the moment the activity is started. The various instances can be executed in parallel or in sequence. In BPMN notation, a multi-instance activity is characterized by three horizontal (sequential) or vertical (parallel) parallel lines at the bottom.

The two types of activities just described have many points of similarity and can often be used interchangeably. The conditions mentioned in their definition are implicit and do not need to be specified in the BPMN diagram. However, it is possible to attach to the activity an annotation that specifies the termination condition of the loop or the expression that generates the number of multi-instance copies.

Example 1.3 - The process in Figure 1.4 shows the activities involved in handling an order received by an online sales company. The employee who takes charge of an order must first enter its details into the system and then send a request to the warehouse for each product included in the order. He then waits until all products have been retrieved. Each product is then packed, one at a time, and at this point the order is shipped. The employee sends the order receipt to the customer and waits for payment to be received. When payment is received, the order is archived.

The modeled process makes use of loop and parallel multi-instance activity types. The parallel multi-instance is used for the activity *Send Warehouse Request*, which must send a separate request for each individual product included in the order. These requests can be sent simultaneously, and their number depends on the number of products in the order. The loop type, on the other hand, is used in the activity *Pack Product*. In this case, the process specifies that products must be packed one at a time. The activity will therefore be repeated until all products have been packed (loop termination condition).

Figure 1.3: *Loop and multi-instance activities (parallel and sequential)*

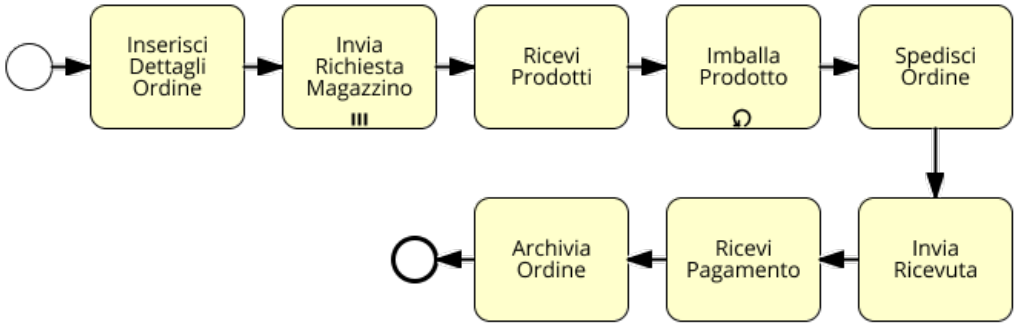


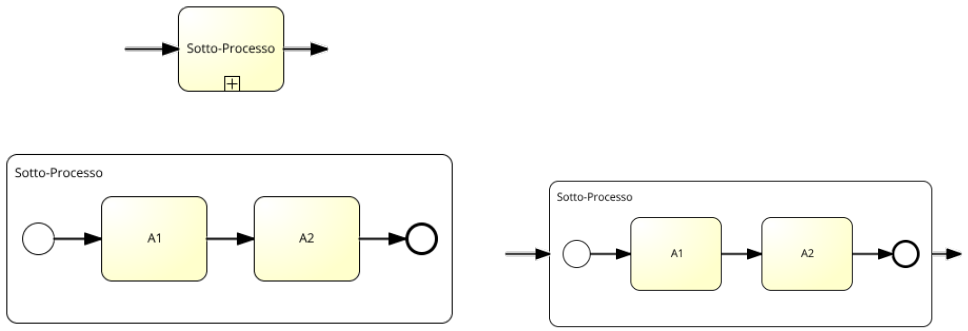
Figure 1.4: *Order management process of a company*

1.2 Subprocesses

We have defined an activity as a unit of work. From the first examples it is possible to see that in reality some activities can be broken down into simpler units that describe in detail how the individual activity must be performed. We can therefore decompose an activity into a set of activities, also characterized by an execution order. Activities that cannot be further decomposed are called atomic activities. Activities that can instead be detailed are called **subprocesses**. Two types of approaches can be used to identify subprocesses:

- hierarchical decomposition: starting from a process activity, I break it down into smaller units that contribute to its realization;
- logical grouping: I group several process activities that together contribute to the achievement of a sub-goal.

BPMN allows these two approaches to be modeled with two different notations shown in Figure 1.5. On the left is the representation of a subprocess obtained by hierarchical decomposition. In the main process, a compressed version of the subprocess is inserted (characterized by a square containing a "+" at the bottom), which is then expanded separately, showing the details of the subprocess itself. On the right, instead, it is possible to see how to represent logical grouping. The subprocess is represented in its expanded version in which the name of the subprocess is shown at the top and within its boundaries are the activities that compose it in the desired order. It is important to note that the flow of the subprocess is isolated from the rest of the process: a subprocess has its own start event and its own end event, and no activity inside it can be connected



(a) Hierarchical representation (b) Functional decomposition

Figure 1.5: *Alternative representations of subprocesses in BPMN*

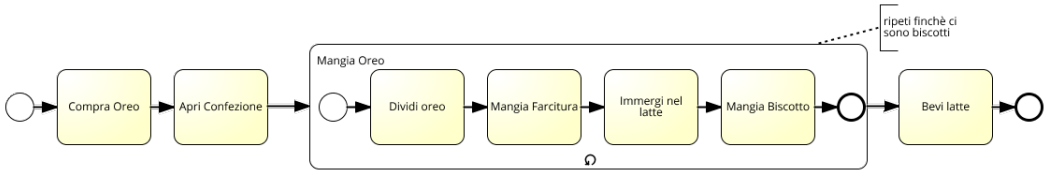


Figure 1.6: *Process describing how to eat a package of Oreo cookies using functional grouping*

with elements outside the subprocess.

Example 1.4 - An example of a subprocess is shown in Figure 1.6. The process describes the order of activities required to eat a package of Oreo cookies. The activities that contribute to achieving the sub-goal of eating a cookie are grouped together in the subprocess *Eat Cookie*, executed in a loop. The termination condition is made explicit using an annotation. The activities that contribute to achieving the sub-goal of eating a cookie are grouped together following the functional grouping approach.

Example 1.5 - The same example is implemented using hierarchical decomposition in Figure 1.7. This approach is convenient when one does not want to place too much emphasis on the subprocess. The main process becomes clearer and more readable in its main parts, leaving the details of the subprocess aside.

Like activities, subprocesses can also be of loop or multi-instance type, indicating that all the activities composing the subprocess are repeated several times based on termination conditions or the defined number of instances. In the case of subprocesses, it is also possible to define other types illustrated below.

1.2.1 Ad-hoc subprocesses

A subprocess can be declared as *ad-hoc*. In this case, the set of activities that compose the subprocess does not have to be executed in a fixed order, but the order and number of executions of each individual activity depend on the performer. Ad-hoc subprocesses

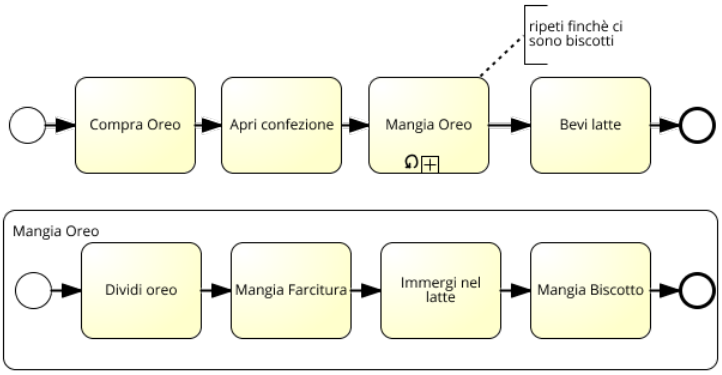


Figure 1.7: *Process describing how to eat a package of Oreo cookies using hierarchical decomposition*

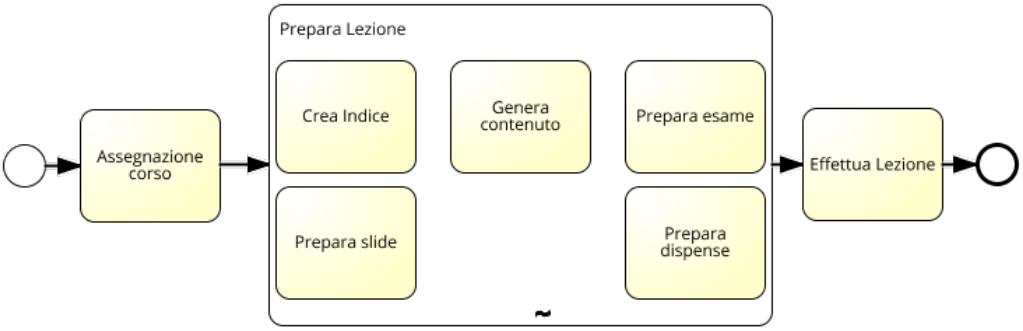


Figure 1.8: *Ad-hoc subprocess describing the activities for preparing a lecture*

are very useful for modeling unstructured parts of the process. Again, the termination of an ad-hoc subprocess depends on an implicit termination condition.

Example 1.6 - Figure 1.8 shows the process of preparing a lecture. In this case, the activities that compose the process do not have to follow a specific order, and the number of times each activity can be executed may vary. For this reason, the set of activities can be represented using an ad-hoc subprocess.[a]

[a] Example taken from Mathias Weske. Business process management: concepts, languages, architectures. Springer Science & Business Media, 2012.

1.2.2 Transactions

Transactions are also a special type of subprocess, represented in BPMN notation by a double border. The activities contained within a transaction, even if complex, are treated as if they were a single atomic activity. This means that in the subprocess, either all activities are successfully executed, or none must be completed. This means that if an activity fails, the effects of the other activities included in the transaction must be undone. This special type of subprocess will be discussed in more detail in Chapter **??**.

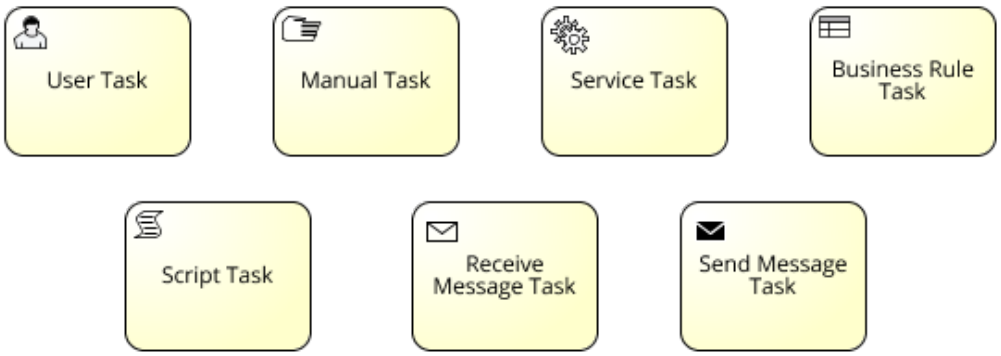


Figure 1.9: *Types of tasks*

1.3 In-depth - Types of tasks in BPMN

With BPMN 2.0 notation, it is possible to define the type of each task through icons in the upper left corner. Several types are defined as shown in Figure 1.9 and are briefly described in this section.

- User Task: activities that require user interaction through software support (e.g., password change);
- Manual Task: activities performed without software support (e.g., installation of equipment);
- Business rule: a business rule is a rule that must be interpreted. A task defined as a business rule provides input to the interpreter and obtains an output when executed;
- Service Task: an activity implemented through software;
- Script Task: a task that uses some form of scripting language. The task is completed when the script ends;
- Receive Message Task: a task that waits for a message from an organization external to the process during its execution;
- Send Message Task: a task that sends a message to an organization external to the process during its execution.

It is worth emphasizing that in BPMN the concept of a message extends not only to the exchange of information but also to the exchange of physical goods. In general, as will be recalled, messages make it possible to model any interaction between the process and an external entity.

2 Modeling alternative flows: gateways

The processes we saw in Chapter 1 are very simple processes in which the activities that are part of the process are executed one at a time sequentially. In reality, business processes are much more complex than a simple concatenation of activities. Often, in fact, the activities to be executed may differ depending on the context in which they are carried out. For example, during a production process, it may be necessary to check whether all products are available in stock and, if not, order them. In other cases, multiple activities can be executed simultaneously, so establishing a sequential execution order as done previously is incorrect. This cannot be modeled with the simple structure we have seen. To overcome this problem, BPMN notation introduces *gateways*.

Gateways are process junctions from which multiple alternative flows branch out or into which they converge. Depending on the type of gateway used, one or more flows can be activated based on the execution context of the process. In BPMN, gateways are represented with diamonds, inside which is a symbol that defines their type. Gateways can be used in two modes:

- *split* mode: a gateway of this type has one incoming flow and multiple outgoing flows;
- *join* mode: a gateway of this type has multiple incoming flows and one outgoing flow.

Gateways can also be used in hybrid mode (simultaneously split and join), but this solution is not recommended. In cases where it is necessary, it is advisable to use a split gateway and a join gateway in succession, or vice versa.

In the rest of the chapter, we will examine the different types of gateways and how they can be used in split and join mode.

2.1 Exclusive Gateway (XOR)

The XOR gateway, or exclusive gateway, is a gateway used when, during process execution, a choice must be made, based on which one set of actions is executed instead of another. This gateway is represented in BPMN with a diamond containing an X and can be used both in split mode and in join mode. Used in split mode, the XOR gateway allows modeling the choice of only one option among several alternatives. On each

outgoing branch from the gateway, a condition is specified, and the first that evaluates to true is the one executed and thus the path followed by the process token. In the XOR gateway, it is also possible to indicate a branch with a default condition, meaning that this branch is executed if none of the conditions on the other branches are verified.

Example 2.1 - On the left side of Figure 2.1 an XOR gateway in split mode is shown. When the token reaches the gateway, it will take one of the three possible flows. The conditions expressed on the branches are evaluated one at a time until a true condition is found. The last branch, instead of a condition, has a line indicating the default option: the branch executed if none of the other conditions are met. Once the branch is selected, the token reaches the activities on that branch, executing them following the process flow. The activities on the other branches will not be executed.

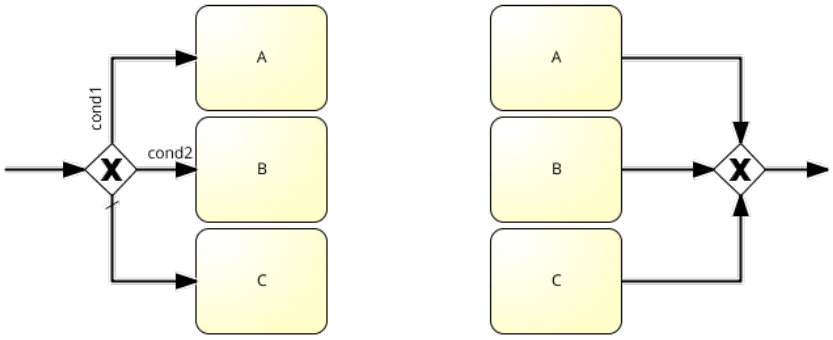


Figure 2.1: *The XOR gateway*

The XOR gateway can be used in join mode. This mode is used to rejoin alternative flows generated by previous gateways in the process.

Example 2.2 - The functioning of an XOR gateway in join mode is shown on the right side of Figure 2.1. When a token arrives at an XOR gateway, it is passed through immediately, without waiting for the completion of the other flows entering the gateway. In this case, no conditions are attached to the different branches, since no choice is being made but rather multiple flows are being rejoined.

The XOR gateway can also be used to model a loop in the process as an alternative to the loop-type activities described in Chapter 1.

Example 2.3 - An example of using an XOR gateway in both split and join mode is shown in Figure 2.2. A company that sells electrical material wants to model a process for handling customer orders. The process begins upon receipt of an order. The employee then enters the order details into the system, and the correctness of the entered data is verified. If the data is correct, availability of the material is checked and a notification is sent to the customer with the estimated delivery time. If the data is not correct, a request for updated data is sent. When the update is received, the process is re-executed starting from the data entry.

The second gateway is used in split mode. The company evaluates whether the

order data are correct, and based on this evaluation, the process token passes either to the availability check if they are correct, or to the update request if they are not. Once the update is received, the two operations of entering details and verifying correctness must be repeated. To unify the two flows pointing to the activity "Enter Order Details," an XOR gateway in join mode is used. This means that whenever a token reaches the gateway, it is immediately passed to the next activity.

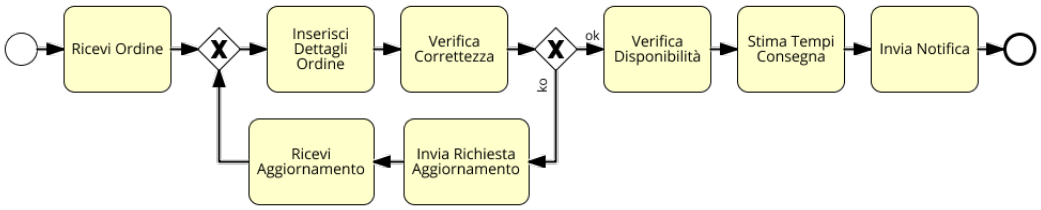


Figure 2.2: *Order management process of a company using the XOR gateway*

2.2 Inclusive Gateway (OR)

The OR gateway, or inclusive gateway, is very similar to the XOR gateway. In BPMN it is represented with a diamond containing a circle. In this case too, the gateway can be used in split mode or in join mode. In split mode, as in the previous case, a condition is attached to each outgoing branch. However, in this case, if multiple conditions are verified, the process token is duplicated and one token is generated for each branch. Therefore, multiple paths can be taken simultaneously. Again, it is possible to have a default condition on one of the branches. The branch with the default condition is executed only if the conditions on all other branches are false, thus ensuring that at least one branch is always activated.

Example 2.4 - On the left side of Figure 2.3 an OR gateway in split mode is shown. When the token reaches the gateway, it is duplicated for each branch whose conditions are verified. The tokens will reach the activities on those branches, executing them even simultaneously, without a predefined order. For example, if conditions cond1 and cond3 are satisfied, activities A and C (and all those following them) will be executed. Although on each individual branch the order in which activities are modeled will be respected, there is no established execution order between the two branches.

The OR gateway can be used in join mode. This mode is used to rejoin alternative flows generated by previous gateways in the process. This use of the OR therefore makes it possible to synchronize the active flows of the process. The join determines how many tokens were originally activated and waits only for those tokens, ignoring the other branches.

Example 2.5 - The functioning of an OR gateway in join mode is shown on the right side of Figure 2.3. The OR gateway waits for tokens coming from all the active incoming branches in the process. When all tokens have been collected, a single

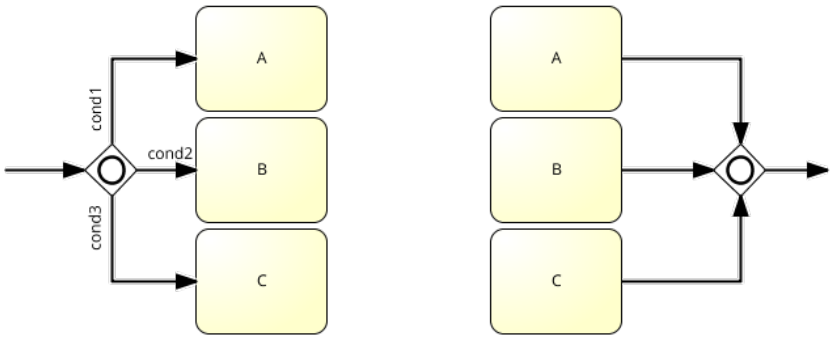


Figure 2.3: *The OR gateway*

token continues along the path through the gateway. If in the example in the figure activities B and C were active, the gateway would wait for the completion of both before continuing with the execution of the process. For this property it enables the *synchronization* of only the active flows in the process instance.

Example 2.6 - An example of using an OR gateway in both split and join mode is shown in Figure 2.4. A bank receives money that must be deposited into its customer's current account. Before the deposit, certain operations must be carried out. If the deposit amount is greater than 10,000 euros, operations for handling large amounts must be performed. If, on the other hand, the deposit comes from a foreign bank, operations to handle the origin of the money must be performed. If neither of the two conditions is verified, the deposit is handled with standard procedures. Once all the necessary operations have been carried out, the deposit record is archived.

The first OR gateway is used in split mode. One or more conditions may be verified. If the deposit is foreign and of a high amount, two tokens are generated and the activities on the first and third branches are executed. If only one of the two conditions is verified, only one token goes to the corresponding branch. If no condition is verified, the token goes to the default branch. Since all operations must be completed before proceeding with the archiving, an inclusive gateway is used as a join to synchronize all the active branches that precede it. When all the active tokens reach it, the flow continues with a single token that reaches the activity "Archive deposit."

2.3 Parallel Gateway (AND)

The AND gateway, or parallel gateway, allows modeling situations where multiple activities can be performed simultaneously or in an undefined order. In BPMN it is represented with a diamond containing a "+". In this case too, the gateway can be used in split mode or in join mode. In split mode, all branches outgoing from the gateway must be executed. Consequently, no conditions are attached to the branches, and a token is always generated for every possible path.

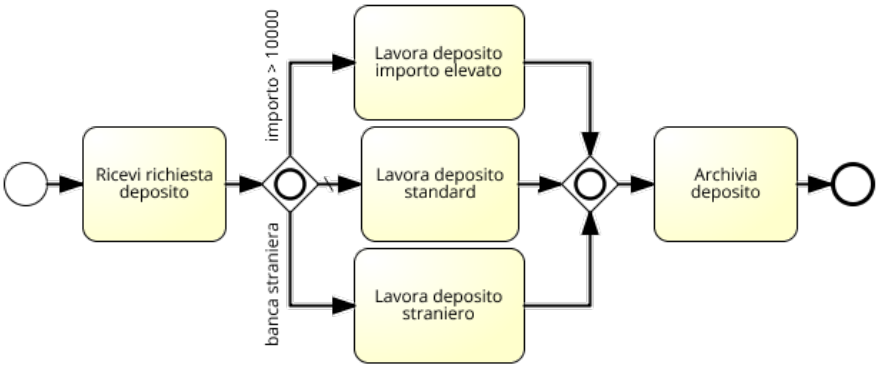


Figure 2.4: *Money deposit management process in a bank*

Example 2.7 - On the left side of Figure 2.5 a parallel gateway in split mode is shown. When the token reaches the gateway, it is duplicated for each branch. The activities on those branches will always be executed, without respecting a predefined order (so also simultaneously). As in the case of multiple active branches with an OR gateway, on each individual branch the order in which the activities are modeled will be respected, while between branches there is no established execution order.

