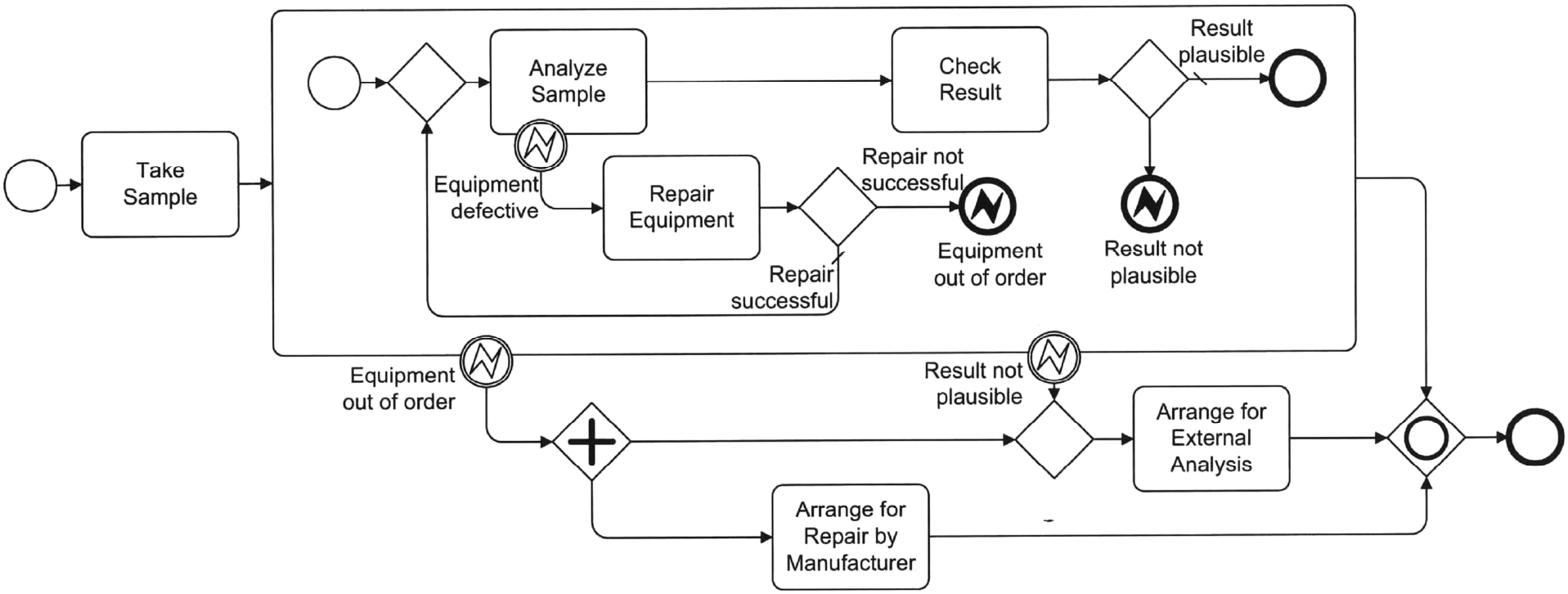
- The *intermediate error events* can only be attached to activity / process boundaries.
- In a process in a laboratory, after taking a sample, the sub-process is started. First the sample is analyzed. This is usually followed by checking the result. If the result is plausible, the sub process finishes. If the result is not plausible, this is an error, and the throwing error and event "Result not plausible" is reached.
- In this case, the sub-process is aborted and the thrown error is caught by the error intermediate event (with the same name) at the sub-process boundary. In this case, the exception flow is followed, and the activity "Arrange for external analysis" is triggered.



- In the sub-process, another error may occur: if during the activity "Analyze Sample" it turns out that the equipment is defective (here also modeled with a catching error event), it will be repaired.

- If repair has been successful, the analysis of the sample will be started again. In the case of an unsuccessful repair, the throwing error event "Equipment out of order" occurs.

