Chapter 3 - Essential Process Modeling

The First Steps With BPMN

- **Events**: things that happen instantaneously, represented by circles.
  - **Start Events**: represented by circles with a thin border.
  - **End Events**: represented by circles with a thick border.
- **Activities**: units of work with a duration, represented by rounded rectangles.
- **Arcs**: relation of sequence event or activity A is followed by event or acctivity B, represented by arrows with a full arrow head.
- **Token**: are created at the start event and flow throughout the process model until they are destroyed in an end event, represented by colored dots.

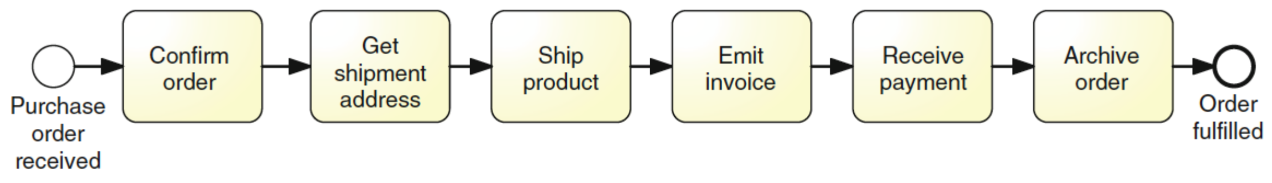


Fig. 3.1 The model of a simple order-to-cash process

Here the token is created in the start event (black dot), then it is at the stage of shipping and the other just received the payment and is about to start archiving the order.

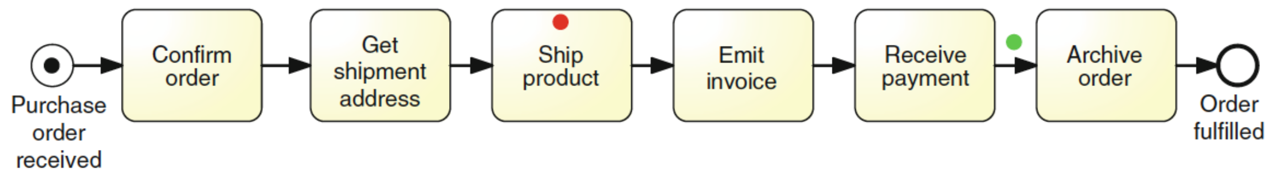


Fig. 3.2 Progress of three instances of the order-to-cash process

Always label activities and events as well (to know the outcome of the process).

Label naming conventions

Avoid labels with more than five words (excluding prepositions and conjunctions).
Capitalize the first word for labels.
Don't use verbs like "to make", " to do", "to perform", "to conduct". Use verbs that capture the specifics of the activity.
Don't use words like "process", "order" as both verbs ("to process", "to order") and as nouns ("a process", "an order"), choose to use always as a noun or always as a verb.

- **For activities**: verb in the imperative form followed by a noun referring to a business object. Ex: Approve order.
  - The noun may be preceeded by an adjective. Ex: Issue driver license.

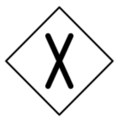
    - The verb may be followed by an adverbial clause to explain how the action is being done. Ex: Renew driver license via offline agencies.
- **For events**: noun referring to a business object and end with a past partciple. Ex: Invoice emitted.
    - The noun may be preceeded by an adjective. Ex: Urgent order sent.
- **For process models**: use a noun preceeded by an adjective. Ex: "loan orginination", "order fulfillment", "claim handling".
    - Can be done by normalizing the verb describing the main action of a process. Ex: fulfill oder becomes order fulfillment.
    - **We do not capitalize the 1st word of process names. Ex: order-to-cash process.**

Branching and Merging

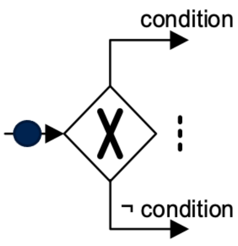
- When two or more activities are alternative to each other, we say they are **mutually exclusive**. Ex: the approval or the rejection of a claim are two activities that exclude each other.
- When two or more activities are not interdependent, they are **concurrent**, can be performed in parallel.
- These behaviours are modeled by **gateways**. Implies there is a gating mechanism that either allows or disallows passage of tokens through the gateway. Tokens can be merged together on input or split apart on output depending on the gateway type.
    - **Split gateway**: a point where the process flow diverges, has one incoming sequence flow and multiple outgoing sequence flows (branches that diverge).
    - **Join gateway**: point where the process flow converges, has multiple incoming sequence flows (branches to be merged) and no outgoing sequence flow.