The functioning of a parallel gateway in join mode is shown on the right side of the figure. The gateway waits for tokens coming from all incoming branches. When all tokens have been collected, a single token continues along the path through the gateway. In the example, all activities (A, B, and C) must be completed before the token can proceed. For this property, it enables the *synchronization* of alternative flows.

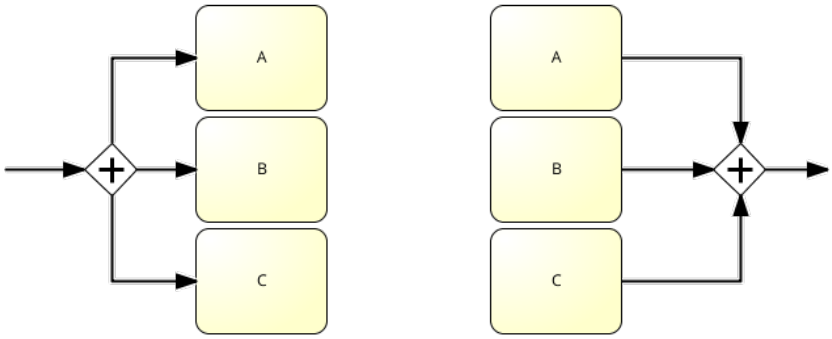


Figure 2.5: *The AND gateway*

Example 2.8 - An example of using a parallel gateway in both split and join mode is shown in Figure 2.6. A company receives an order from a customer. At this point, it must prepare the shipment of the products and send the invoice to the customer. The order in which these activities are carried out is not important. When both have

been completed, the process can continue.

The first AND gateway is used in split mode. All branches are activated simultaneously and, in this specific case, two tokens are generated, activating the two activities on the branches. A parallel join gateway is then used to wait for the completion of both activities before proceeding.

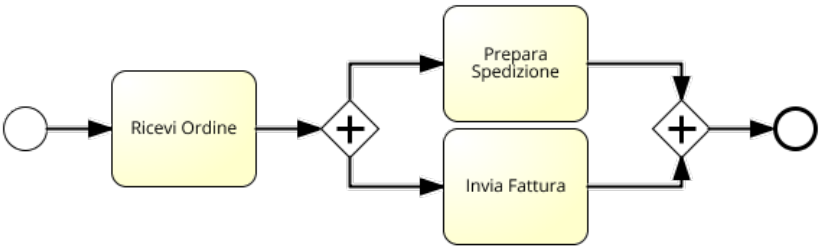


Figure 2.6: *Order preparation process using the parallel gateway*

2.4 Complex Gateway

The complex gateway allows modeling situations in which the activation conditions of the branches are composite. It can, for example, express conditions such as "any pair of branches can be activated." In BPMN it is represented with a diamond containing a "*". In this case, the condition is not attached to the branches but is defined by the user and implicitly represented by the gateway. Since the behavior of the complex gateway is not visually explicit, it should be used with caution in modeling. An example of a complex gateway is shown in Figure 2.7.

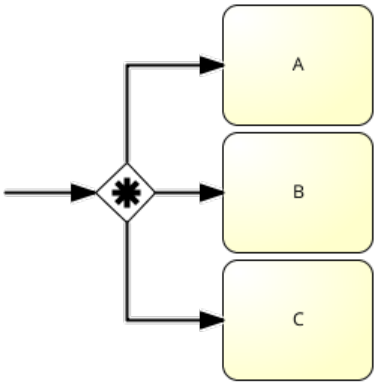


Figure 2.7: *The complex gateway*

2.5 Event-based Exclusive Gateway

The gateways discussed so far in this chapter base the decision of which branch of the process to follow on data internal to the process. In some cases, however, the decision

on which action to take may depend on something that happens externally. For example, in the case of a business process that sends a sales proposal to a customer, the action to be taken after sending the proposal may depend on whether the customer accepts or rejects it. In these cases, where the decision is based on something that happens outside the process, another type of exclusive gateway is used: the event-based gateway. This gateway will be discussed in detail in Chapter 3 when we introduce the concept of an event.

2.6 Summary on Gateways

Gateways allow modeling alternative flows within a process. With the gateway one decides which paths to follow based on certain conditions on the data used in the process. Depending on the type of gateway used, one or more flows may be taken. To rejoin multiple flows, gateways can be used in join mode.

Gateways are elements that do not contain logic within them. They simply route tokens based on certain conditions that must be evaluated on the data present in the process. If the data on which the gateways must perform the evaluation is not explicitly available, it must be calculated before making the decision. It may therefore be necessary to precede an XOR gateway or an OR gateway with an activity that calculates the data on which the conditions will be evaluated. A process that uses different types of gateways to model its flow is described in the following example.

Example 2.9 - A company that supplies electronic equipment receives a request for a quotation from a partner. First, it checks whether the partner has any outstanding debts with the company. If so, the request is rejected. Otherwise, the company checks the availability of the requested equipment and, if it is not in stock, estimates the date when it will be. Meanwhile, while checking availability, it also estimates the time required for shipping. Once these operations are completed, it sends the proposal to the partner.

The modeling of the described process is shown in Figure 2.8. The first XOR gateway allows two alternative paths: if the partner is in debt, the request is rejected; otherwise, the process continues with the preparation of the quotation. Since the estimation of shipping times can be performed independently of the other activities, an AND gateway is used from which two parallel paths depart. Thanks to the use of an XOR gateway, the activity "Estimate Warehouse Times" is executed only if the products are not available. Immediately after, an XOR join re-joins the two alternative paths (only one will be active). Before sending the proposal, the activities on both parallel branches must be completed. It is therefore necessary to insert an AND gateway to synchronize the two flows before proceeding.

Example 2.10 - In Figure 2.9 the process for depositing a check through the ATM of a financial institution is shown. The user goes to the bank to deposit a check. Once authentication is completed, the user chooses the option to deposit a check. At the system's request, the user enters the information about the issuer of the check, the date, and the amount, and inserts the check into the appropriate slot. The system verifies the information. If the data entered by the user does not match the data

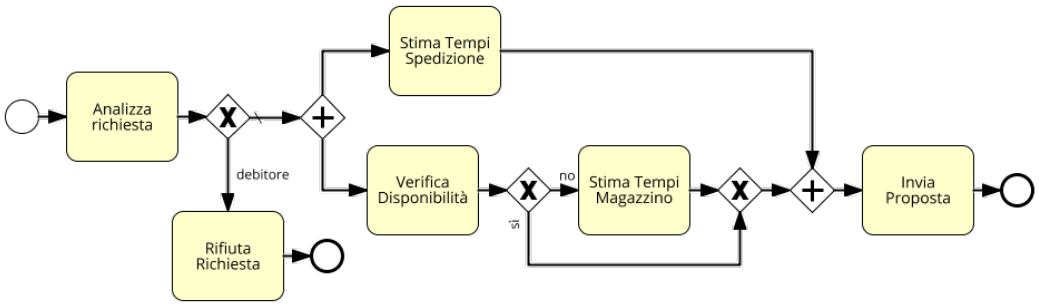


Figure 2.8: *Quotation creation process*

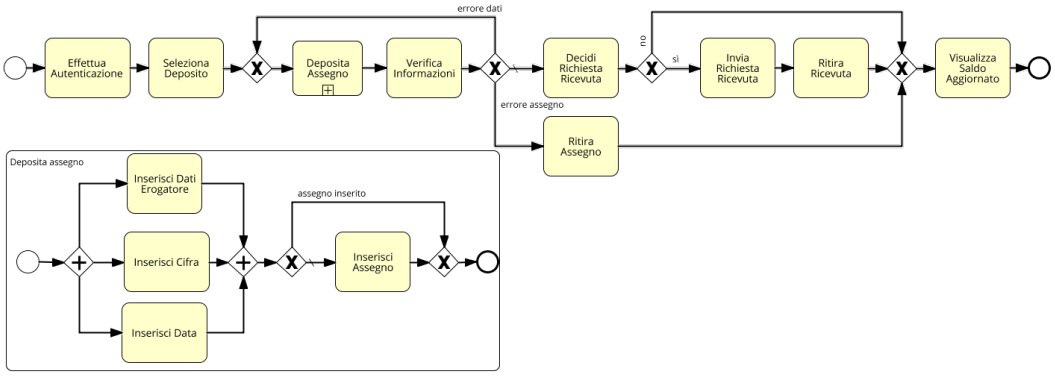


Figure 2.9: *Check deposit process*

read from the check, the system asks the user to re-enter the information. If the check is not considered valid by the system, it is returned to the user, who retrieves it. Otherwise, if everything is correct, the user chooses whether to request a receipt from the system. In that case, the user sends the request and collects the receipt. Finally, in any case, the system shows the updated balance to the user.

The operations necessary for the deposit of the check have been grouped into the subprocess "Deposit Check."

The gateways analyzed in this chapter allow defining the path that a process instance must follow based on information contained within the process itself. In some cases, however, the choice of path may depend on information coming from events that occur during the execution of the process. This case will be addressed in Chapter 3.

3 Events

Through the concept of an event, BPMN allows modeling situations in which the flow of a process is conditioned by external occurrences of various kinds, or modeling situations in which the process itself generates such occurrences. The generic symbol identifying an event is a circle, within which the icon corresponding to the type of event being modeled can be inserted.

According to the BPMN specification, there are three main classes of events:

start event (circle with thin border): events that produce a new process instance. The instance can refer to either a process or a subprocess;

intermediate event (circle with double thin border): events that occur during the execution of the process. They can be inserted within a process flow (intermediate catching/throwing event) or as the boundary of an activity (boundary intermediate event);

end event (circle with thick border): events that define when a process ends and in what way.

Example 3.1 - A simple example of their use is shown in Figure 3.1. In this case, there is a simple process that starts (start event), performs activity A, waits for an event to occur (intermediate event), and after executing activity B, ends (end event). Here, the intermediate event corresponds to a timeout expiration. It is important to note that placing the intermediate event in the process flow forces the process to wait for the occurrence of the event (in this case, the timeout) before executing activity B.

Example 3.2 - A second example, shown in Figure 3.2, illustrates a different semantics of intermediate events dictated by their placement. In this case, the event is positioned on the boundary of an activity (a *boundary intermediate event*) and its occurrence modifies the execution flow. This creates:

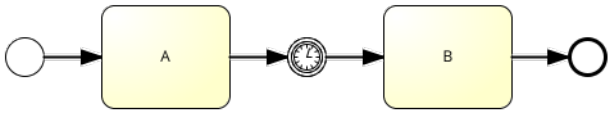


Figure 3.1: *Process with an intermediate event (timer type) in the process flow.*

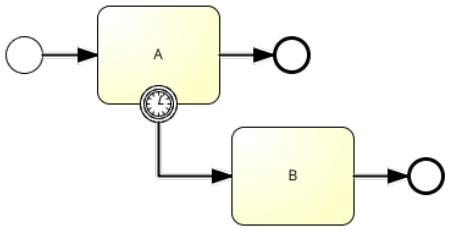


Figure 3.2: *Process with an intermediate event on the boundary of an activity.*

- a *normal flow* (also called *happy flow*), corresponding to the flow executed if the boundary event does not occur. The activity then ends normally and the process terminates;
- an *exceptional flow*, where the event occurs before the completion of the activity, and the process flow continues according to the outgoing transitions from the intermediate event.

In Figure 3.2, the intermediate event represents a timeout expiration. Activity B will therefore be executed only if activity A does not complete before the timeout expires. If activity A completes before the timeout, activity B will never be executed.

3.1 Catching and Throwing Events

Up to this point, events have been used to define an occurrence that is captured (*catching*) and handled during the process. In BPMN, it is also possible to model situations in which the process generates (*throwing*) events. The distinction between the two cases is indicated by the color of the icon inside the circle representing the event. If the icon has no fill color, it identifies a catching event; if it is completely black, it identifies a throwing event.

Example 3.3 - The distinction between catching and throwing can be exemplified in Figure 3.3. Here, the process starts upon receiving a message (catching). Then, activity A is performed, after which the process ends by generating a message (throwing).

It should be noted that intermediate events can be positioned on the boundary of both tasks (atomic activities) and activities containing subprocesses. In the latter case, the subprocess definition can also include the generation of the event, which is then caught at the process level.

Figure 3.3: *Distinction between event catching and event throwing.*

Example 3.4 - An example of this is shown in Figure 3.4, where on the left is a process similar to the one described in Example 4.2, with the only difference that the intermediate event on the boundary is of type *error*: this indicates an error situation occurring within the activity. The details of activity A are shown on the right side of the figure. As can be seen, the subprocess can end in two ways: all information is available, allowing the requested operations to be completed, or some information is missing, so the subprocess ends by generating an error situation (throwing end event).

Through this structure, BPMN allows modeling situations in which occurrences generated at the subprocess level (in this case the error) are handled at a higher level.

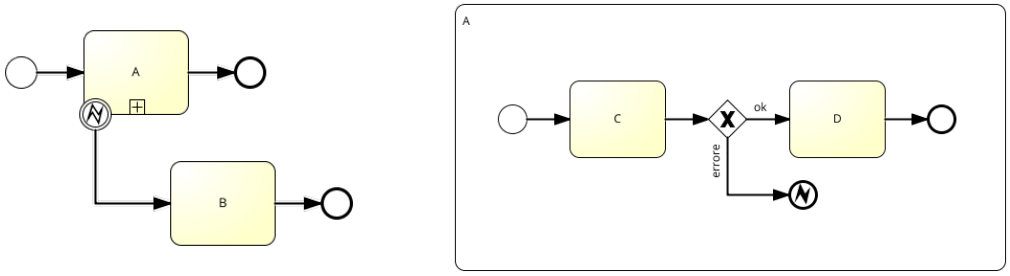
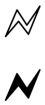


Figure 3.4: *Boundary intermediate event associated with a subprocess.*

3.2 Trigger

In the previous examples, only a few types of events were introduced: timer, message, and error. In reality, BPMN allows defining other types of event *triggers*, for some of which it makes sense to speak of both catching and throwing events, while for others only catching events are possible:

Message		The event is related to any exchange with the outside (e.g., data, material) originating from or addressed to a participant in the process (a pool different from the one containing the message event).

Timer		The event corresponds to any temporal occurrence, such as a timeout expiration (even repeatable) or reaching a specific day or time.
Error		Used to manage error situations during process execution. If an error occurs within a subprocess, it is the containing process that must handle the error.
Conditional		The event is triggered when a condition defined in the process occurs.
Parallel Multiple		Used to indicate situations in which the occurrence of *all* events in a set of events, possibly of different types, triggers the event itself.
Link		Allows modeling jumps within a process flow. Often used to improve diagram readability.
Escalation		Represents an event triggered within a subprocess that must be handled by the containing process. Structurally similar to an error event but does not necessarily indicate an abnormal situation.
Cancel		Used only for transactional subprocess modeling (see Chapter 6) and indicates the need to cancel a transaction.
Compensation		Used only for transactional subprocesses (see Chapter 6) to manage compensation activities in response to a transaction cancellation request.
Signal		Identifies an event generated by a process and broadcast to other processes, or, unlike messages, it can also be sent to the process itself if at a different level.
Multiple		Used to indicate situations where the occurrence of *at least one* of a series of events, even of different types, triggers the event itself.
None		Generic event whose actual type is unknown or does not fall into the previous cases. In this case, the circle symbol of the event contains no icon.

3.3 Start event

A start event defines an occurrence that generates a new process instance, and a process can have multiple start events. The process begins when the first of the events occurs and follows the flow generated by that event. Start events can be of different types as defined in the previous section. An exception is subprocesses, for which only a generic start event is allowed. BPMN allows only catching start events since it does not make sense to start a process by generating an event (throwing).

Example 3.5 - In the left diagram of Figure 3.5, a data backup process can start for two reasons: at midnight, or upon an explicit request. Normally, the backup starts at midnight, generating a new process instance. Assuming the backup takes less than 24 hours, there will never be two parallel instances. If a backup request is received at 11:59, the process starts, but this does not prevent a new instance from starting at midnight.

On the right of Figure 3.5, an *event-based parallel* gateway is used, defining that both events must occur for the process to start: midnight has passed, and a request has been received. Note that the two events do not need to occur simultaneously. If only one event occurs, the process does not instantiate until the second occurs. For example, if the backup request occurs before midnight, the process will not start until midnight.

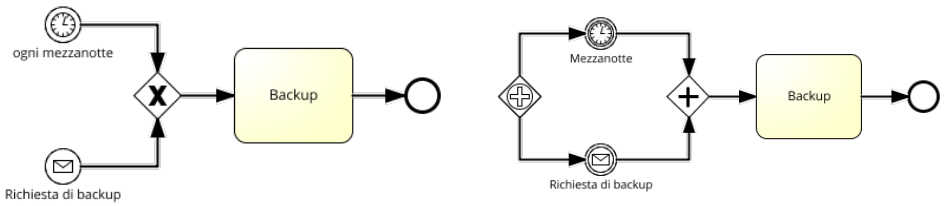


Figure 3.5: *Multiple process start events.*

3.4 End event

An end event is triggered when it receives a token and consumes it to declare the process completed. Depending on the nature of the event, it may generate an event just before the process ends, which can then be consumed by other processes. Thus, unlike start events, end events can only be of type throwing, not catching.

Example 3.6 - Figure 3.6 expands the backup process scenario. Once the backup ends, a signal is generated to be handled by processes listening for that signal. In this case, the relevant processes are: the administration process that requested the backup and the logging process that records the activity. This example also shows the difference between message events (which have a source and destination) and signal events (which are broadcast).[a]

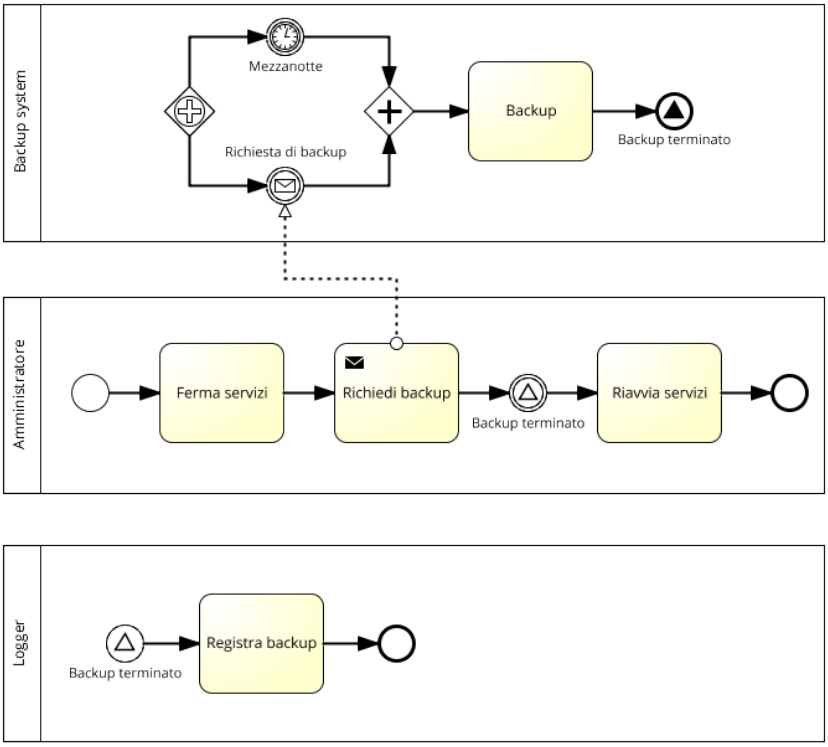


Figure 3.6: *End event with signal throwing.*

[a] This example uses the pool construct, which will be covered in Chapter 4, associating a process with an actor. Here, three processes are managed by the backup system, the administrator, and the logger, respectively.

It should be noted that end events usually consume only the token they receive. Therefore, if multiple branches are active, the remaining branches continue execution. Reaching an end event does not necessarily terminate a process instance. A process instance ends only when all tokens generated for it have been consumed.

Example 3.7 - In Figure 3.7, a process with two parallel branches each ending in an end event is modeled: one message type and one none type. The start event generates a token, the parallel gateway consumes it to generate two tokens (one per branch), and each end event consumes the token of its branch. The process ends only when execution reaches both end events.

A variant of this behavior is the terminate end event. This end event consumes the token it receives and terminates the entire process, automatically canceling any ongoing activities in other branches.[1]

Example 3.8 - Figure 3.8 shows a process behaving differently depending on whether activity A or B finishes first. If A finishes first, the message end event consumes the token and sends the message. When B also finishes, the token consumed by the terminate end event is the only remaining active token, and the process ends. If B

[1] As discussed in the transactions section, the terminate end event does not trigger any compensation requests.

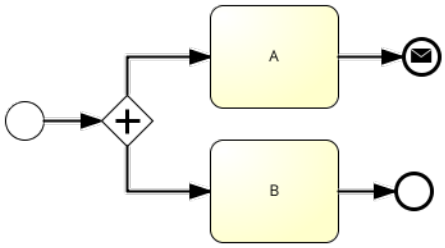


Figure 3.7: *Parallel end events.*

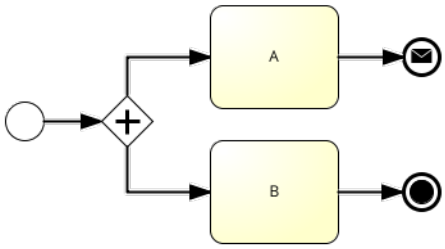


Figure 3.8: *Parallel end events with forced instance termination.*

finishes first, execution of the terminate end event immediately stops activity A, so the message is never sent.

In processes including subprocesses, the scope of an end event is limited to the process level it belongs to. This is particularly important for terminate end events. If a terminate end event is placed inside a subprocess, it terminates only the subprocess, not the entire process.

Example 3.9 - Figure 3.9 shows a process in which a task is a subprocess identical to the previous example. Once task S executes, the subprocess starts and, regardless of how it ends (mail end event or terminate), execution continues with task F.