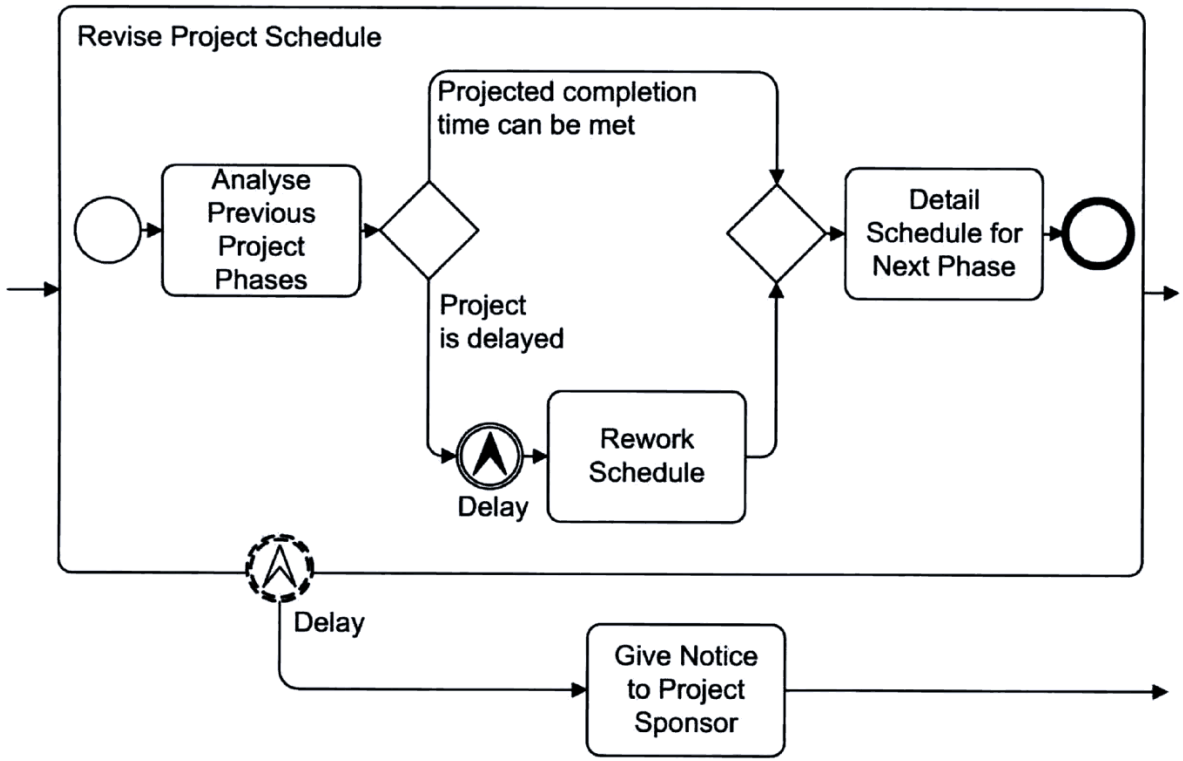
- Again, the sub-process is aborted and the error is thrown to the intermediate error event (with the same name) that triggers the exception flow to the activity “Arrange for repair by Manufacturer” in addition to “Arrange for External Analysis”.

- Note that if there are several error intermediate events attached to a sub-process boundary, they should be always labeled.

- Since unlabeled error intermediate event always catches any type of error, the resulting implicit parallel flows are difficult to understand.