Exclusive Decisions

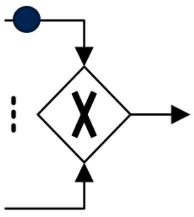
- **XOR split**: used to model two or more alternatice activities like approval or rejection.
- **XOR join**: used to merge two or more alternative branches that may have been forked with an XOR-split. Represented by an empty diamond or a diamond with and X. As soon as it receives a token it will advance to the output arc.



An *XOR Gateway* captures decision points (XOR-split) and points where alternative flows are merged (XOR-join)



XOR-split ➔ takes **one** outgoing branch



XOR-join ➔ proceeds when **one** incoming branch has completed

Example 3.2 Let us consider the following invoice checking process.

> As soon as an invoice is received from a customer, it needs to be checked for mismatches. The check may result in any of the following three options: (i) there are no mismatches, in which case the invoice is posted; (ii) there are mismatches but these can be corrected, in which case the invoice is resent to the customer; and (iii) there are mismatches but these cannot be corrected, in which case the invoice is blocked. Once one of these three activities is performed the invoice is parked and the process completes.

To model this process we start with a decision activity, namely "Check invoice for mismatches" following a start event "Invoice received". A *decision activity* is an activity that leads to different outcomes. In our example, this activity results in three possible outcomes, which are mutually exclusive; so we need to use an XOR-split after this activity to fork the flow into three branches. Accordingly, three sequence flows will emanate from this gateway, one towards activity "Post invoice", performed if there are no mismatches, another one towards "Re-send invoice to customer", performed if mismatches exist but can be corrected, and a third flow towards "Block invoice", performed if mismatches exist which cannot be corrected (see Figure 3.4). From a token perspective, an XOR-split routes the token coming from its incoming branch towards one of its outgoing branches, i.e. only one outgoing branch can be taken.

When using an XOR-split, make sure each outgoing sequence flow is annotated with a label capturing the condition upon which that specific branch is taken. Moreover, always use mutually exclusive conditions, i.e. only one of them can be true every time the XOR-split is reached by a token. This is the characteristic of the XOR-split gateway. In our example an invoice can either be correct, or contain mismatches that can be fixed, or mismatches that cannot be fixed: only one of these conditions is true per invoice received.

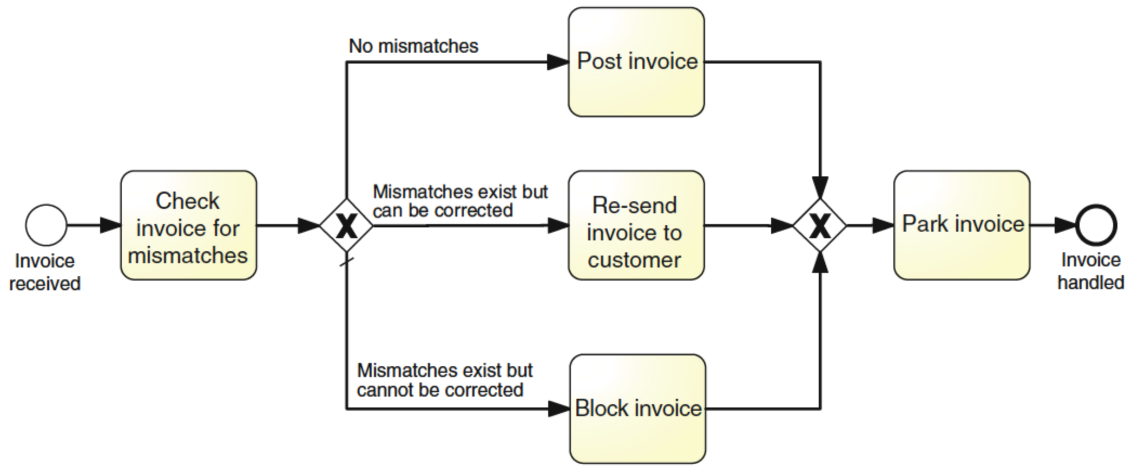

Invoice received
Check invoice for mismatches
No mismatches
Post invoice
Mismatches exist but can be corrected
Re-send invoice to customer
Mismatches exist but cannot be corrected
Block invoice
Park invoice
Invoice handled

Fig. 3.4 An example of the use of XOR gateways

- **Default flow**: The oblique cut in one of the labeled flows is used to indicate the flow that will be taken by the token coming for the XOR splot in case the conditions to all the other outgoing flows evaluate to false.

Parallel Execution

- **AND gateway**: used to model when two or more activities do not have any order dependencies on each other and can be executed in parallel. It is represented by a diamond with a + mark.
    - **AND split**: model parallel execution of two or more branches.
    - **AND join**: synchronize execution of two or more parallel branches (we proceed once all branches have been completed).