When specifying a subprocess, end events can also communicate to the parent process how the subprocess ended. This is possible using end events of type error, escala-

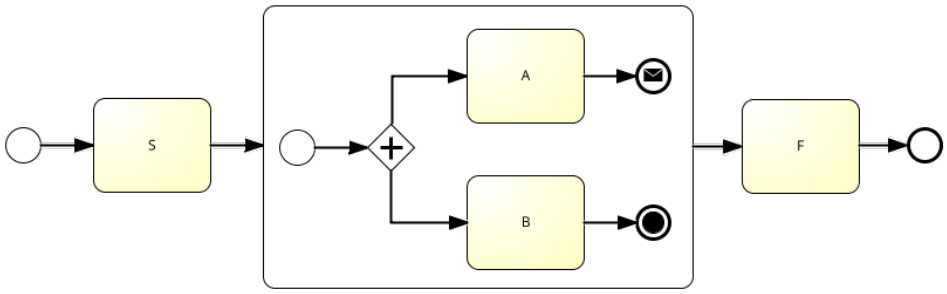


Figure 3.9: *Terminate end event inside a subprocess.*

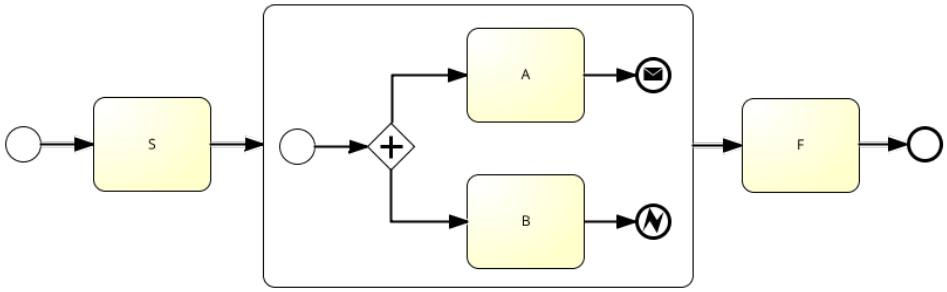


Figure 3.10: *Error end event inside a subprocess.*

Figure 3.11: *Intermediate event of type throwing.*

tion, cancel, compensation, or signal.

> **Example 3.10 -** Modifying the previous example slightly, the subprocess in Figure 3.10 can terminate by sending a message or signaling an error (instead of terminate). In the latter case, error handling can be delegated to the higher-level process, as will be described in the next section using boundary intermediate events.

3.5 Intermediate event

The class of intermediate events is very rich and allows handling many situations that may occur during the execution of a business process. As illustrated earlier, the semantics of this type of event differs depending on whether the event is placed in the flow of execution or on the boundary of an activity: if the intermediate event is placed in the flow, the process continues only upon the occurrence of the event; if the intermediate event is placed on the boundary of an activity, an exception flow is triggered, and depending on whether the event is blocking or not, the activity may continue execution.

When an intermediate event is placed in the flow, it is possible not only to catch an event but also to generate an event (throwing). In this case, the process does not wait for the occurrence of an event; rather, the event is generated, and the process can continue afterwards.

> **Example 3.11 -** As shown in Figure 3.11, once activity A is completed, a message is sent and the process continues with activity B. Note that activity B starts as soon as the message is sent. There is no check to verify whether the message is actually received before executing activity B.

In the case of boundary intermediate events, this class of events is very useful for modeling dependencies between processes and subprocesses. Special situations may occur within subprocesses that cannot be handled locally and must be managed at

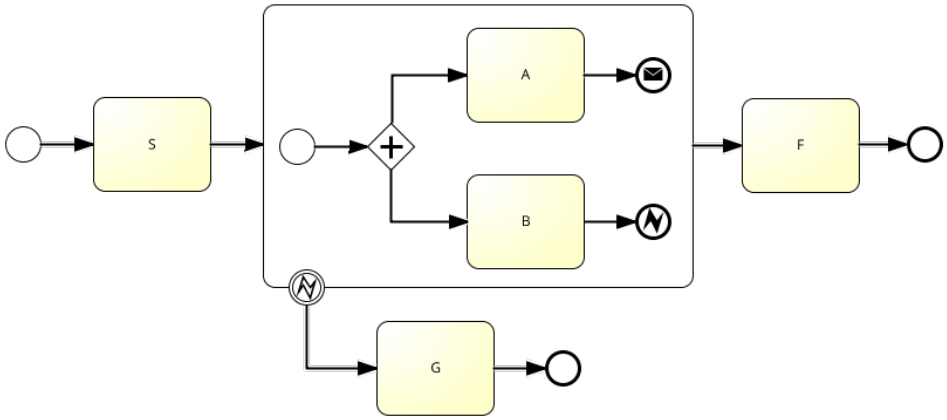


Figure 3.12: *Intermediate event generated inside a subprocess.*

higher levels. Therefore, throwing events (both intermediate and end types) at the subprocess level can be caught at the process level and managed. As mentioned earlier, this is possible only for certain triggers: error, escalation, cancel, compensation, and signal.

Example 3.12 - Figure 3.12 completes the modeling of the process in Figure 3.10 by using a boundary intermediate event that can catch any error generated inside the subprocess. In case of such an error, the subprocess terminates, while the higher-level process follows the exceptional path, which requires executing activity G.

3.6 Event-based exclusive gateway (event-based XOR)

Now that the meaning and role of events in BPMN have been clarified, we can fully describe the event-based exclusive gateway briefly mentioned in Chapter 2.

This gateway is allowed only in split mode: it cannot have more than one incoming flow. Moreover, for each outgoing flow, there must be an associated catching intermediate event.

With this configuration, when one of the events associated with the outgoing flows occurs, execution continues along that specific path. Unlike a traditional XOR, which selects the outgoing flow based on known data, the event-based XOR selects the outgoing path based on the occurrence of an event.

Example 3.13 - An example of using an event-based XOR is shown in Figure 3.13. The modeled process involves sending a request to another actor (not shown), and then the process waits for one of the three events linked downstream of the event-based XOR. According to the modeled process, the recipient actor can react in three ways: accept the request, reject it, or not respond.

Since this is an exclusive gateway, only one of the events can occur, and execution continues only along the branch associated with that event, while events on other branches are invalidated. If the customer accepts, the process records the response and ends. If the customer does not respond, the process closes at the end

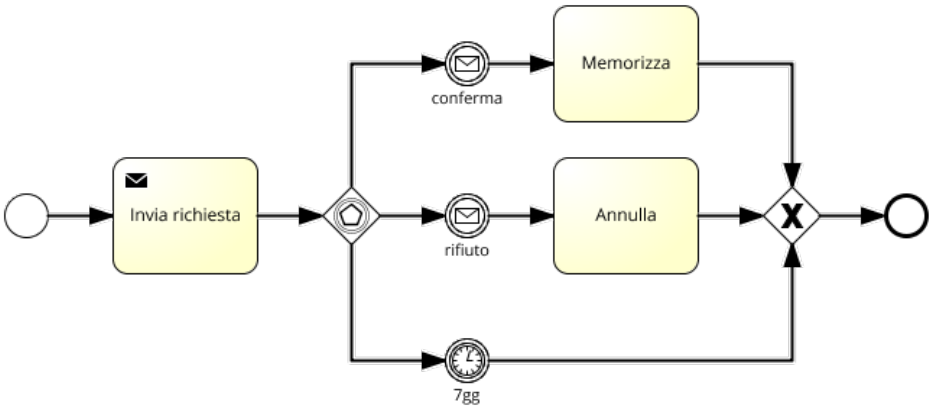


Figure 3.13: *Event-based XOR.*

> of the seventh day. If an acceptance arrives on the eighth day, it is ignored because execution has already passed the gateway handling the events.

The exclusive event-based XOR can also be used to discriminate between events that trigger the start of a process. Referring to Example 3.5, on the right side of Figure 3.14, an alternative way of modeling the process on the left is shown using an event-based XOR gateway. Graphically, in this case, the event-based XOR symbol has a single circle inside, unlike the double circle used for intermediate events. The process modeled starts when one of the two events downstream of the gateway occurs, then executes the backup activity.

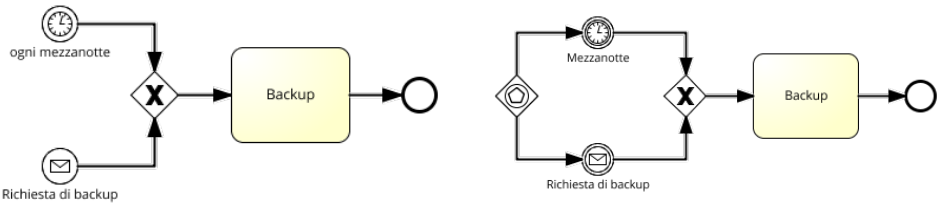


Figure 3.14: *Multiple process start events.*

3.7 Blocking and Non-Blocking Events

Orthogonally to the classification into start, intermediate, and end events, and their types, events can also be distinguished as:

blocking events (circle with solid border)

non-blocking events (circle with dashed border)

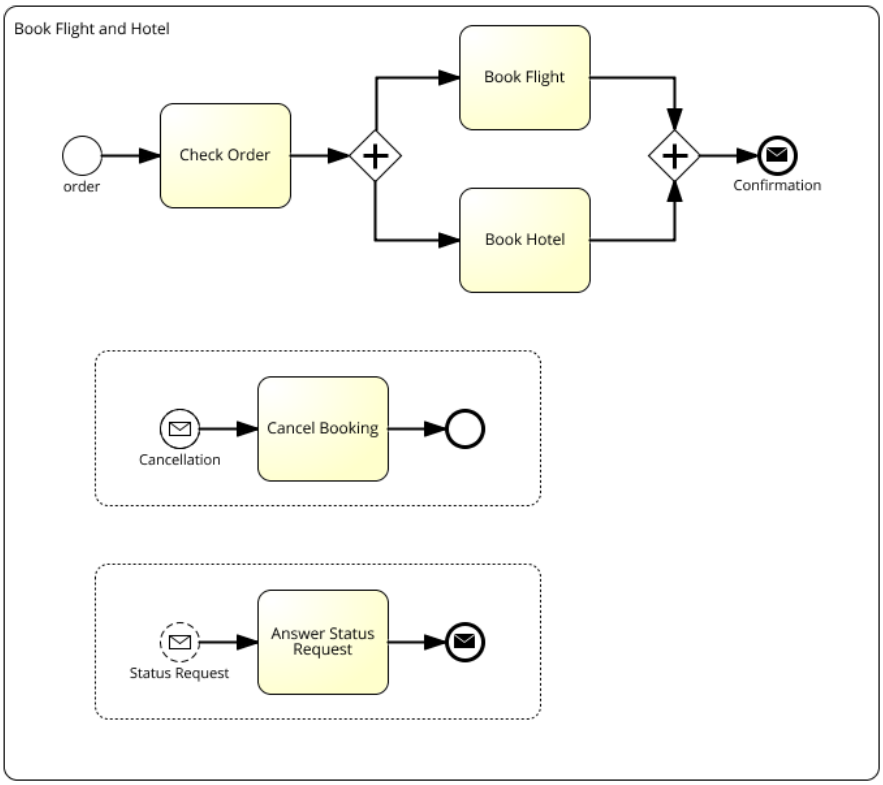


Figure 3.15: *Blocking and non-blocking events*

This distinction is only applicable to start events and intermediate events; it does not make sense for end events, which are blocking by definition.

For start events, a blocking or non-blocking event determines how, within a subprocess, different flows may be initiated and is used in the so-called event-based subprocesses.

Example 3.14 - As shown in Figure 3.15[a], the travel booking subprocess is activated upon the arrival of an order, thus starting the corresponding booking flow. During this flow, a cancellation request message may arrive. Being a blocking start event, it interrupts the ongoing flow and activates the cancellation flow. Conversely, if a status request message arrives during the booking flow, it activates the status notification subprocess without interrupting the ongoing booking flow, as it is a non-blocking start event.

[a] Source: http://en.bpmn-community.org/tutorials/6/

It is important to note that a trigger can be specified for a start event only in the main process or an event sub-process, but not for a subprocess. Subprocess activation must always be linked to the completion of a preceding activity at the higher level. This ensures that event sub-processes manage special situations only when the subprocess is already running.

For intermediate events, only boundary events can be non-blocking. It is therefore

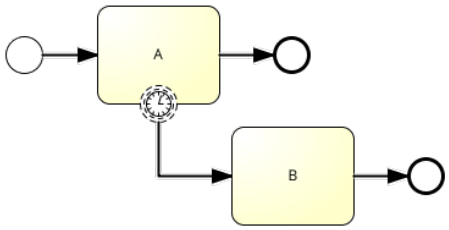


Figure 3.16: *Non-blocking intermediate event on boundary*

not possible to define a non-blocking intermediate event in the main process flow.

Example 3.15 - Figure 3.16 shows the same structure as Example 3, but the intermediate event is now non-blocking (double dashed border) rather than blocking (double solid border). Being a boundary event, both the normal flow and the exceptional flow are defined. Unlike Example 3, the non-blocking event allows both flows to remain active when the timeout occurs; thus, executing activity B does not prevent the completion of activity A. If activity A finishes before the timeout, the exceptional flow will never be activated.

To conclude this overview of events, Figure 3.17[2] summarizes all the event types that can be inserted in a BPMN diagram.

It is important to note that not all types of events can be associated with all triggers. For instance, a timer trigger makes sense for start and intermediate events but not for end events. Similarly, a cancel trigger cannot initiate a process.

Example 3.16 - As a final example, Figure 3.18[a] illustrates a procurement process. On the left, the process defines the purchasing policy, while on the right, the actual procurement activities are detailed. If the Procurement task encounters no errors, the process completes normally. If the item is unavailable, it is removed from the catalog, modeled as a throwing error event. A third scenario involves a non-blocking escalation event signaling a delay in availability without preventing normal completion.

The right side of the figure shows the Procurement task in detail. Depending on supplier availability, three situations may occur: (i) available within 2 days, (ii) available but delayed, (iii) unavailable. In cases (ii) and (iii), throwing events inform the higher-level process. The error event (end type) terminates the task, triggering the exceptional flow, whereas the intermediate escalation event notifies the process of late delivery without interrupting it. The corresponding non-blocking boundary event activates the task to inform the customer without closing the procurement process, allowing its internal execution to continue.

[a] Example from "Object Management Group, BPMN 2.0 by Example, June 2010, `http://www.omg.org/spec/BPMN/20100601/10-06-02.pdf`"

[2] Source: http://blog.goodelearning.com/bpmn/common-bpmn-modeling-mistakes-best-practices-basic-events/

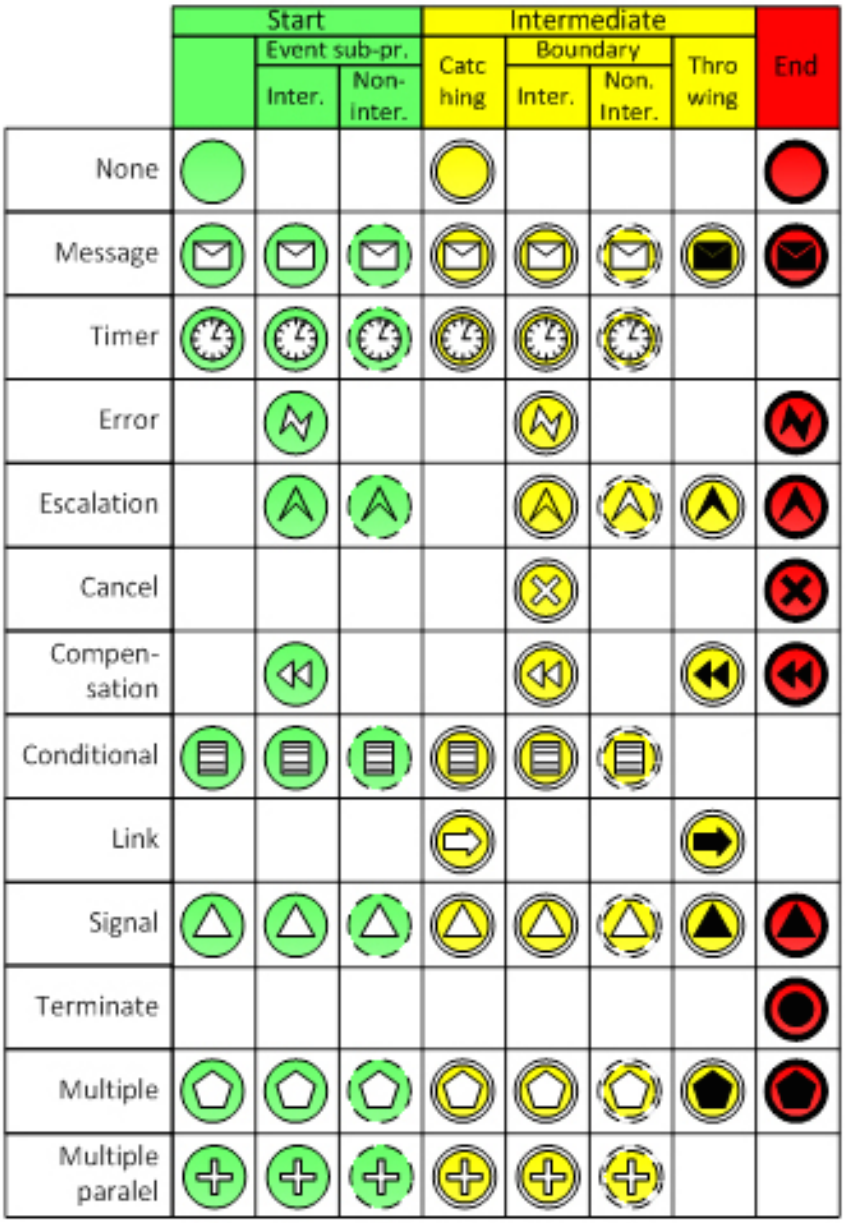


Figure 3.17: *Allowed event types in BPMN*

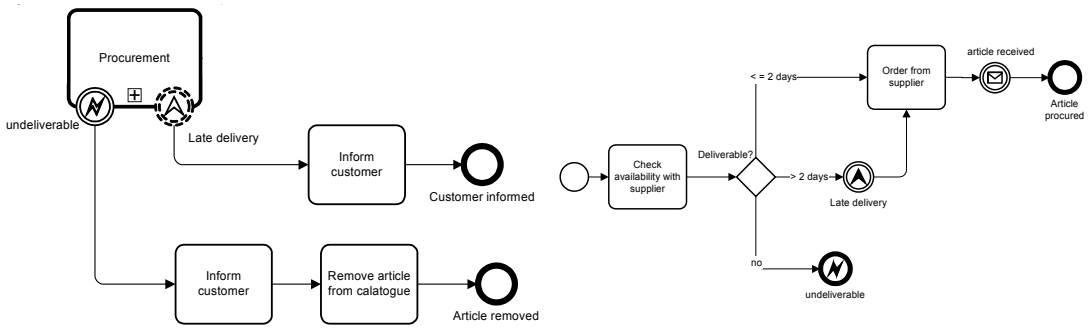


Figure 3.18: *Stock management process and procurement subprocess.*

4 Collaboration between processes: pools and lanes

So far, the BPMN constructs and examples presented have focused on modeling a single process. This process consists of various activities or is organized into subprocesses. Through gateways, it is possible to organize the control flow and capture or generate events.

However, defining a single process is often not sufficient to model a business process: it is necessary to consider the context in which it operates. This context is defined by other processes executed by other organizations, with which the process being defined often exchanges goods and information.

4.1 Collaboration and pools

In BPMN terminology, the modeling of interdependent processes is called *collaboration*. As defined by the BPMN specification, a collaboration diagram is composed of two or more participants, the message flows between them, and optionally, the processes executed by the participants. A participant is therefore an entity that has its own autonomy and collaborates in the execution of a process, and in BPMN it is represented by a *pool*. A pool is represented as a rectangular element that extends horizontally or vertically in the diagram. A pool can be represented as a *white box* or a *black box*. In the first case, the pool contains the process that the participant is responsible for executing and controlling. It is therefore assumed that the participant also acts as the orchestrator of that process. In the second case, the pool contains no information. This indicates that the activities performed by the participant are either unknown or not relevant for the purposes of modeling the business process.

Once the participants are defined through their pools, the actual collaboration between participants occurs when *messages* are exchanged between them. These messages represent the transfer of goods and information between participants and are indicated by a dashed arrow, with one end circular and the other end triangular.

Example 4.1 - Suppose we want to model the vacation booking process, assuming the use of services provided by a travel agency. Figure 4.1 shows the collaboration diagram, which includes two participants: the client and the travel agency, each rep-

resented with its own pool. The client, whose process is being modeled, is further defined by the sequence of activities they perform. For the travel agency, it is either unnecessary or unknown what activities are performed. What matters for the modeling is that there is an exchange of three messages between client and agency in a specific order: travel date (from client to agency), list of options (from agency to client), and finally the choice of option (from client to agency).

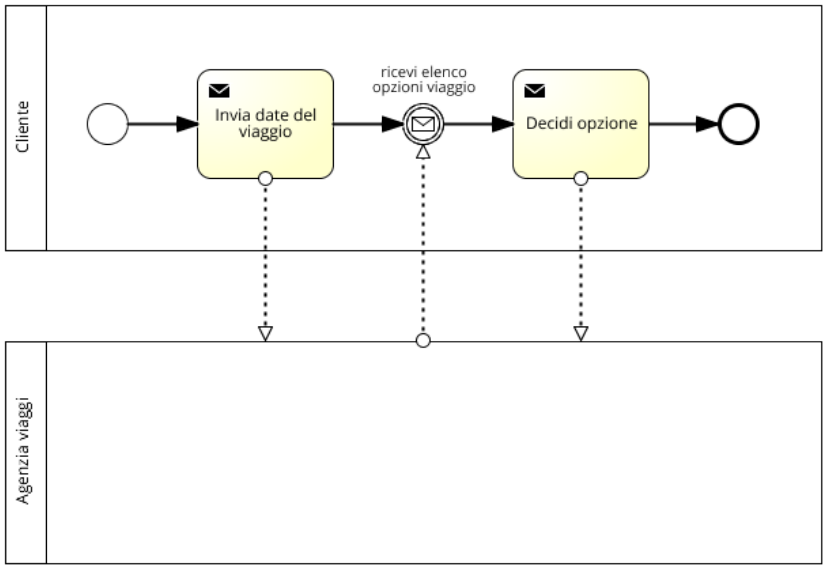


Figure 4.1: *Collaboration diagram*

As highlighted in the example, the exchange of messages is closely linked to the concept of a message event described in the previous section. A message sent from one pool and expected by another can be modeled as an intermediate event placed on the control flow. The exchange of messages also affects the nature of activities. When activities generate messages, such as *Send travel dates* and *Choose option*, they can be modeled as *Send Message Tasks*. Similarly, instead of using a catching message intermediate event such as *Receive list of travel options*, it is also possible to use a *Receive Message Task*.