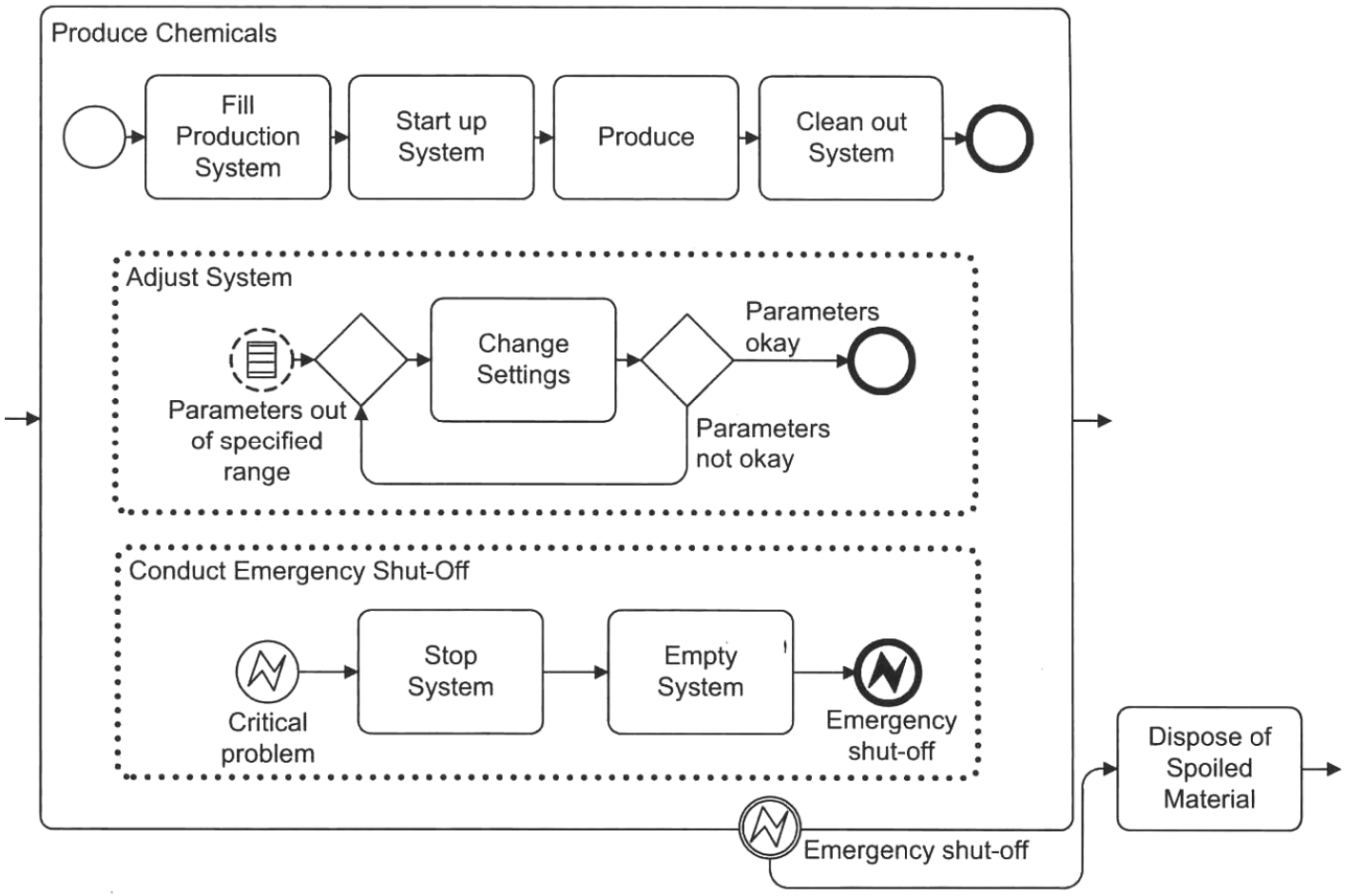
- *Escalation events* are rather similar to error events. Error events are mainly used for technical problems, escalation events primarily stands for problems on a business level, e.g., a task not completed, a goal not reached, a require agreement not achieved.

- In contrast to errors which always abort the activity, escalations come in both variants: the interrupting and the non-interrupting event. Non-interrupting escalation events are regarded as the default.