4.2 Private and public processes

The previous example allows introducing the concept of *private process* and *public process*. A private process concerns activities, organized in a control flow, that are internal and controlled by a specific participant. This process is the one that the participant must execute and orchestrate. The public process, on the other hand, is composed of a private process and its interaction with other processes. In the example of Figure 4.1, the private process consists of the content within the client pool, while the public process is the entire collaboration diagram.

Example 4.2 - Considering the previous example, consider the process in Figure 4.2. In this case, the private process of the travel agency is also modeled. This makes it possible to know who consumes and produces the exchanged messages on the agency side as well. Therefore, there are two private processes (inside the pools) and two public processes (considering the message exchange and, in one case, the private process of the client, and in the second case, the private process of the agency).

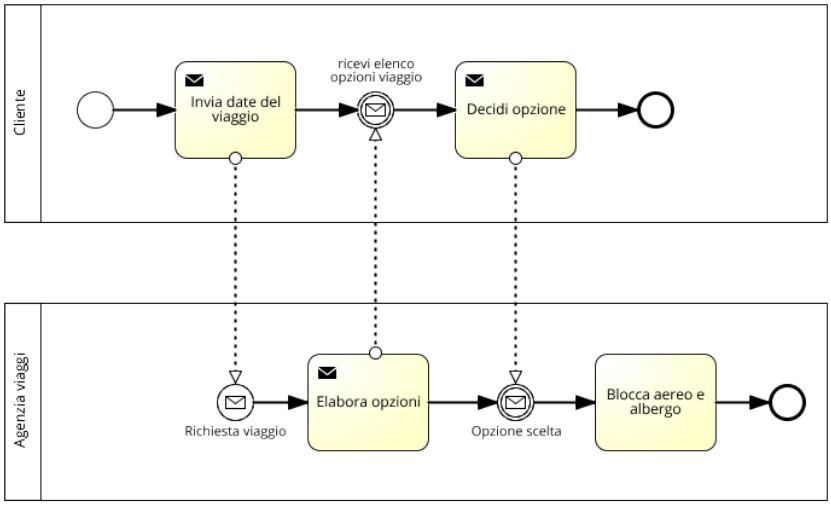


Figure 4.2: *Collaboration diagram with white-box pools*

To better emphasize the private process modeled in a diagram, the participant executing that process can also be represented outside of a pool.

Example 4.3 - For example, the diagram in Figure 4.3 still shows the vacation booking process, in which the client's activities are not enclosed within a pool. This indicates that the modeler has taken the client's point of view and considers the client's process as the private process. This means that, even if the travel agency pool is a white box, the process shown is not necessarily the actual internal process of the agency. It could be a simplification, modeling only the activities necessary to specify the message exchanges with the client.

It should be noted that in a collaboration diagram, only one private process can be represented outside of a pool.

Since a pool represents a conceptual participant in a process—who, although collaborating with other participants, has its own autonomy—it is not allowed for a process's control flow to cross pool boundaries. The only way to communicate between different pools is by using messages. This restriction makes sense in the semantics of the control flow associated with a process. The control flow defines the sequence to be managed and executed by an orchestrator. Therefore, it is assumed that each participant has an orchestrator capable of deciding how and when to execute the process activities based on the control flow. Allowing a control flow to connect two participants would mean permitting an orchestrator of one participant to "command" inside another participant,

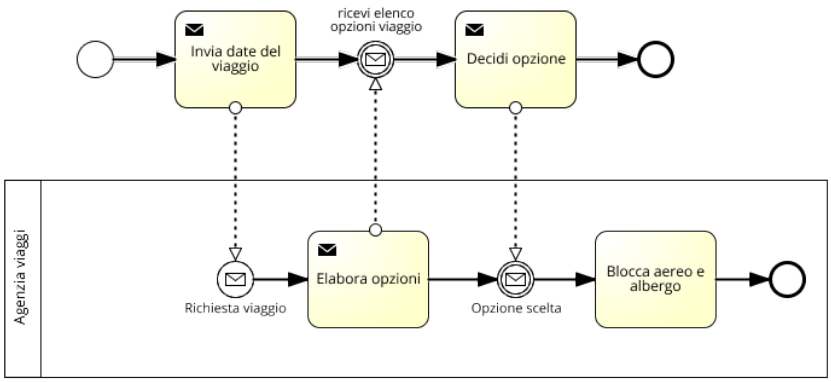


Figure 4.3: *Collaboration diagram emphasizing the client's private process*

which is not allowed to preserve autonomy. Conversely, when a participant requests collaboration from another, the tool available is the message. A participant sends a message to another participant, implicitly informing them of the state of execution of their process and optionally requesting a response.

4.3 Multi-instance pool

As with activities and subprocesses, in BPMN it is possible to define multi-instance pools. The marker used for this purpose is the same as that used for activities (three parallel vertical lines). This indicates the possibility of specifying interactions with different participants whose role in relation to the process being modeled is identical, as well as the message flows that support them.

Example 4.4 - Suppose that a customer's vacation booking does not involve interaction with a single travel agency but with several agencies. The customer proposes travel dates to multiple agencies and, after selecting the best offer, concludes the transaction with only one of them. The corresponding process is illustrated in Figure 4.4, where the customer's activities of sending dates and receiving options have been functionally grouped into a macro-activity which, to support communication with the different travel agencies, is also of the parallel multi-instance type.

4.4 Lane

Given a pool, BPMN allows further specification of the corresponding participant's characteristics through the *lane* construct. From a graphical perspective, a lane has the same shape as a pool. From a semantic perspective, a lane represents a role or a specific actor belonging to a participant. For this reason, a lane is an element placed inside a pool.

Example 4.5 - Suppose we want to better specify the internal process of the travel agency. The diagram in Figure 4.5 shows the subdivision of the corresponding pool into two lanes. The two lanes refer to two sub-organizations of the travel agency. The

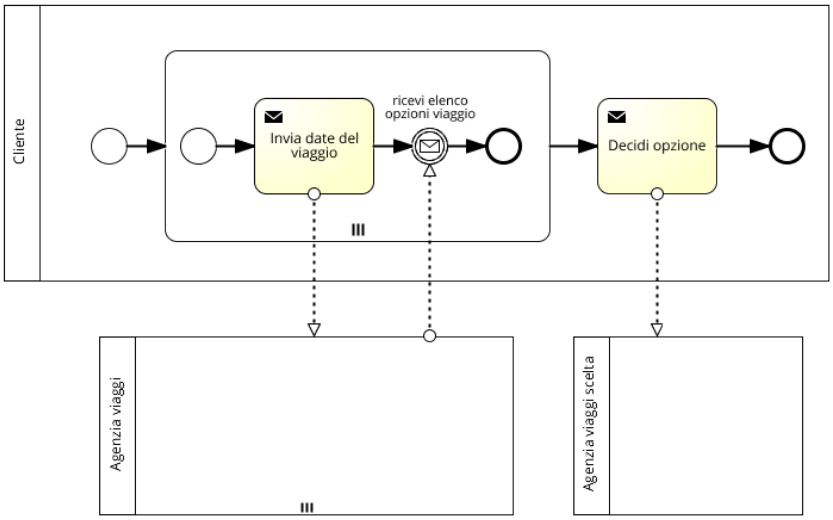


Figure 4.4: *Multi-instance pool*

> first, i.e., the sales office, manages the customer relationship and hotel bookings; the second, i.e., the flight division, handles only air bookings.

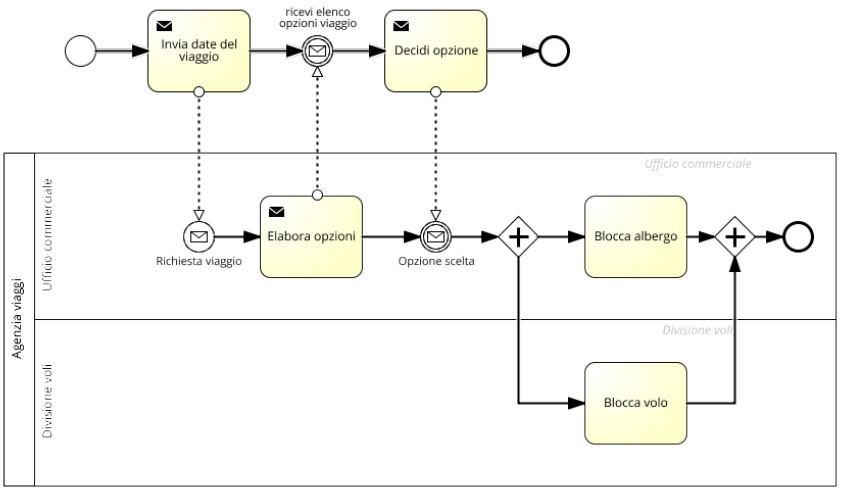


Figure 4.5: *Use of lanes*

5 Summary Examples

5.1 PalmaViaggi s.r.l.

Model the following travel package booking process using BPMN.

The process begins when the user, an employee of the company TalDeiTali s.r.l., accesses the online booking website of PalmaViaggi s.r.l. and specifies the dates of their trip. Upon receiving the dates, the company searches its database for all possible travel solutions and presents them to the customer, who selects one. Once the trip is selected, the company proposes a list of hotels that can be booked in the destination city. The customer can decide whether or not to book one of the proposed hotels or reject the request. Once this phase is completed, PalmaViaggi sends a summary of the bookings and payment is carried out. Payment is processed by the administrative department of TalDeiTali. Upon receiving the payment, PalmaViaggi proceeds to finalize the flight and hotel bookings. The bookings are carried out simultaneously (the hotel booking is made only if it was selected). Once the bookings are finalized, PalmaViaggi sends confirmation of the itinerary.

The process must comply with the following time constraint: payment must be received within 5 minutes of sending the summary information. Otherwise, PalmaViaggi cancels the process and notifies the cancellation of the booking procedure (Figure 5.1).

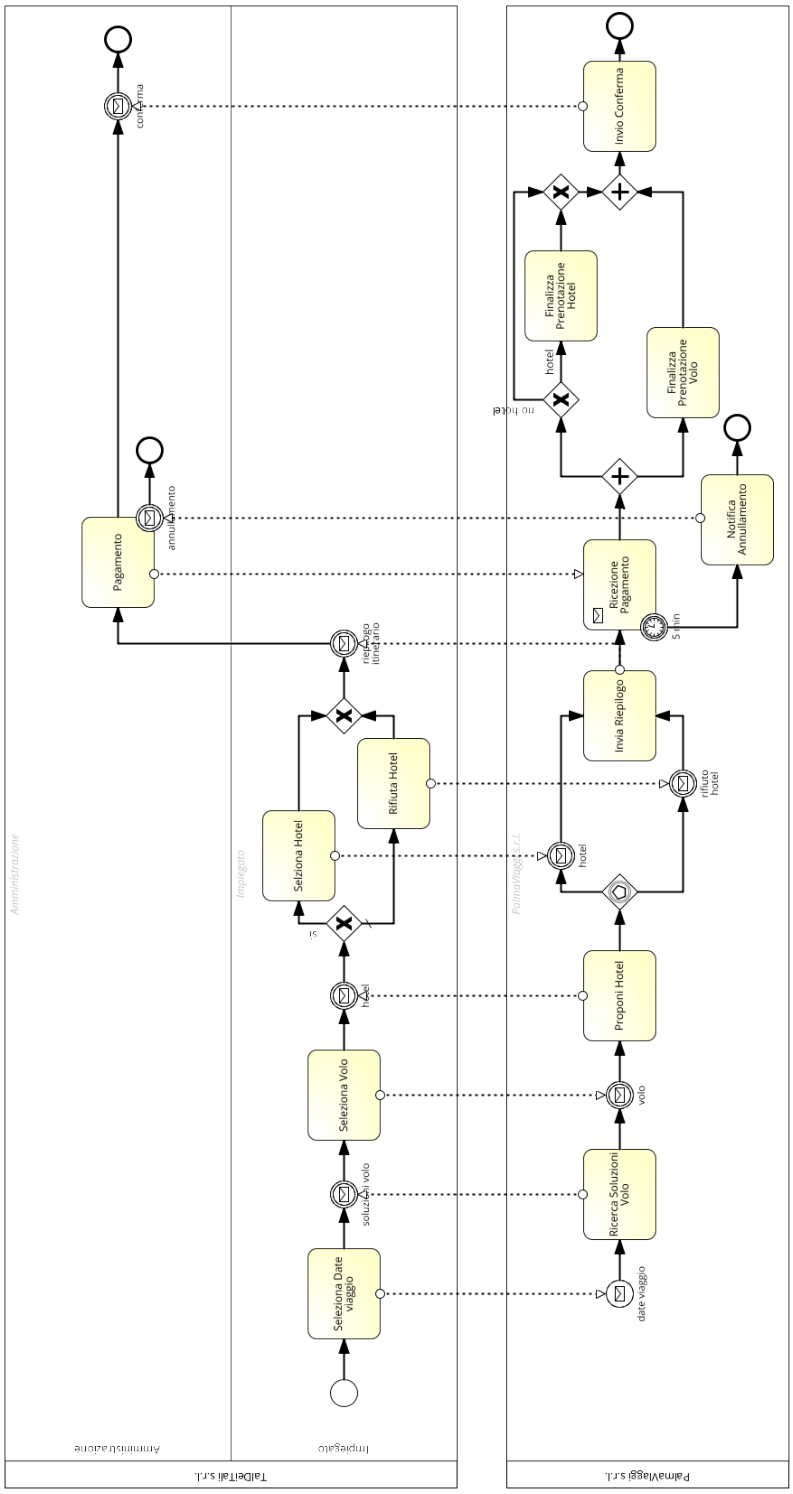


Figure 5.1: *PalmaViaggi s.r.l. - payment timer*

5.2 PalmaViaggi s.r.l. - variation

Model the following travel package booking process using BPMN.

The process begins when the user, an employee of the company TalDeiTali s.r.l., accesses the online booking website of PalmaViaggi s.r.l. and specifies the dates of their trip. Upon receiving the dates, the company searches its database for all possible travel solutions and presents them to the customer, who selects one. Once the trip is selected, the company proposes a list of hotels that can be booked in the destination city. The customer can decide whether or not to book one of the proposed hotels. Once this phase is completed, PalmaViaggi sends a summary of the bookings and payment is processed. Payment is carried out by the administrative department of TalDeiTali. Upon receiving the payment, PalmaViaggi proceeds to finalize the flight and hotel bookings. The bookings are carried out simultaneously (the hotel booking is made only if it was selected). Once the bookings are finalized, PalmaViaggi sends confirmation of the itinerary.

The process must comply with the following time constraint: from the moment PalmaViaggi begins the travel package booking procedure (receiving the travel dates), the booking process must be completed within 15 minutes. Otherwise, PalmaViaggi cancels the process and notifies the cancellation of the booking procedure (Figure 5.2).

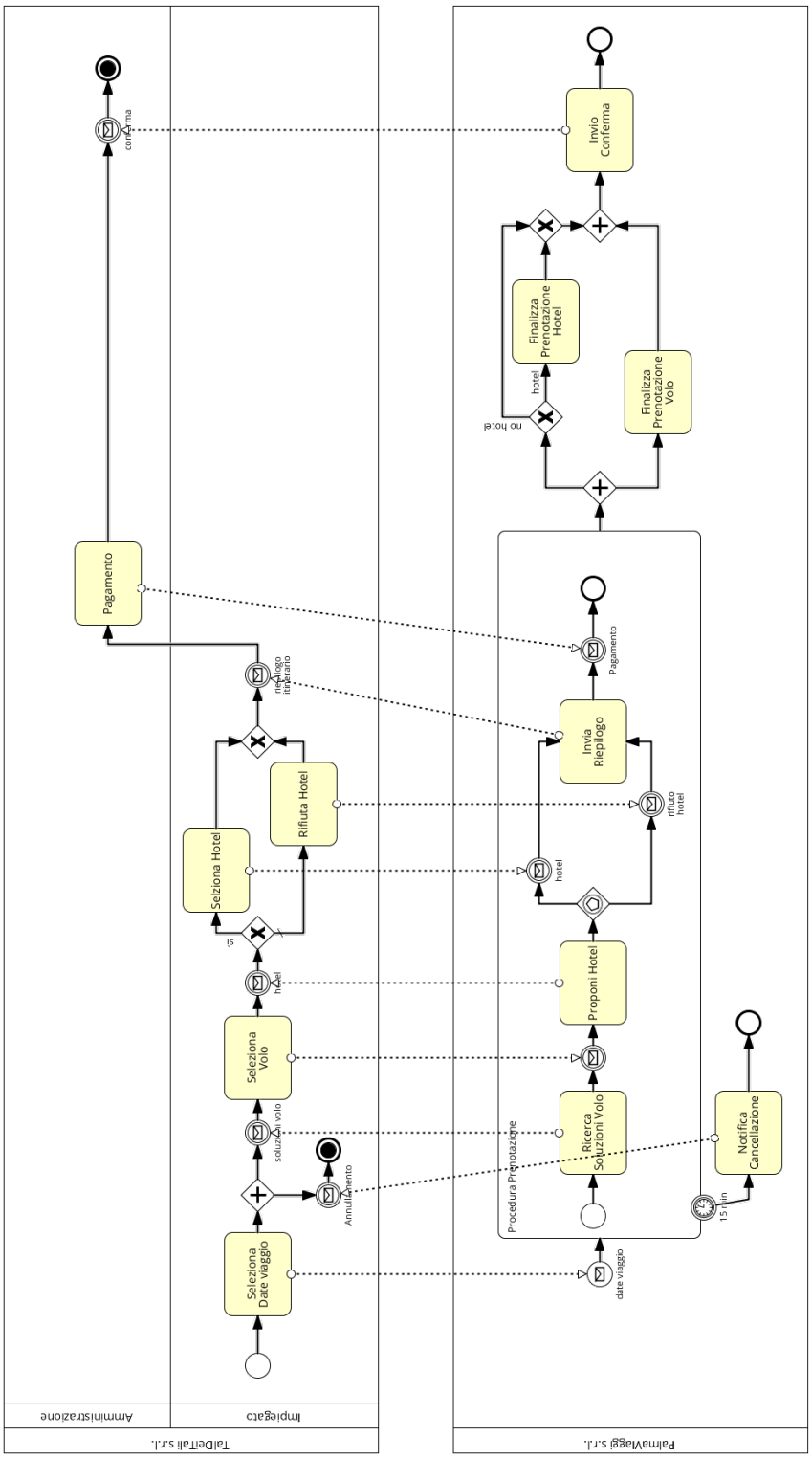


Figure 5.2: *PalmaViaggi s.r.l. - booking timer*

5.3 Request for Quotation

An RFQ (request for quotation) system is a typical process aimed at inviting suppliers to submit offers for products. The process thus involves a customer, interested in the product, and a seller who provides it.

The process is initiated by the customer, who contacts the seller to obtain an offer for a product. Upon receiving the request, the seller first checks the reliability of the customer. If the customer is not reliable, the request is rejected and the process ends. If the customer is reliable, the seller estimates the date when the product will be available. Simultaneously, the seller also determines the delivery times. This latter operation is necessary only for shipments to a foreign country. For shipments within the same country, estimating the delivery date is not required. The processed data are then sent to the customer as a proposal, and the seller waits for the customer's final order. Upon receiving the order, the seller sends a confirmation message and the process ends. The process also ends if the customer rejects the offer or if no communication is received within 48 hours.

Model both the seller's process and the customer's process (Figure 5.3).

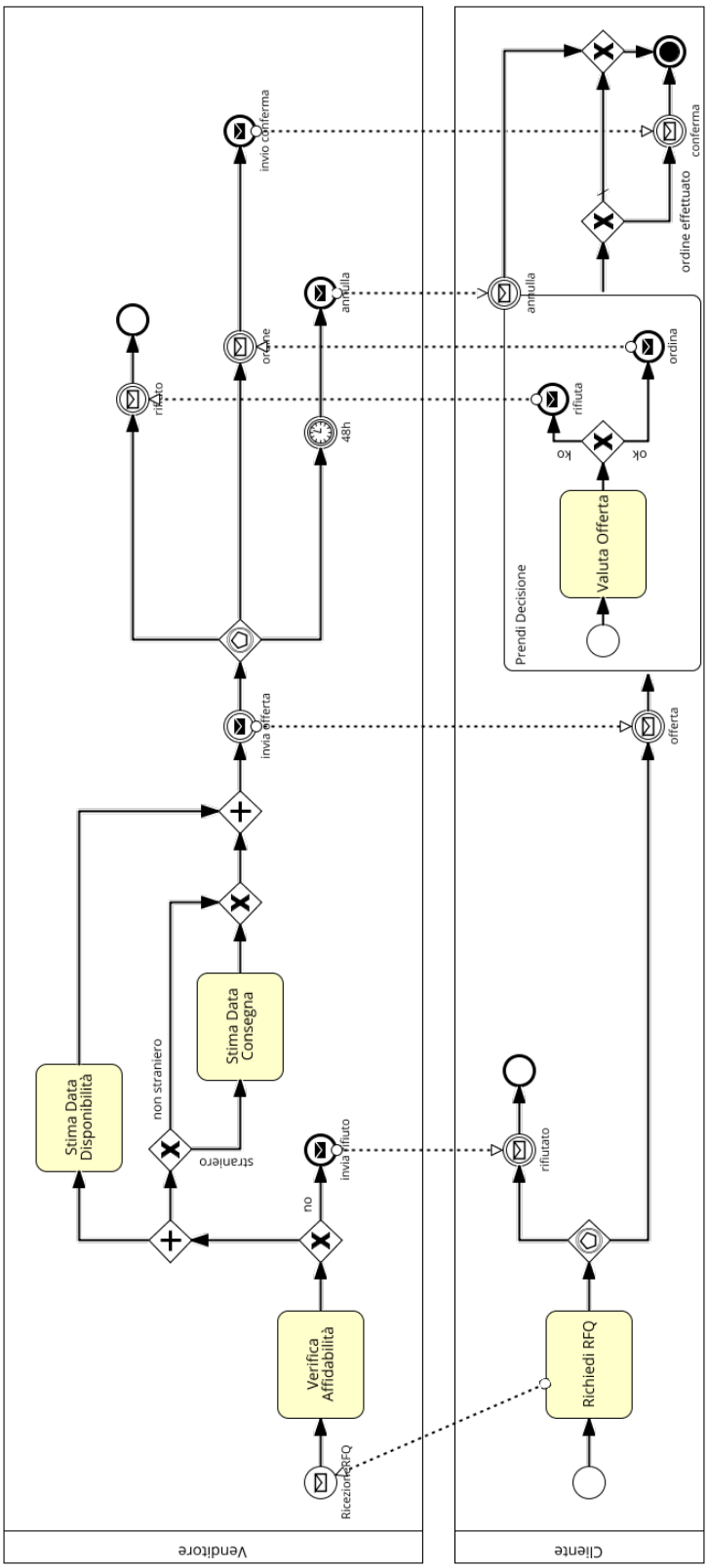


Figure 5.3: *Request for Quotation*

5.4 Software ABC

Describe, using the BPMN formalism, the process outlined below.

The operational context of the process concerns the management of problems encountered by a customer while using the ABC software product provided by XYZ. The customer is in direct contact with a staff member of XYZ, who acts as the sole interface between the customer and the supplier. This arrangement follows the company's customer management strategy and is never bypassed. The customer reports the problem to the contact person and, if requested, provides further explanations. The staff member verifies whether they have the necessary skills to respond directly to the customer and, if so, does so immediately. If the problem cannot be solved directly, the first-level help desk is involved, receiving a detailed description of the issue. If the first-level help desk can resolve the problem, it communicates the solution to the staff member responsible for customer interaction; otherwise, the problem is forwarded to the development team. The development team examines the issue, and if the solution does not require software integration, it communicates the response to the help desk, and the solution is conveyed to the customer through the same operational path (help desk, staff member). If software development is required, the team proceeds with the integration. Once integration is complete, the solution is conveyed to the customer via the same operational path. If, starting from the help desk taking charge of the problem, no solution is communicated within 10 days, the help desk still sends a message to the staff member to forward to the customer, indicating the delay in providing the solution. Once the solution is found, it is communicated to the customer using the prescribed method. At the end of the process, the staff member communicates a solution to the customer (even if not fully resolving the issue). Solution in Figure 5.4.

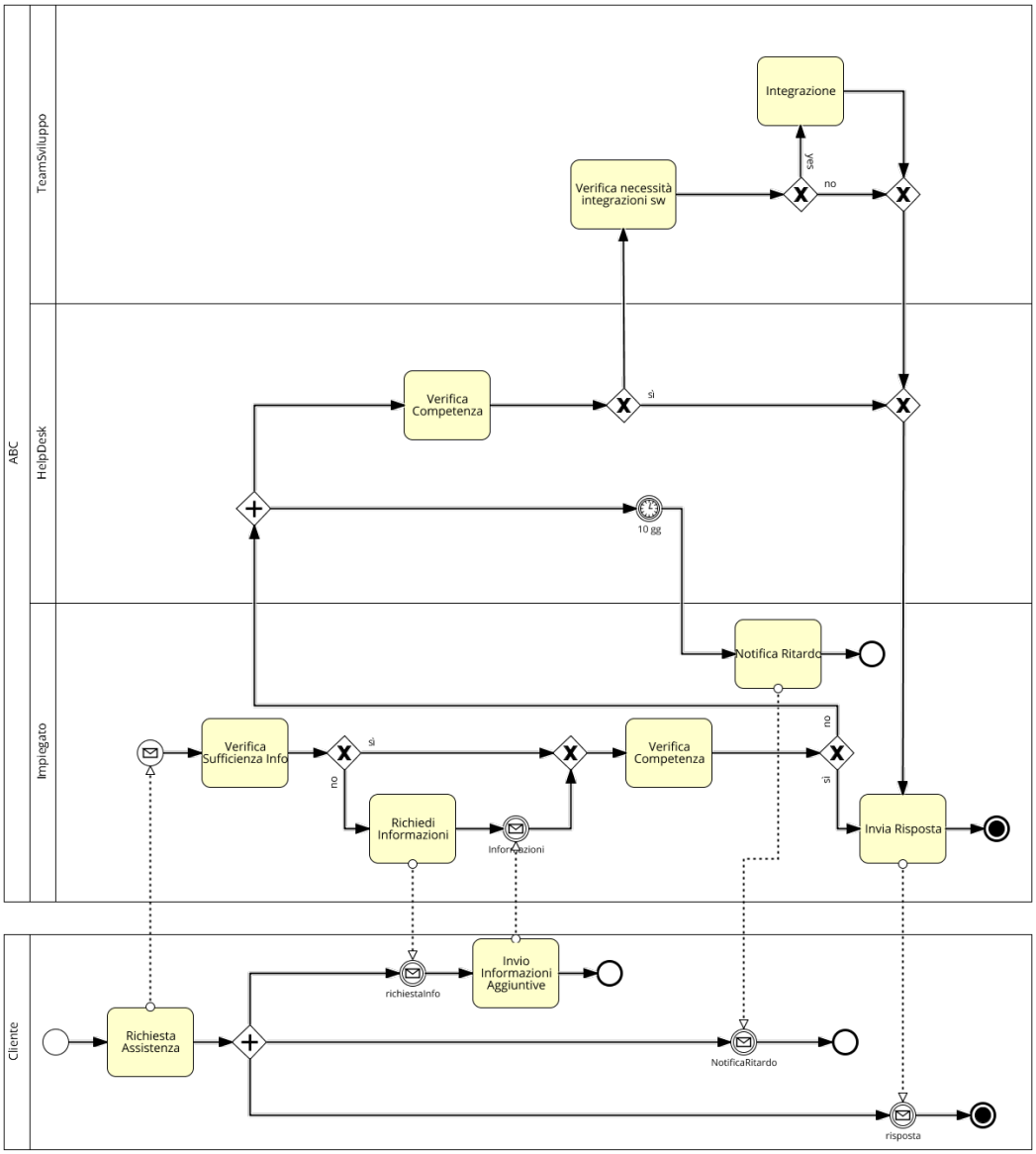


Figure 5.4: *Software ABC*

5.5 Editorial Committee

Represent the process described below (Figure 5.5). The LMN magazine uses a highly collaborative method for selecting and publishing articles. The participants in this process are:

- The Director of the Editorial Committee
- The Editorial Committee: composed of 10 members
- The Contributors: individuals (approximately 15) who actually write the articles for the magazine.

The entire process begins when the Director decides it is time to collect contributions for the next publication. This communication is sent to the members of the Editorial Committee, who can compile a List of Topics to send to the Director. Submitting the list is not mandatory, but must be done within 5 days of the request. Lists submitted late will be ignored by the Director. After 5 days, the Director counts the received Lists of Topics and checks if it is possible to prepare a single list using the following criteria:

- There are at least 7 Lists of Topics: the Director compiles a list of 20 topics, ordered by importance; if 20 topics are not available, they personally add the missing topics in the order deemed appropriate.
- There are 2 to 6 Lists of Topics: the Director compiles a list of 15 topics, ordered by importance, possibly adding missing topics as above.
- There is no List of Topics or at most one: the Director informs the Editorial Committee that it is not possible to create a List of Topics.

If the lists are sufficient, the Director communicates the final list of topics to the Editorial Committee and arbitrarily assigns one or two topics to each Contributor, who will write their contribution and send it to the Director. The Director waits for the contributions for a maximum of 5 days. Late contributions will be ignored.

In the solution, also indicate data management.

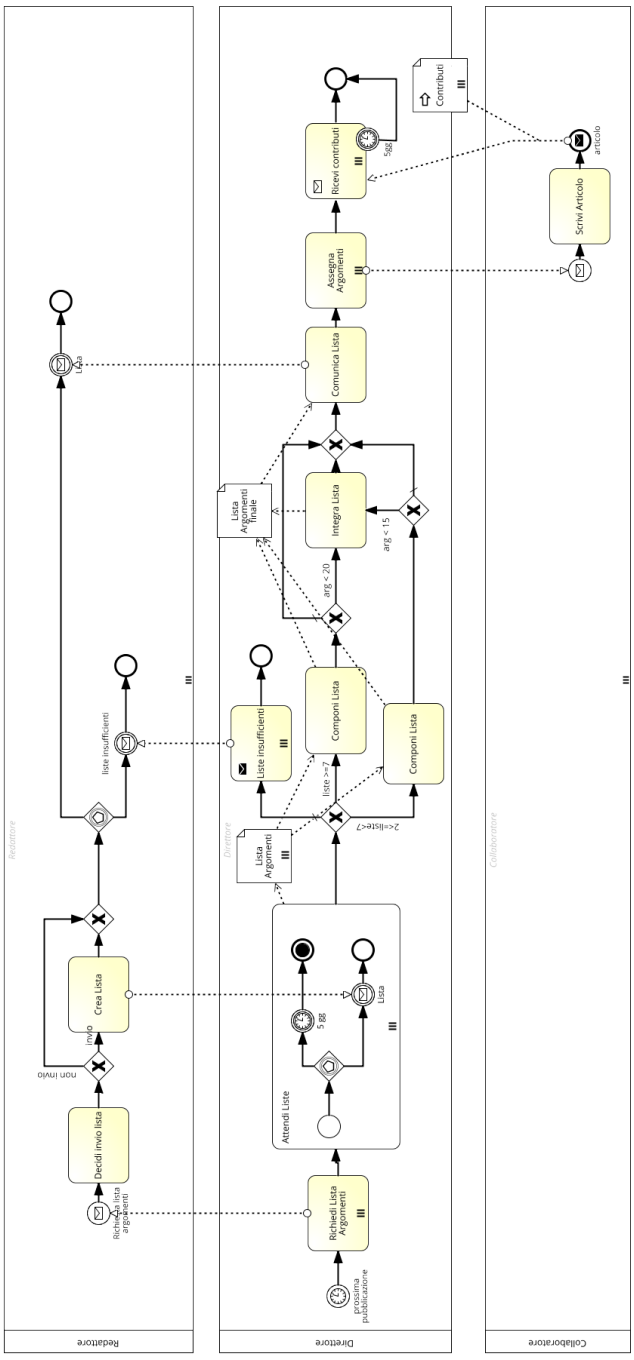


Figure 5.5: *Editorial Committee - List of Topics*

5.6 Editorial Committee - variation

Represent the process described below (Figure 5.6). The LMN magazine uses a highly collaborative method for selecting and publishing articles. The participants in this process are:

- The Director of the Editorial Committee
- The Editorial Committee: composed of 10 members
- The Contributors: individuals (approximately 15) who actually write the articles for the magazine.

The entire process begins when the Director decides it is time to collect contributions for the next publication. For this purpose, the Director notifies a list of topics to the Editorial Committee and arbitrarily assigns one or two topics to each Contributor, who will write their contribution and send it to the Director. The Director waits for contributions for a maximum of 5 days. Any late contributions will be ignored. After five days, the Director proceeds as follows:

- If at least 10 articles have been received, the magazine will be published; otherwise, the Director informs the Editorial Committee that there are not enough articles.
- If more than 10 articles are received, the Director compiles an ordered list and submits it to the vote of the Editorial Committee.

Each member of the Editorial Committee must indicate within two days which articles, in their opinion, should be published. The Director collects the votes from the Editorial Committee and identifies the 10 articles that will be published, i.e., those that received the most votes. The Director then communicates the selected articles to the committee.

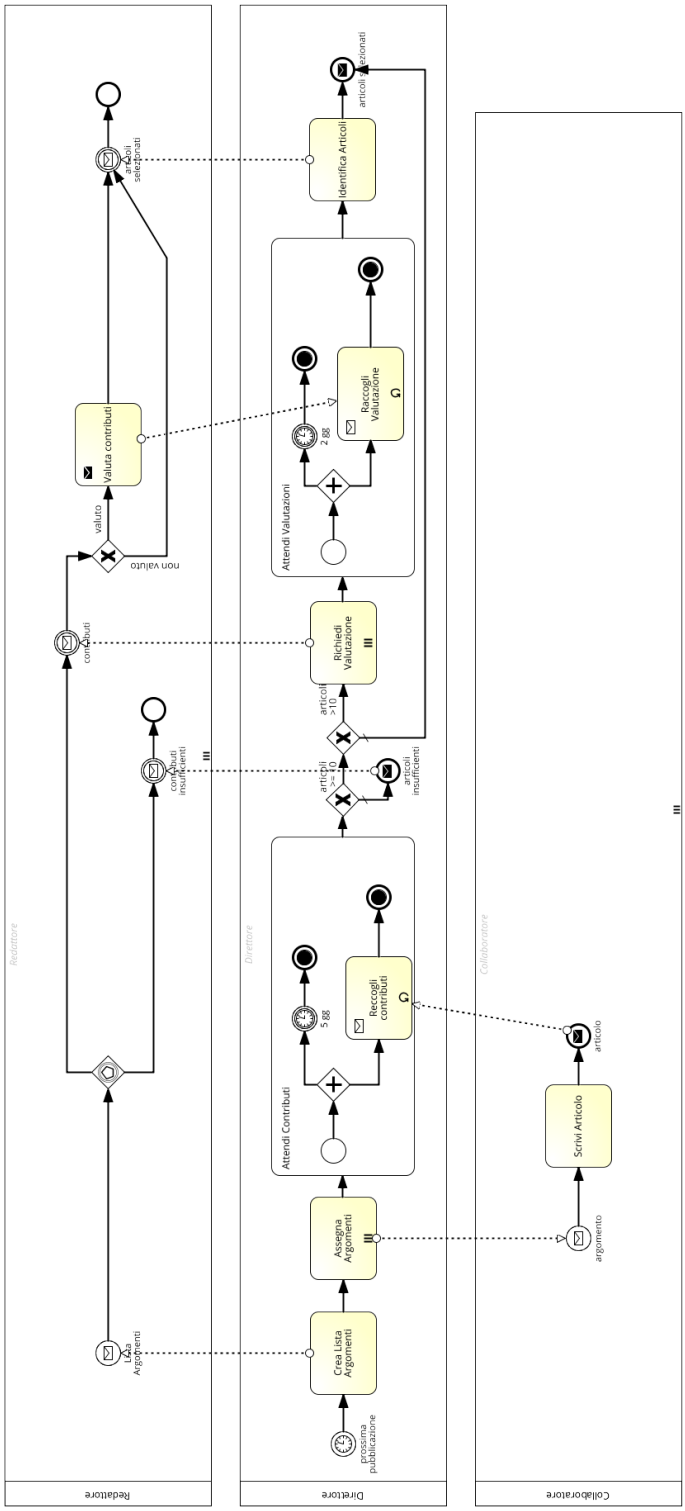


Figure 5.6: *Editorial Committee - contributions*

5.7 ArteInStrada

The municipality of Milan recently inaugurated the “ArteInStrada” project, which allows street artists to book locations throughout the historic center for free performances.

An artist who wishes to perform accesses the system and enters information about their performance (title, category, noise level), then selects a date, and finally specifies the required space dimensions. Once completed, the request is sent to the booking department of ArteInStrada. Upon receiving the request, the system checks among the available locations on the specified date for those that meet the space requirements and sends the list to the artist. If no location meets the requirements, the system notifies the artist and the process ends.

Upon receiving the list of locations, the artist selects one and waits for final approval from ArteInStrada. The full request is reviewed by the artistic committee of ArteInStrada, which verifies whether the proposal complies with the project regulations. If approved, the approval is communicated to the artist; otherwise, the request is rejected and the process ends. To avoid reminders from artists waiting for confirmation, if the approval process is not completed within one day, the system sends a notification to the user indicating that the request is under review.

When the artist receives the approval notification, they prepare a final version of the proposal following any instructions in the notification and send the confirmation to ArteInStrada. The booking department, upon receiving the confirmation, completes the process. The booking department waits for the artist’s confirmation for 2 days, after which it cancels the booking and informs the artist, ending the process (Figure 5.7).

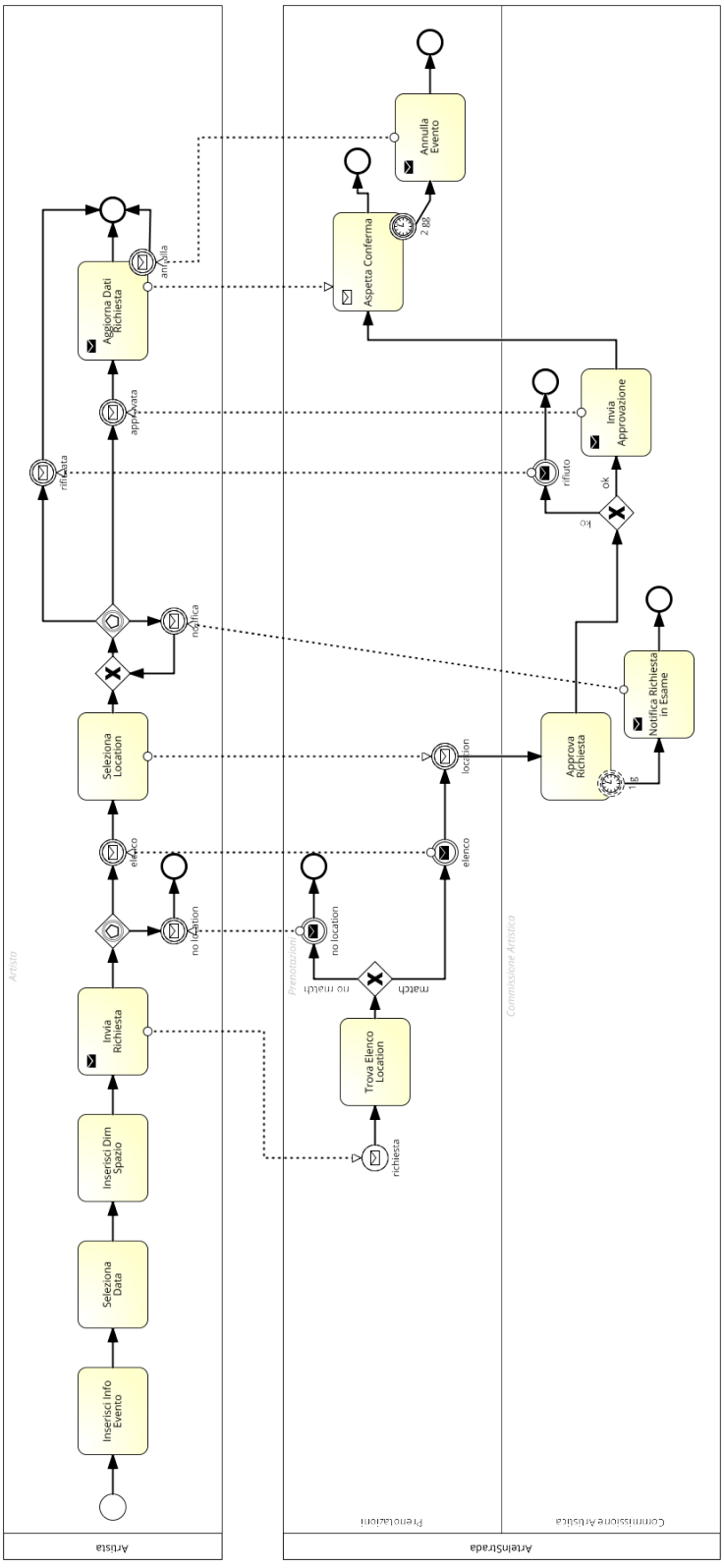


Figure 5.7: *ArteInStrada*

5.8 XP Association Administration

Represent the process described below using BPMN.

The XP association makes decisions regarding the management of the association's assets according to the following logic. Decisions are made by:

- A manager;
- A board of directors, called the BoD;
- A supervisor.

All communications between these entities occur via email (consider the subjects as separate entities). The manager is responsible for proposing investments regarding the assets. Depending on the type of investment, the decision can follow three modes:

- Spontaneous Decision (DS): the manager communicates their decision to the BoD and the supervisor. The BoD acknowledges the decision and waits for confirmation. The manager waits for 3 days for any input from the supervisor, after which the proposal is approved. If the supervisor intervenes within 8 days, the proposal is canceled and the process ends. Otherwise, approval is confirmed to the BoD.
- Decision with BoD Consultation (DC): the manager communicates a proposal to the BoD, which holds an internal meeting and collects opinions. At the end of the meeting, a report is sent to the manager. The manager evaluates the report and decides whether to cancel or approve the proposal. In case of approval, it is communicated to the supervisor, who has 3 days to express a negative opinion. At the end of the 3 days, the manager finally approves the proposal and communicates the approval to the BoD. If the supervisor intervenes, the proposal is canceled.
- Majority Vote Decision by the BoD (DV): in this case, the BoD's opinions are binding, and the manager requests a vote on the proposal. At the end of the voting procedure, the vote count is sent to the manager. The manager evaluates the outcome and, if the favorable votes are sufficient, definitively approves the proposal and informs the BoD. An absolute majority of BoD members is required. The manager waits for the voting results for a maximum of 8 days, after which the proposal is canceled. In this case, the supervisor is not involved.

The solution is shown in Figure 5.8.