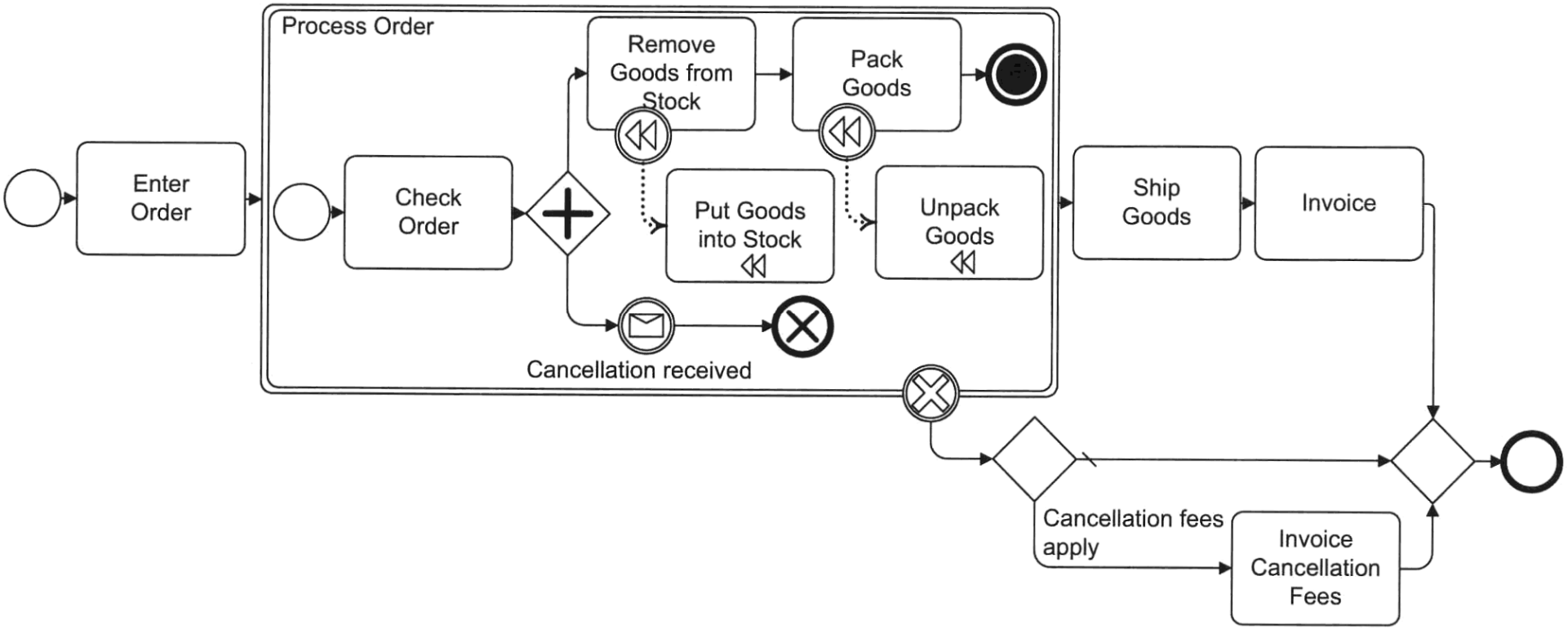
- Note that if an escalation is required to abort the entire activity, the throwing escalation event in the sub-process needs to be an end event. If a throwing escalation intermediate event was used, the subsequent sequence flows could never be used, because the sub-process would be aborted by the escalation.

- An *Event Sub-Process* is triggered by an event occurring during process execution, and is characterized by a dotted border. The sub-process "Produce Chemicals" contains two events sub-process, the first one is non-interrupting, i.e. the system may be adjusted several times during production. The second one is interrupting, and also throws an error by an end event, which is caught in turn by an error intermediate event (Emergency shut-off) so as to emit a token to the exception flow.



About transactions

- Parts of business processes can be treated as transactions, i.e., complete units of work. A transaction can be either carried out in its entirety, or not at all. If something goes wrong, the transaction must be rolled back.
- A BPMN transaction is drawn with a double line border. Aborting a transaction automatically reverses the effects of the activities that have already been carried out. For this purpose, a *compensation* activity is assigned to each activity that requires it.



- In the above example, the activity “Put Goods into Stock” contains a rewind-symbol (two white arrowheads) to express it as a compensation activity. Indeed it compensates the activity “Remove Goods from Stock”. The compensation association is a dotted line with an arrow. Do not confuse it with message flow.