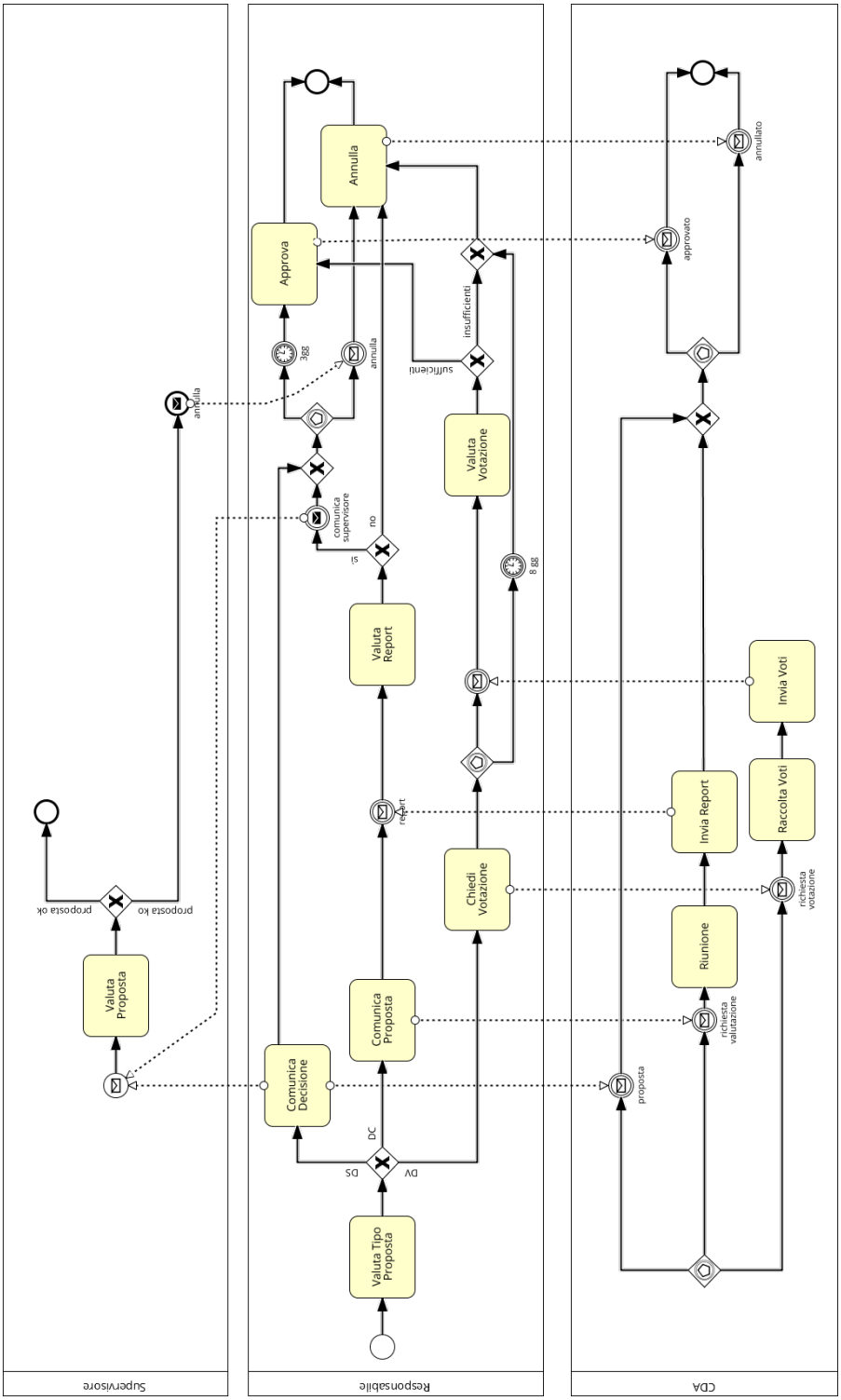


Figure 5.8: *XP Association Administration*

5.9 Virgo

Model the personnel selection process of the company Virgo. When a new position is requested within the company, the technical office prepares all necessary documentation, including the requirements the candidate must meet. Once the documentation is completed, it is published online to allow interested applicants to submit their applications. Applications are automatically collected by the system (i.e., without manual intervention), which records the received data in the database. The application collection period lasts exactly 15 days, after which no further applications are accepted.

After the application period ends, the evaluation phase begins. The technical office staff evaluates applications one at a time in the order they were received. For each application, the candidate's data is sent to three members of an external committee, and the office waits for their evaluations (based on a discrete scoring scale). Once all evaluations are received, it is checked whether the candidate has received at least two positive votes out of three. If not, the next candidate is evaluated (if any remain); otherwise, the candidate's data is recorded in the system. Once all candidates are evaluated, the technical department selects at its discretion one of the candidates who passed the voting phase (if any). Note that there are no time constraints for the committee's evaluations. However, the entire evaluation phase must, according to company policy, be completed within 5 days from its start. If this does not occur, the evaluation phase ends, and the process restarts from the preparation of new documentation. If the 5-day period is met without finding a suitable candidate, the announcement is reposted online without preparing new documentation, awaiting new candidates.

All actors involved in the process must be modeled, except the candidate (Figure 5.9).

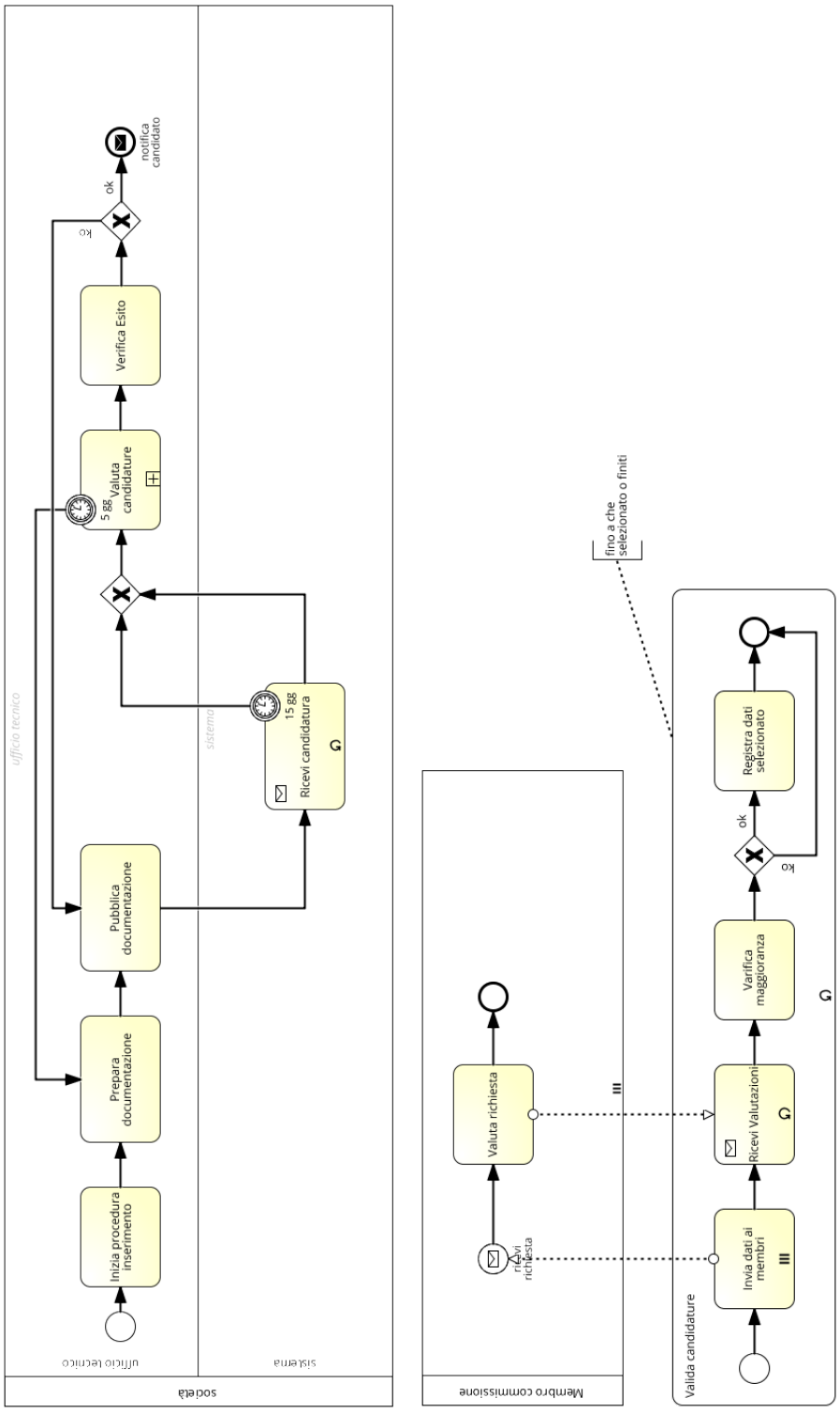


Figure 5.9: *Virgo - personnel selection*

5.10 Slow Food

A slow food chain specializing in hamburgers wants to use an information system to manage its restaurant.

Model the process of submitting a remote order described below using BPMN notation. The customer, using the chain's website, first selects the product category they want to order (hamburgers, sides, desserts), then selects a specific product and may choose to customize it with optional ingredients. Once the product is selected, the information is sent to the system, which adds it to the shopping cart. The customer can choose to add more products (repeating the same process described above) or confirm the order. Note that once the first product is added, the process must be completed within 10 minutes; otherwise, the system cancels the order and notifies the customer. Once all desired products are selected and the order is confirmed, the system generates the invoice and shows it to the customer, who must enter their credit card details to finalize the payment. The system then verifies the data and notifies the customer of the outcome. If positive, the process ends; otherwise, the customer can re-enter the data or cancel the order (Figure 5.10).

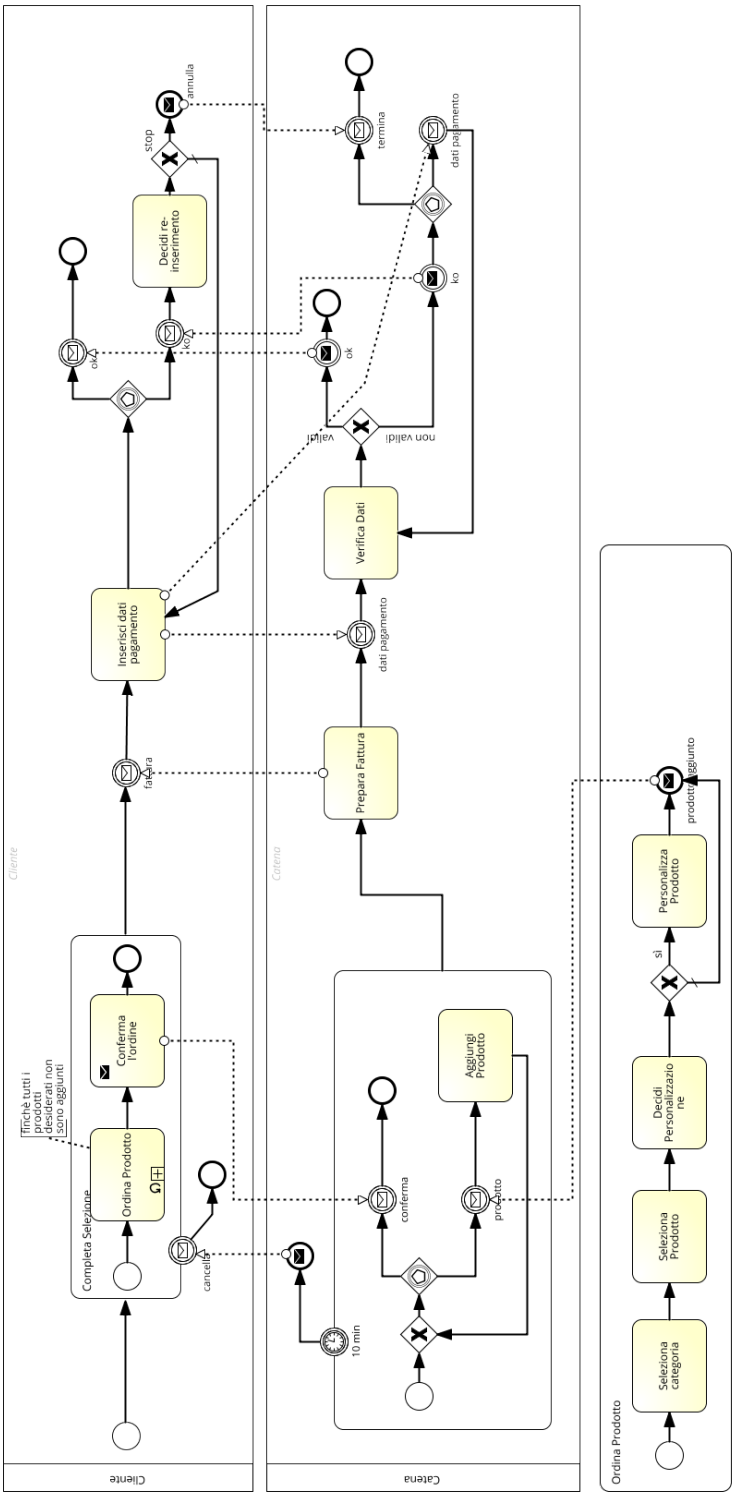


Figure 5.10: *Slow Food*

5.11 Loan Offer

The Marketing Department of a credit mediation company decides to launch a marketing campaign by sending some clients an offer for a loan at a preferential rate. The department prepares a list of clients to contact and sends the loan proposal. If no response is received within two weeks, or if the client declines the offer, the department records the response and removes the client from the list. If the client accepts the offer and submits the required documents, the case management is transferred to the Administration.

The Administration checks the submitted documents and, if complete, schedules an appointment with the client to sign the paperwork. If the submitted documents are incomplete, the Administration requests the missing documents. Upon receipt, the Administration repeats the completeness check. If no response is received within 10 days, the Administration sends a reminder to the client. In any case, after one month from the initial request, the department records the failure and removes the client from the list.

Model the above process using BPMN notation (Figure 5.11).

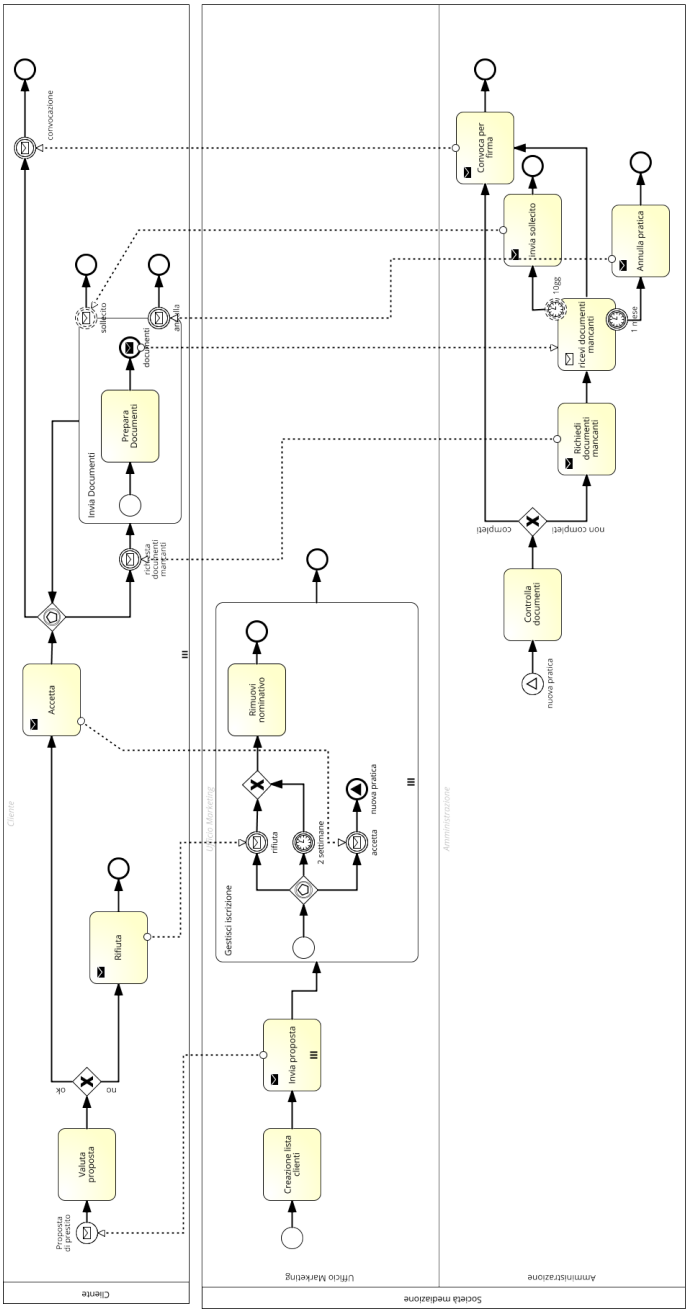


Figure 5.11: *Loan Offer*

5.12 FabbricaLib

The company FabbricaLib offers its clients a system for creating customized bookshelves. Model, using BPMN, the process describing the interaction between the customer and the company during the configuration and purchase of a customized bookshelf.

The customer who wants to purchase a bookshelf first chooses the dimensions and the material. These two operations can be performed simultaneously. Once configured, the customer sends a quote request to FabbricaLib. The company, upon receiving the request, calculates the quote based on the configuration and sends it to the customer. The customer can then choose to accept the quote and proceed with the purchase, or cancel the process. In case of cancellation, the process ends after sending a notification to the company. Otherwise, the customer confirms the order and the company, upon receiving the confirmation, sends the payment details to the customer. The company waits for the payment, which must be completed within 10 minutes; otherwise, the order is canceled. The customer, after receiving payment details, can also decide not to proceed and cancel the order. If payment is made, the company forwards the order and sends the invoice to the customer. Any cancellation by the customer must be handled by the company to properly terminate its process.

Model both the company and the customer (Figure 5.12).

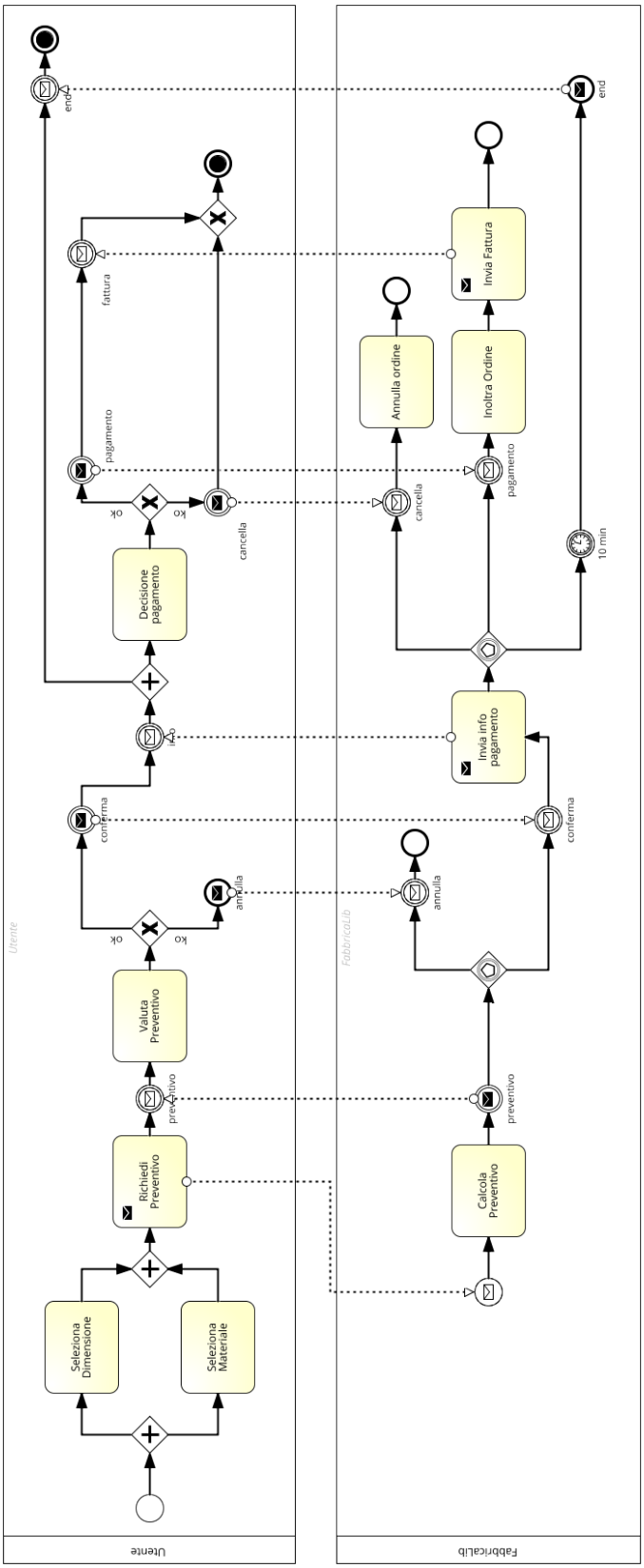


Figure 5.12: *FabbricaLib version 1*

5.13 FabbricaLib - variation

The company FabbricaLib offers its clients a system for creating customized bookshelves. Model, using BPMN, the process describing the interaction between the customer and the company during the configuration and purchase of a customized bookshelf.

The customer who wants to purchase a bookshelf first chooses the dimensions and the material. These two operations can be performed simultaneously. Once configured, the customer sends a quote request to FabbricaLib. The company, upon receiving the request, calculates the quote based on the configuration and sends it to the customer. This operation can last at most 2 days, after which an error message is sent to the customer and the process ends. If the quote is successfully sent, the customer can choose to accept the quote and proceed with the purchase, or cancel the process. In case of cancellation, the process ends after sending a notification to the company. Otherwise, the customer confirms the order and the company, upon receiving the confirmation, sends the payment details to the customer and waits for the credit card information. The customer, after receiving the payment details, can also decide not to proceed and cancel the order. If the data is provided, the company performs a verification. If the data is correct, the company forwards the order and sends the invoice. If the data is incorrect, an error message is sent to the customer, who can either re-enter the payment data or cancel the operation. Any cancellation by the customer must be handled by the company, which must terminate its process.

Model both the company and the customer (Figure 5.13).