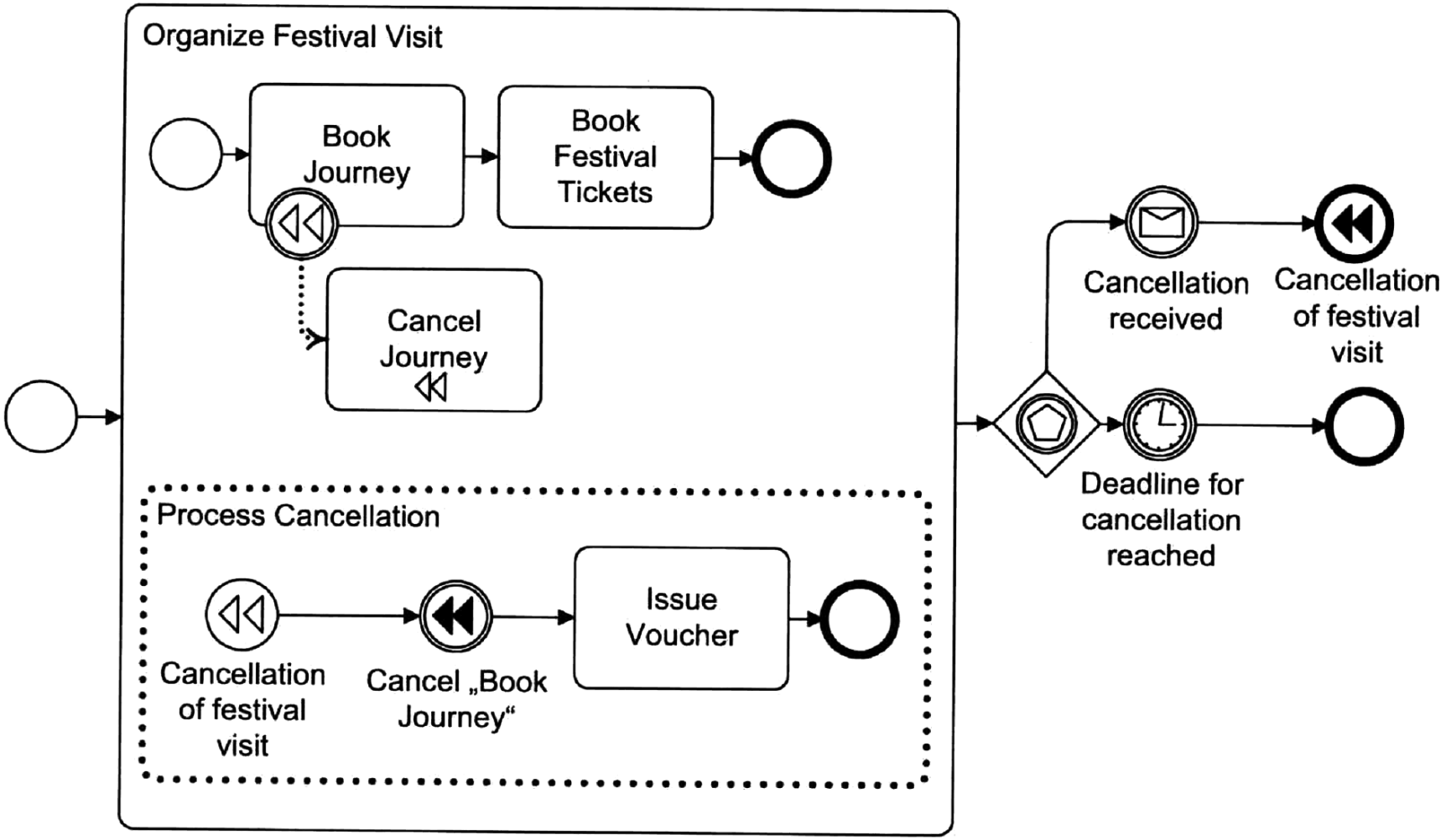
- The activity “Check order” does not have a compensating activity, because it does not have any effects that need to be reversed.

- The throwing end event of type “cancel” is used to abort a transaction. Once the required compensation has been carried out, the exception flow starts as the catching cancel intermediate events.