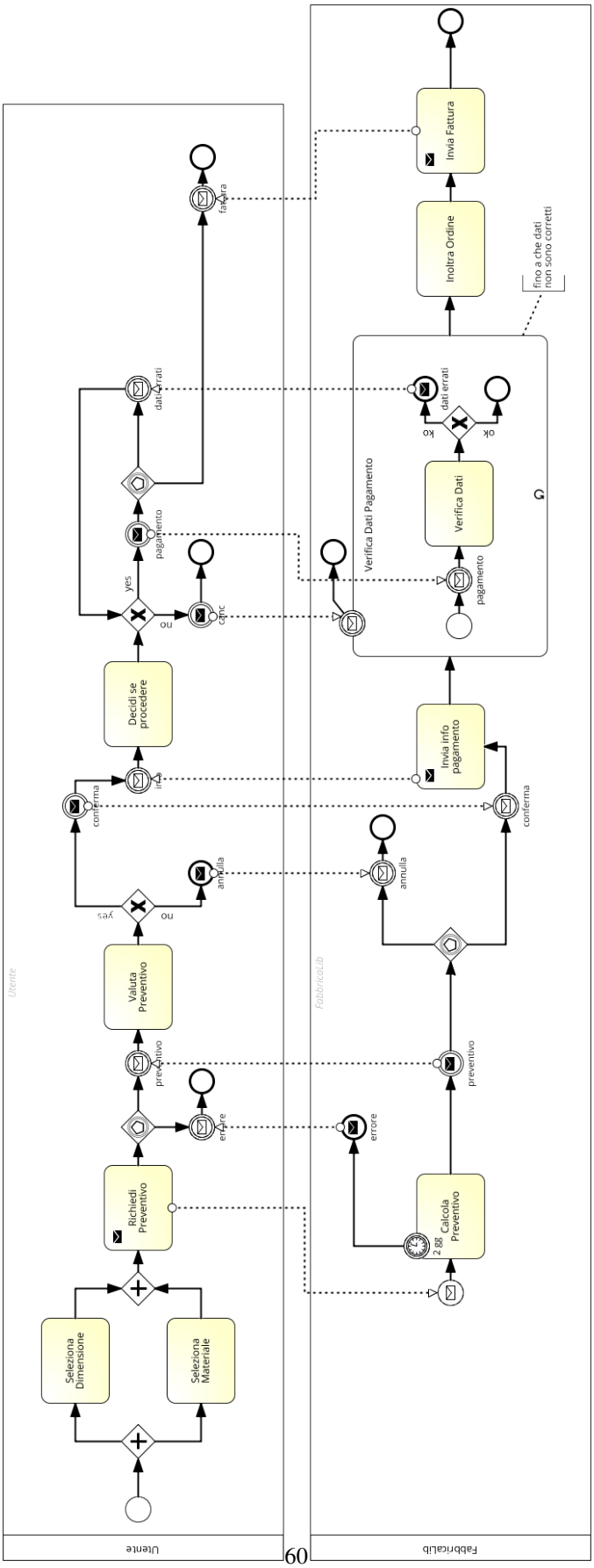


Figure 5.13: *FabbricaLib version 2*

5.14 ACME

The ACME agency organizes tourist trips offered to clients from a catalog. Interested clients send the agency a request for information to join a trip, specifying the destination, period, and number of participants. The agency checks if the trip is available. If not, a notification is sent to the client and the process ends. If available, a quote is sent. The client may accept the offer. If 7 days pass from sending the quote, the offer expires, and both the agency's and the client's processes end. If the client accepts, the agency simultaneously books the trip and hotel; when both bookings are completed, a confirmation is sent to the client, and the booking process ends (Figure 5.14).

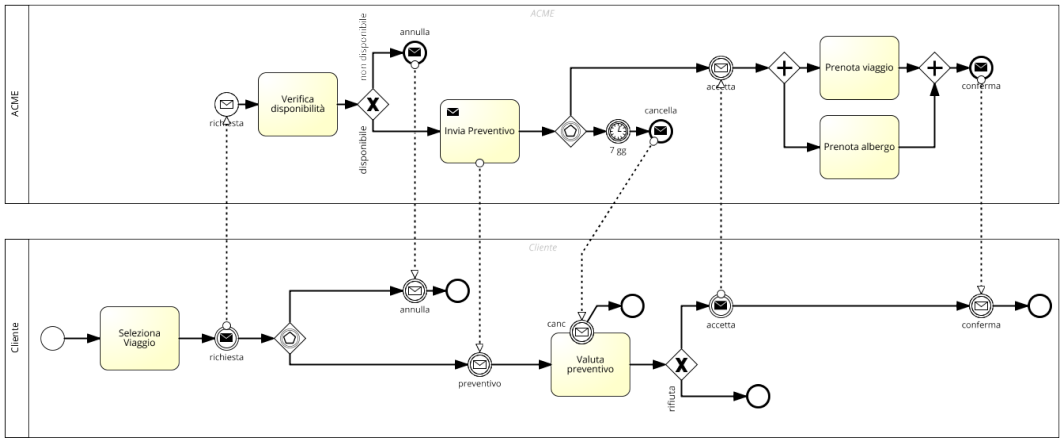


Figure 5.14: *ACME trips*

5.15 Refund Management

Represent the following process in BPMN, which involves two entities, named Receiving Office and Approval Entity. The two entities are distinct and follow separate processes that interact. The first entity collects refund requests from members via a service desk. It is the only one interacting directly with members. The Receiving Office verifies submitted requests; if information is missing, it sends a clarification request to the member. If no response is received after a clarification request, the case is archived after 7 days, and the process ends. Once the request is complete, the Receiving Office enters the subscriber information it holds and sends the case to the entity, awaiting a response.

The Approval Entity receives the case in the Accounting office and, in parallel, Accounting calculates the amount due while the Verification department evaluates eligibility. If eligible, individual expenses are verified and the refundable amount is calculated. After completing this activity, the Verification department sends the decision to Accounting, which responds to the Receiving Office (rejection if not eligible or with the approved amount if eligible). The Receiving Office then communicates the result to the member.

Model the processes of the Receiving Office and the Approval Entity (Figure 5.15).

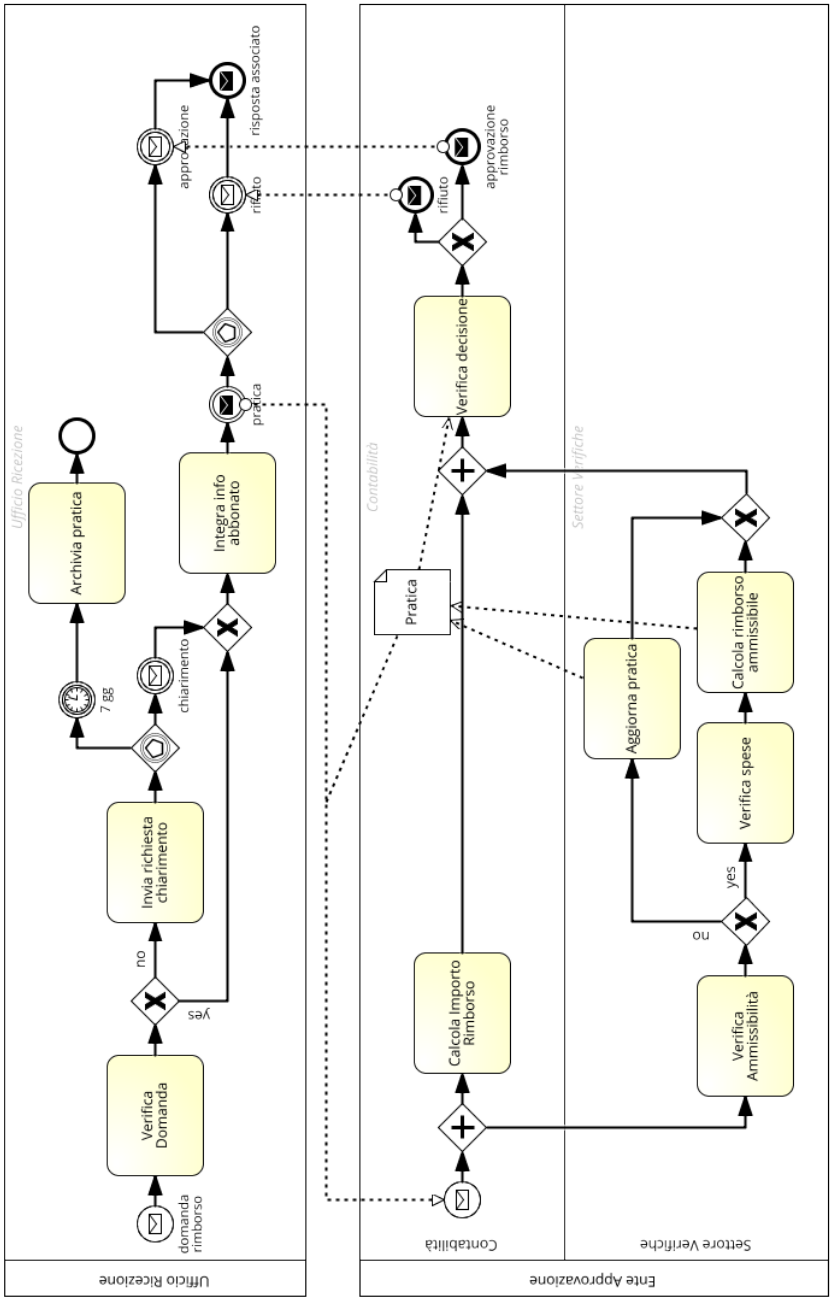


Figure 5.15: *Refund Management*

5.16 Driving School

Model the following process using BPMN notation, describing the interaction of a customer with a driving school portal for a theoretical test simulation.

The customer submits a request to the driving school, specifying whether they want a general test or a test on a specific topic. Upon receiving the request, the driving school randomly selects 10 questions from the database. In the case of a topic-specific test, the topic of interest must first be specified. The driving school then sends the set of questions to the customer. Upon receiving the questions, the customer answers and submits the responses to the driving school, which calculates the score and sends the test result to the customer. The customer reviews the results and, if they believe an answer is incorrect, can submit a report. Otherwise, they confirm the test outcome, and the process ends. Error reports must be sent within 10 minutes of receiving the result from the driving school; otherwise, the test is considered approved, and the driving school and customer processes conclude.

In the case of a report, the driving school analyzes the request and sends a response to the user, ending the process (Figure 5.16).

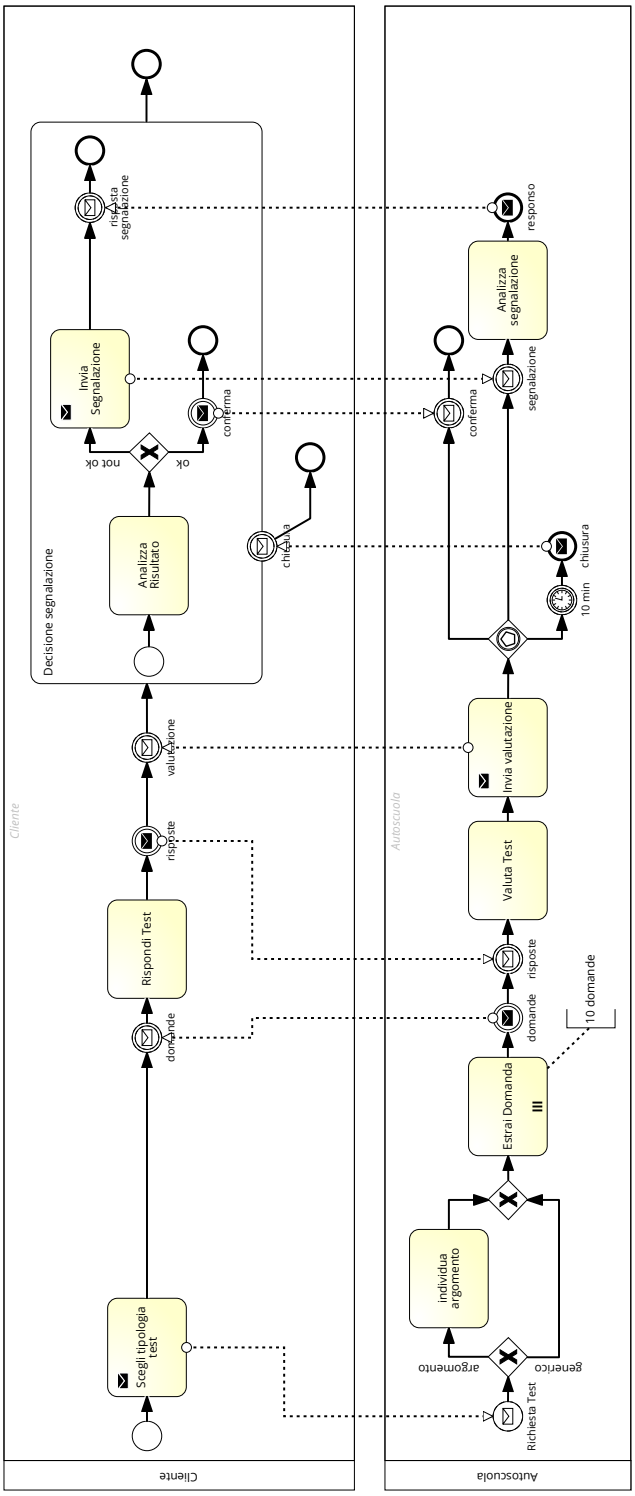


Figure 5.16: *Driving School*

5.17 Penna&Calamaio

Model in BPMN the sales process of Penna&Calamaio, a wholesale distributor of stationery items. The sales process occurs between Penna&Calamaio and retail stores.

The process starts with an order from a store specifying the desired goods. The Penna&Calamaio sales office, after verifying inventory, forwards the order to the warehouse, which packs and sends it to the courier and informs the store that the shipment is in progress. If the requested goods are not fully in stock, the sales office estimates the fulfillment time. If the time is less than 10 days, the order proceeds as described; otherwise, it is canceled and the store is informed.

If the store does not receive the order within 5 days of the shipment notification, it cancels the order and informs the sales office. Upon receiving the goods, the store verifies correctness; if everything is in order, payment is made, otherwise the goods are returned, and the process ends. When Penna&Calamaio receives payment confirmation, it sends the invoice to the store. The solution is shown in Figure 5.17.

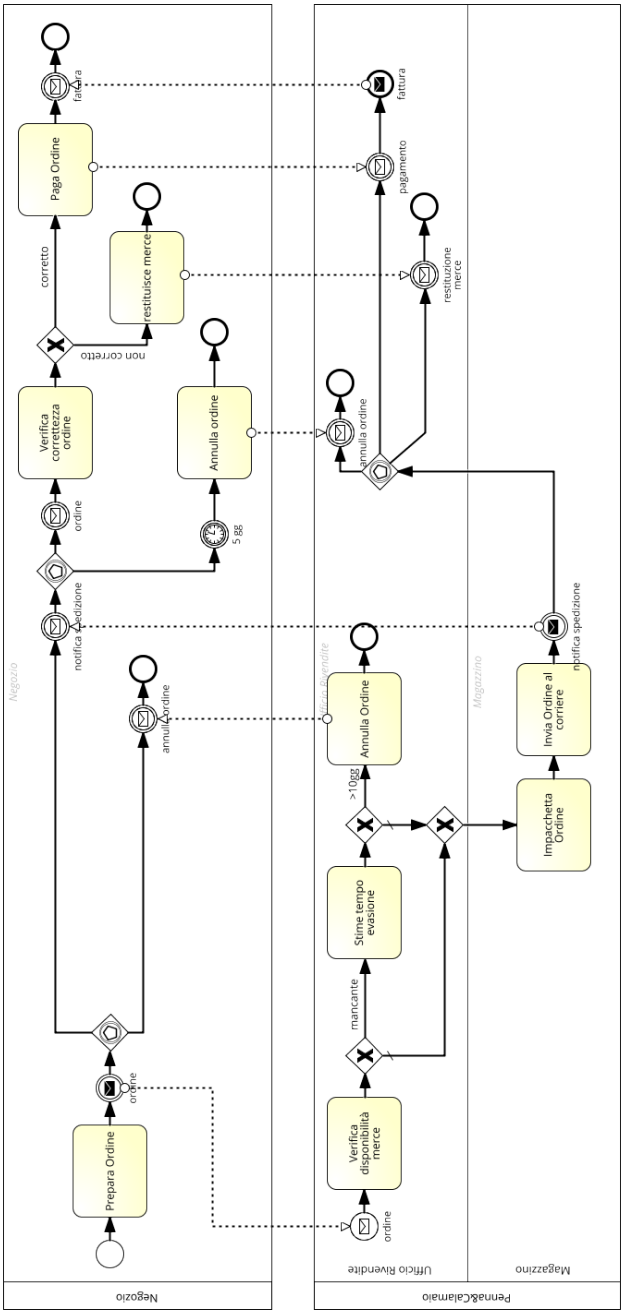


Figure 5.17: *Penna&Calamaio*

5.18 Bank Complaints (ABF)

Model using BPMN the following process for handling bank customer complaints.

Three actors participate in the process: the customer, the bank, and the ABF (a third-party entity that may intervene in dispute resolution). The process begins when a customer decides to file a complaint against a bank. Upon receiving the complaint, the bank processes it and sends a response to the customer once a decision is made. If 30 days pass without a response, the customer can approach the external ABF entity to resolve the dispute, submitting the complaint documentation. The ABF checks the admissibility of the complaint. If not admissible, it notifies the customer, who then waits for the bank's response. If admissible, the ABF notifies the bank.

Upon receiving the notification, the bank halts processing the response and sends the complaint documentation to the ABF. The ABF prepares the investigation and notifies its decision to both the customer (who remains waiting for the bank's response) and the bank. The bank then complies with the ABF decision and communicates the response to both the customer and the ABF. Upon receiving the response, the ABF ends the process.

If the ABF does not receive a response within 30 days, it publishes the bank's non-compliance (while still waiting for the complaint response). Solution in Figure 5.18.

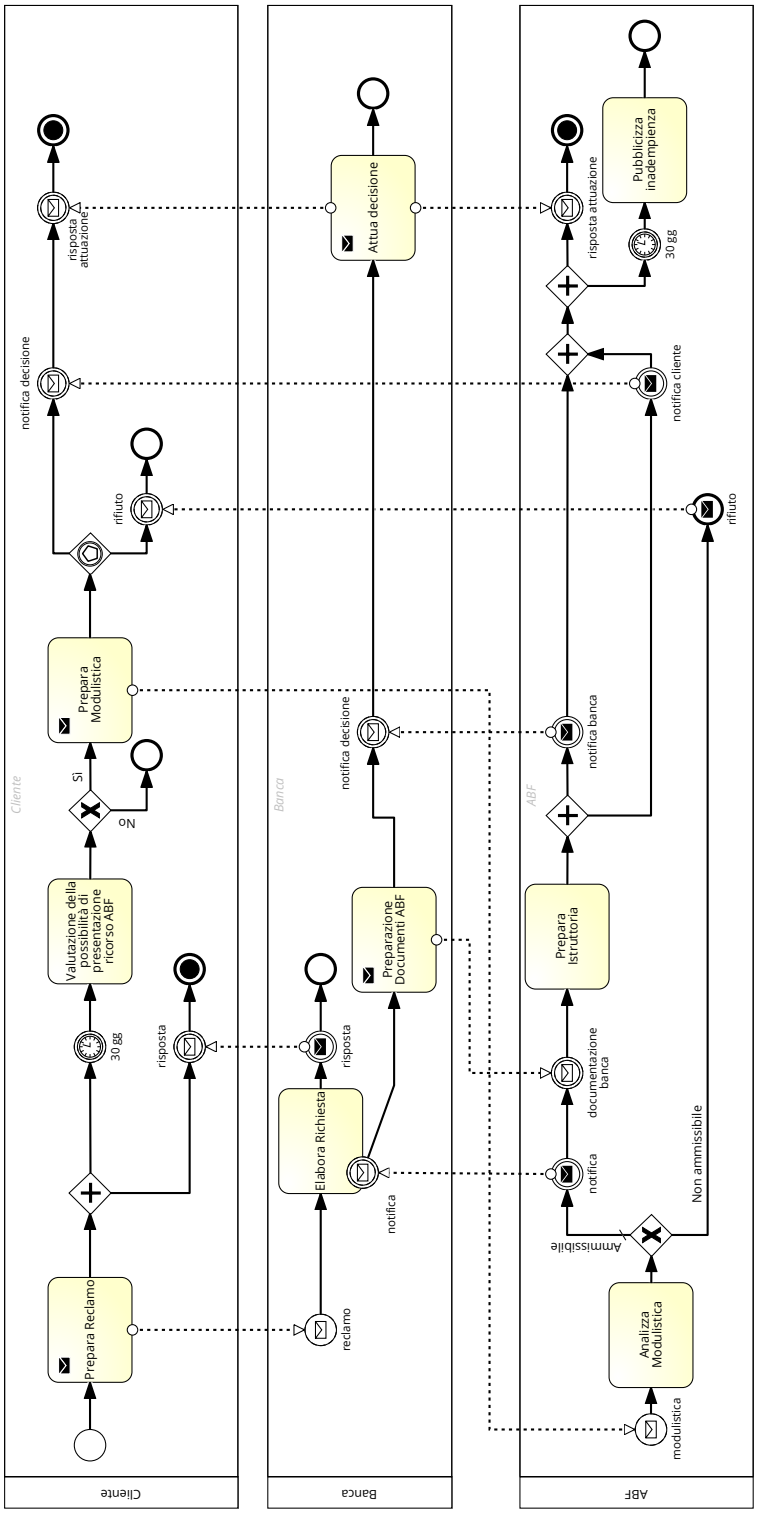


Figure 5.18: *ABF Complaint*

5.19 U2Bike

We want to design the information system for U2Bike, a bicycle manufacturer and retailer. The system must allow users to order a bicycle.

The order management process is as follows. The process begins when the customer selects a model from the catalog and chooses to purchase it. The order is sent to U2Bike's sales department for analysis, then to the production department to estimate production times. The sales department sends the estimate to the customer. The customer can then decide to proceed or cancel. If proceeding, an advance payment is made. Once the sales department receives the payment, production starts. When production is complete, the sales department notifies the customer to collect the bicycle, and both processes end. If production is delayed, new delivery times are estimated, and the sales department notifies the customer. In this case, the customer may cancel the order, sending a notification to the company, which interrupts production and ends the process.

Create the BPMN diagram for the new order management process, modeling both the customer and the company (Figure 5.19).

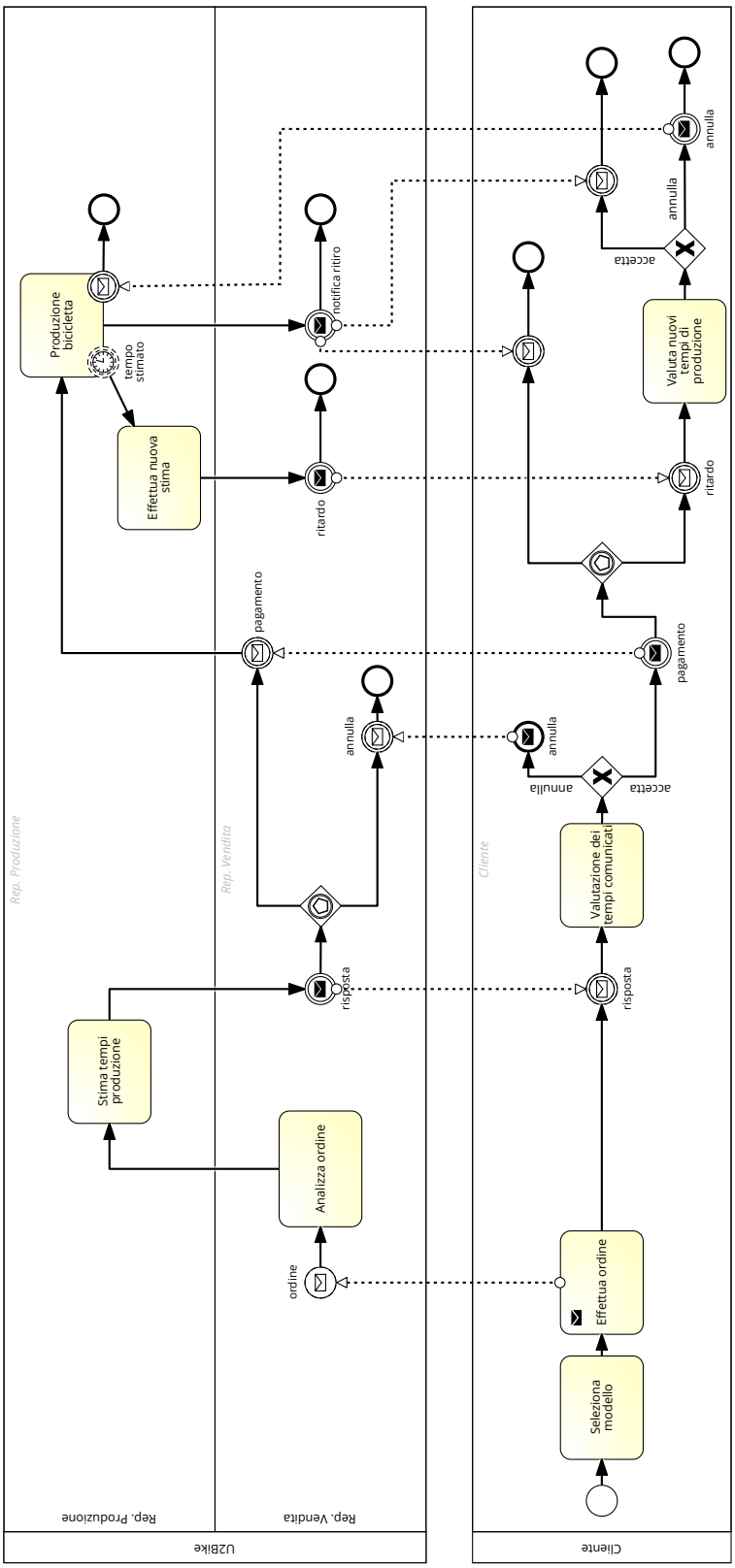


Figure 5.19: *U2Bike Order*

5.20 U2Bike - variation

We want to design the information system for U2Bike, a bicycle manufacturer and retailer. The system must allow users to request bicycle repairs.

The repair process is as follows. The process begins when the customer describes the bicycle malfunction and submits a repair request to U2Bike. The request is received by customer service, which analyzes it and forwards it to the workshop for repair cost estimation. Customer service sends the estimate to the customer, who can decide to proceed or cancel. If proceeding, the bicycle is delivered. Once received, the workshop starts the repair process, first analyzing the vehicle's condition. It then evaluates whether the initial cost estimate was correct. If updated, a notification with the new cost is sent to the customer, who can decide to proceed or cancel. If the initial estimate is correct or the customer accepts the updated estimate, the workshop executes the repair. Once completed, customer service notifies the customer to collect the bicycle, and both processes end.

Create the BPMN diagram for the bicycle repair process, modeling both the customer and the company (Figure 5.20).

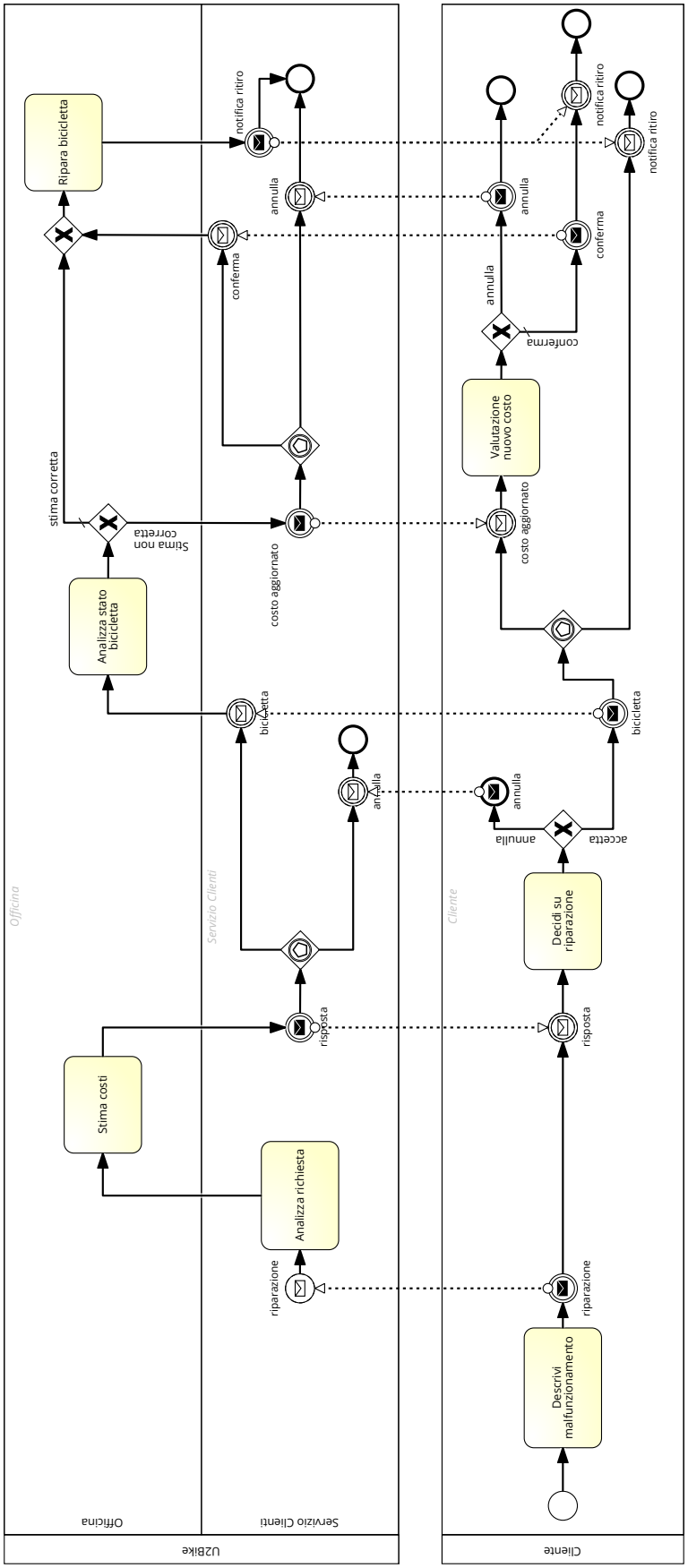


Figure 5.20: *U2Bike Repair*

5.21 CondominiumManagement

We want to design the information system of the company CondominiumManagement, which handles condominium administration.

Describe the process of paying an installment using BPMN notation. The process begins when the administrator enters a new installment to be paid into the system. A notification of the entry is then sent to the individual condominium owner, who reviews the payment request.

If the condominium owner finds no errors, they proceed directly with the payment. If errors are noticed, they can send a verification request to the administrator describing the issue. Upon receiving the request, the administrator examines it and re-sends the payment request, possibly updated (a single communication is sent whether modified or confirmed). The administrator then waits for the condominium owner's payment, who, upon receiving the new request, must proceed with payment.

The verification request can be sent within two weeks; after this period, the administrator ignores any requests and only waits for payment. From the time the initial payment request is sent, the condominium owner has two months to make the payment. If the payment is not made within this period, the administrator sends a reminder. Upon receiving the reminder, the condominium owner stops any ongoing activities (analysis, payment, or waiting for verification) and must make the payment with a late fee. Once payment, with or without a late fee, is made, the company sends a receipt to the customer, and the process ends (Figure 5.21).

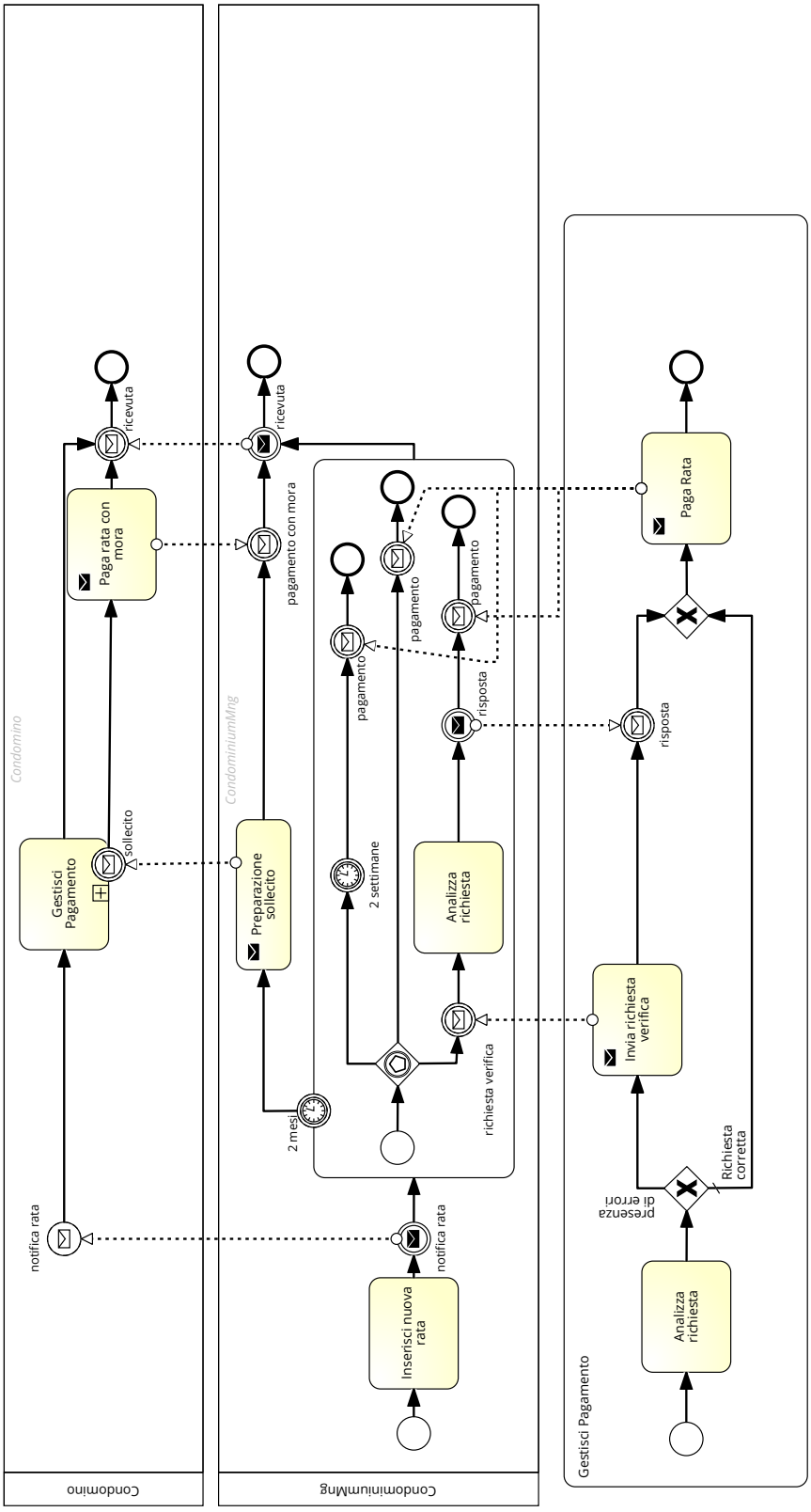


Figure 5.21: *CondominiumManagement*

5.22 Order Management – Using Transactions

A manufacturing company wants to model the order management process using BPMN. The process starts with a customer order request. The administrative office receives the order and must manage it. First, this office asks the Warehouse to pick the requested items. If items are unavailable, the Warehouse must place an order with suppliers. If items do not arrive within 5 days, the order is canceled, a cancellation notice is sent to the customer, and any picked items are made available again.

If everything goes well, the items along with the order are sent to the logistics office, which prepares the package and sends the goods to the customer. Once the goods are shipped, the customer may return them within seven days. If this happens, the administration notes the return and sends the goods back to the warehouse. If no return occurs, the administrative office closes the process. (Do not model the customer) (Figure 5.22).

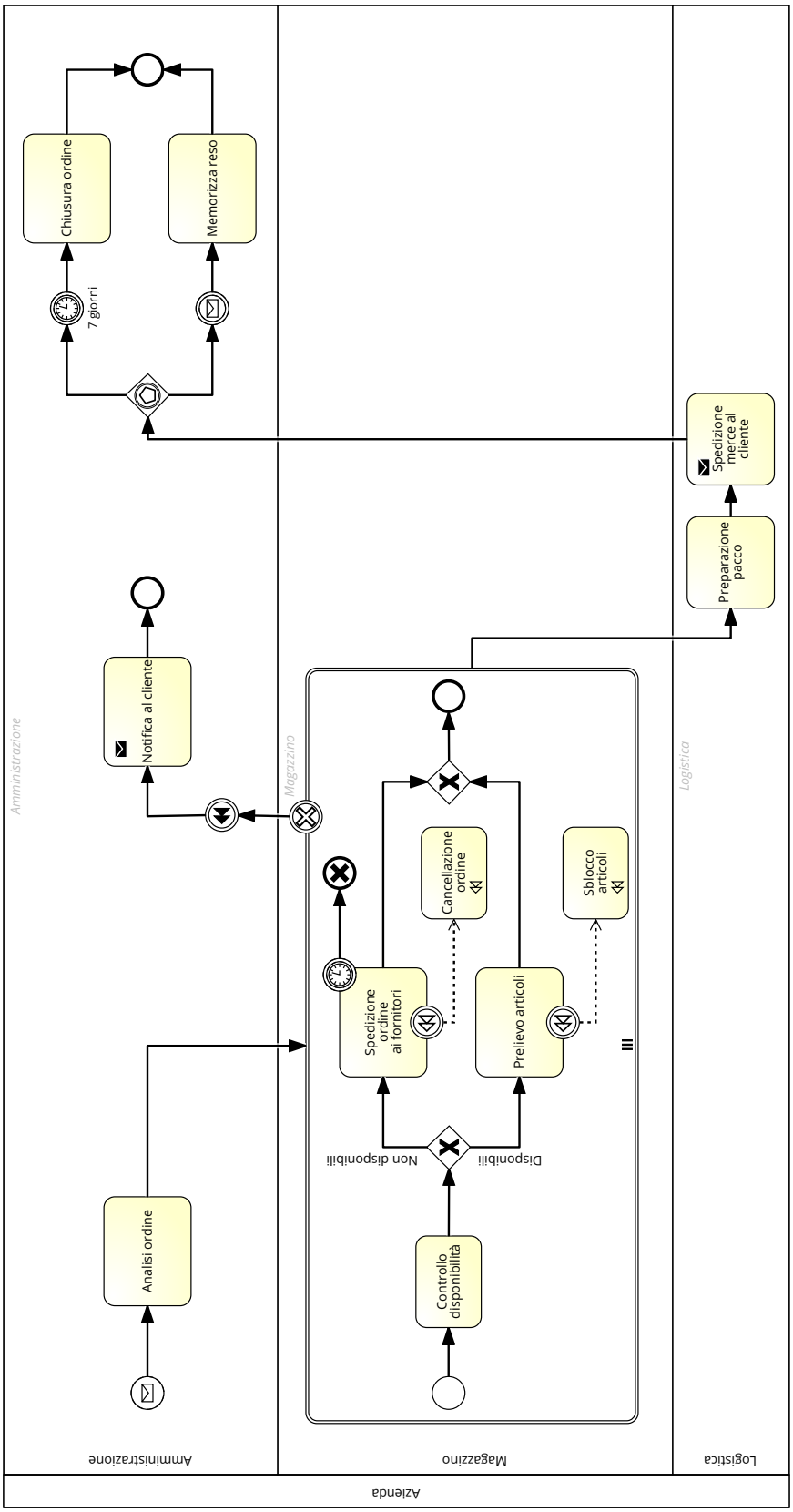


Figure 5.22: *Order Management*

5.23 Expense Reimbursement – Using Transactions

A company wants to model using BPMN the process for reimbursing employee travel expenses. When an employee returns from a trip, they must send a form to the HR office containing a list of expenses incurred. Upon receiving the form, the HR office verifies whether the employee exists in the system. If not, the employee's data is added to the HR database. The expense list is then analyzed for approval. If the total expenses are below €1000, the reimbursement is automatically approved; if above €1000, further verification is required. If irregularities are found and the reimbursement is denied, the employee receives an email notification. If approved, the reimbursement is automatically deposited into the employee's account.

At any point during the review, the employee may send a Request to Modify the reimbursable amount. In this case, the request is recorded, and the form is re-analyzed. Moreover, if the form is not analyzed within 30 days, the process is interrupted. If this occurs after the reimbursement has been credited, the funds must be reversed. (Do not model the employee) (Figure 5.23).

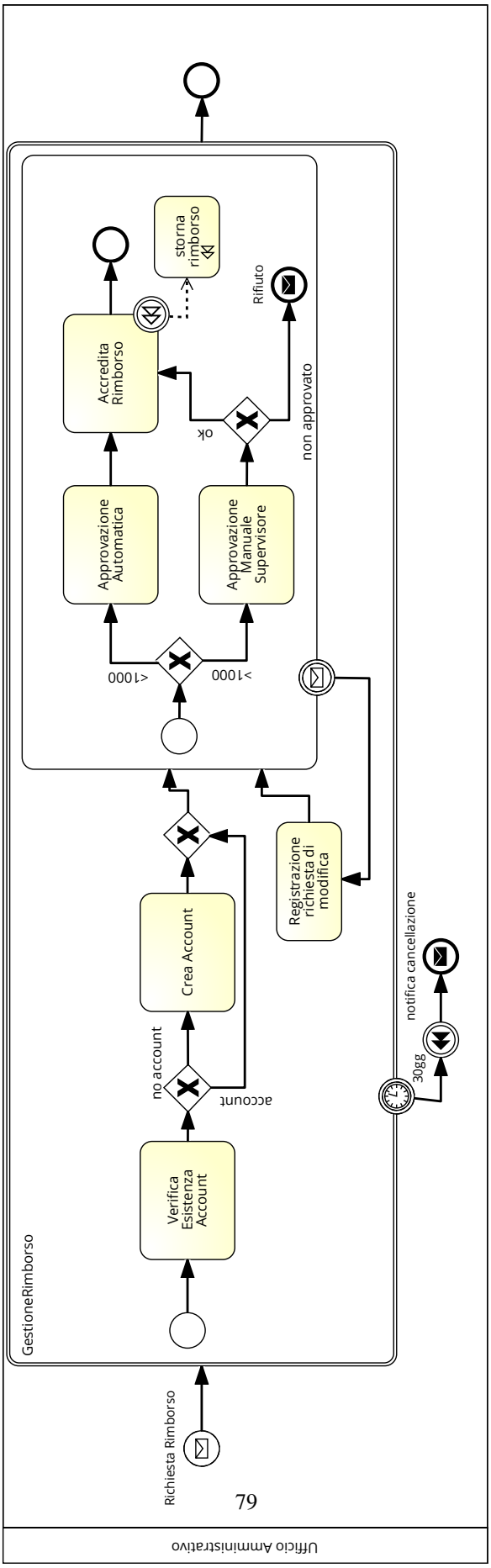


Figure 5.23: *Company Travel Reimbursement*

5.24 CRM – Using Transactions

Every Saturday at 03:00, an automatic process runs routine operations on the database of the manufacturing company Texile.

The process performs a full database backup and sequentially checks all tuples in the ”Unpaid Invoices” table (containing overdue and unpaid invoices) for new or updated tuples since the last backup. If no new tuples are found, the process analyzes the CRM system as described below. If new tuples exist, the data of customers who have not paid invoices on time are inserted into a new table, and the system checks if it is the first time the customer has missed a payment. Customers with multiple late payments are stored in a new ”Messages” table. At the end of this procedure, the ”Messages” table (if created) is sent to the Accounting Office. These activities must be completed by 14:30; otherwise, a warning is sent to the supervisor without interrupting the process. If the analysis of new tuples ends exceptionally, the ”Messages” table is deleted, and the process ends.

Once all the described activities are complete, the CRM system is checked to verify that all late payments received in the last week are correctly recorded. Any inconsistencies found are corrected. The entire process must be completed within 13 hours; otherwise, a notification is sent to the supervisor (again without interrupting the process) (Figure 5.24).

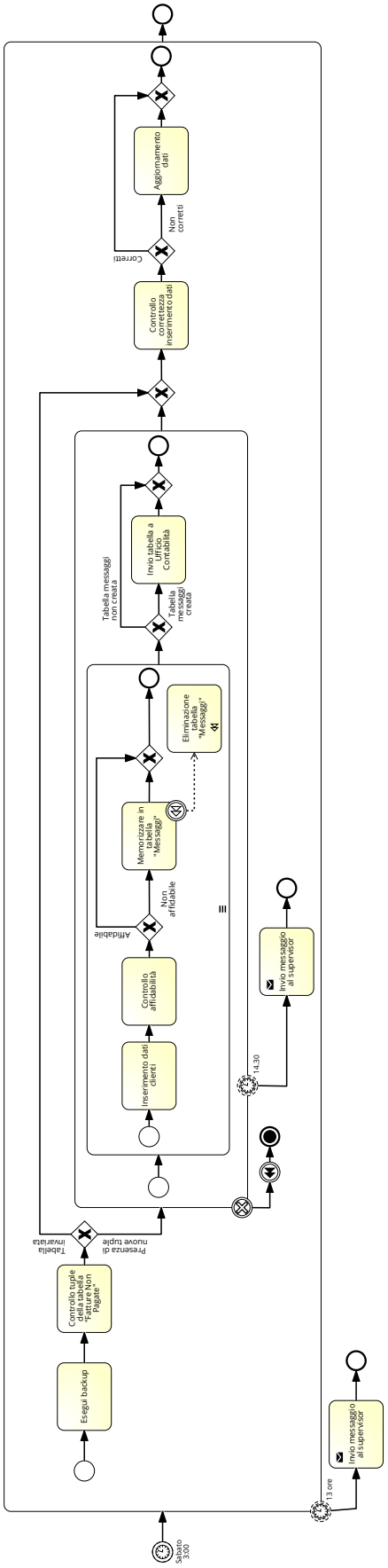


Figure 5.24: *Unpaid Invoices Backup*