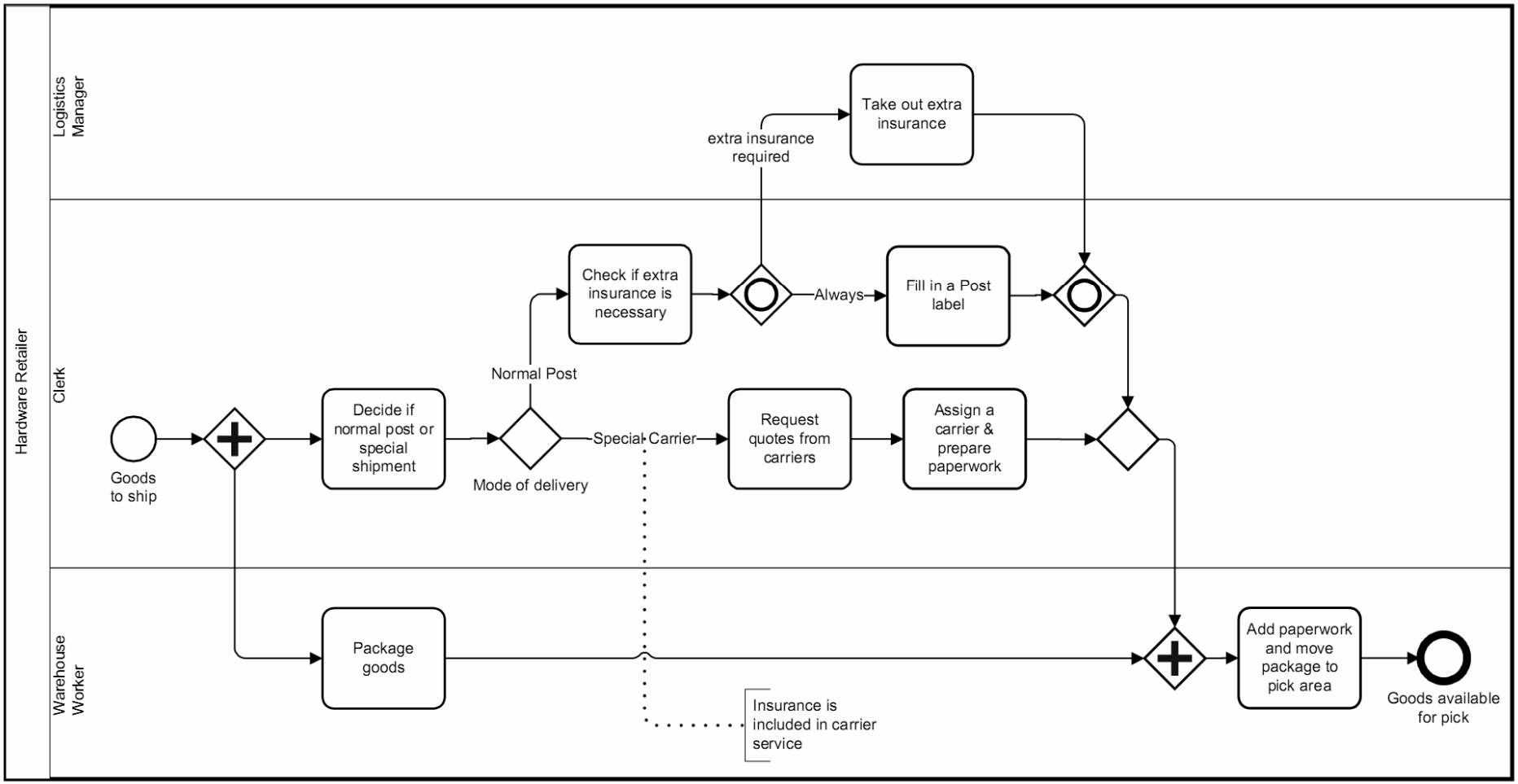
- Note that the terminate end event is needed to cancel the token waiting for receiving cancellations.

- Compensation can also be handled by event sub-processes. They have a compensation start event, and can be started when a throwing compensation event of the same name occurs **after the surrounding process is already finished**.

- In contrast with compensation events, error/escalation events can be caught only in the surrounding process during its execution.


Organize Festival Visit
Book Journey
Book Festival Tickets
Cancel Journey
Process Cancellation
Cancellation of festival visit
Cancel „Book Journey“
Issue Voucher
Cancellation received
Cancellation of festival visit
Deadline for cancellation reached

- Exercise: describe in natural language the following process diagram, concerning the shipment process of *a hardware retailer*.

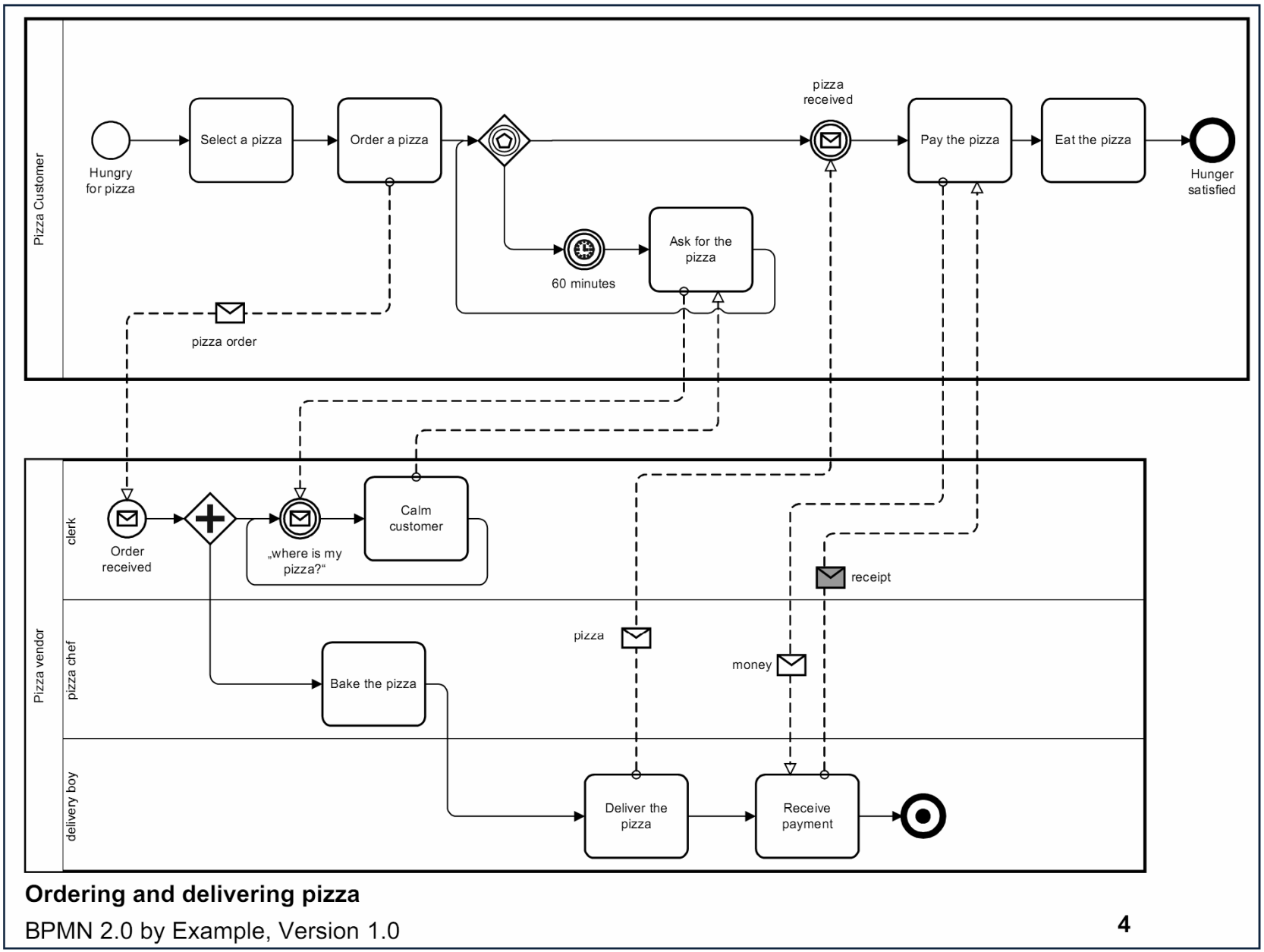


Shipment Process of a hardware retailer

BPMN 2.0 by Example, Version 1.0

3

- Exercise: describe in natural language the following process diagram, concerning the interaction between *a pizza customer and the vendor*.



- Exercise: describe in natural language the following process diagram, concerning the *incident management process*.