Example 3.3 Let us consider the security check at an airport.

> Once the boarding pass has been received, passengers proceed to the security check. Here they need to pass the personal security screening and the luggage screening. Afterwards, they can proceed to the departure level.

This process consists of four activities. It starts with activity "Proceed to security check" and finishes with activity "Proceed to departure level". These two activities have a clear order dependency: a passenger can only go to the departure level after undergoing the required security checks. After the first activity, and before the last one, we need to perform two activities which can be executed in any order, i.e. which do not depend on each other: "Pass personal security screening" and "Pass luggage screening". To model this situation we use an AND-split linking activity "Proceed to security check" with the two screening activities, and an AND-join linking the two screening activities with activity "Proceed to departure level" (see Figure 3.5).

The AND-split *splits* the token coming from activity "Proceed to security check" into two tokens. Each of these tokens independently flows through one of the two branches. This means that when we reach an AND-split, we take all outgoing branches (note that an AND-split may have more than two outgoing arcs). As we said before, a token is used to indicate the state of a given instance. When multiple tokens of the same color are distributed across a process model, e.g. as a result of executing an AND-split, they collectively represent the state of an instance. For example, if a token is on the arc emitting from activity "Pass luggage screening" and another token of the same color is on the arc incident to activity "Pass personal security screening", this indicates an instance of the security check process where a passenger has just passed the luggage screening but not yet started the personal security screening. □

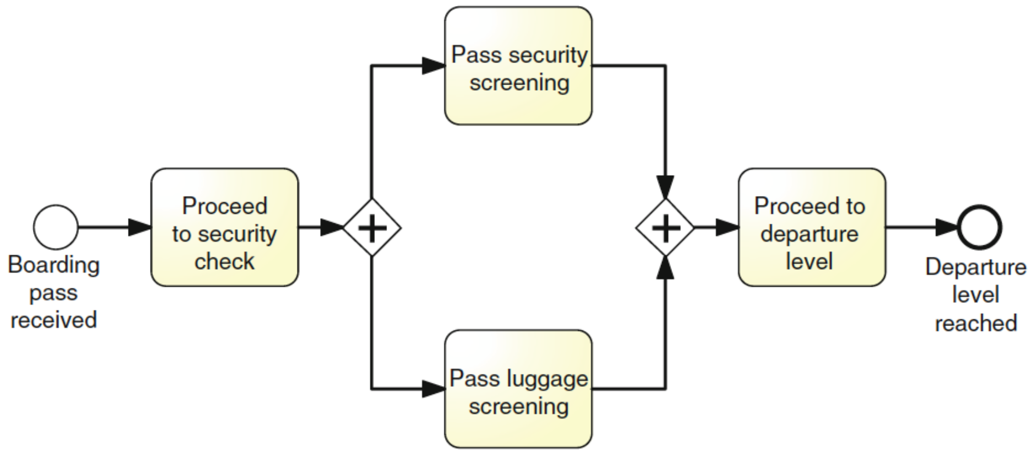


Fig. 3.5 An example of the use of AND gateways

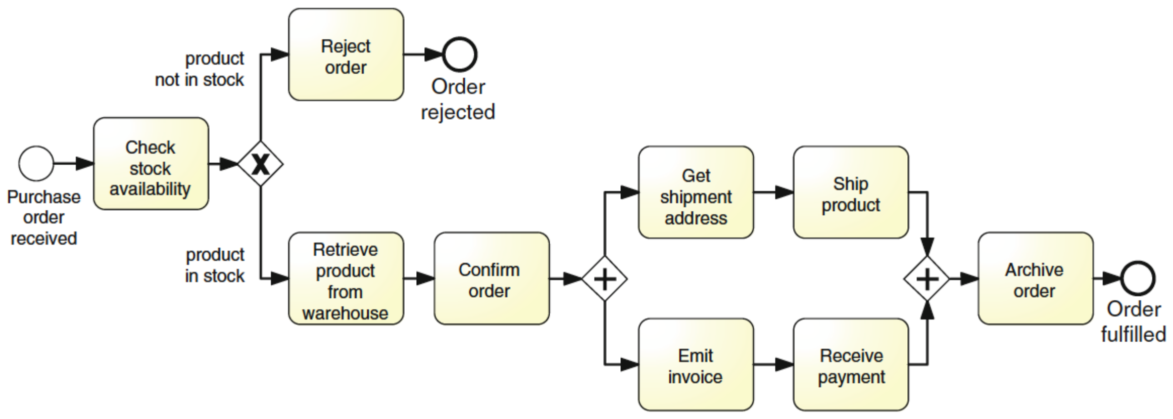


Fig. 3.6 A more elaborated version of the order-to-cash process diagram

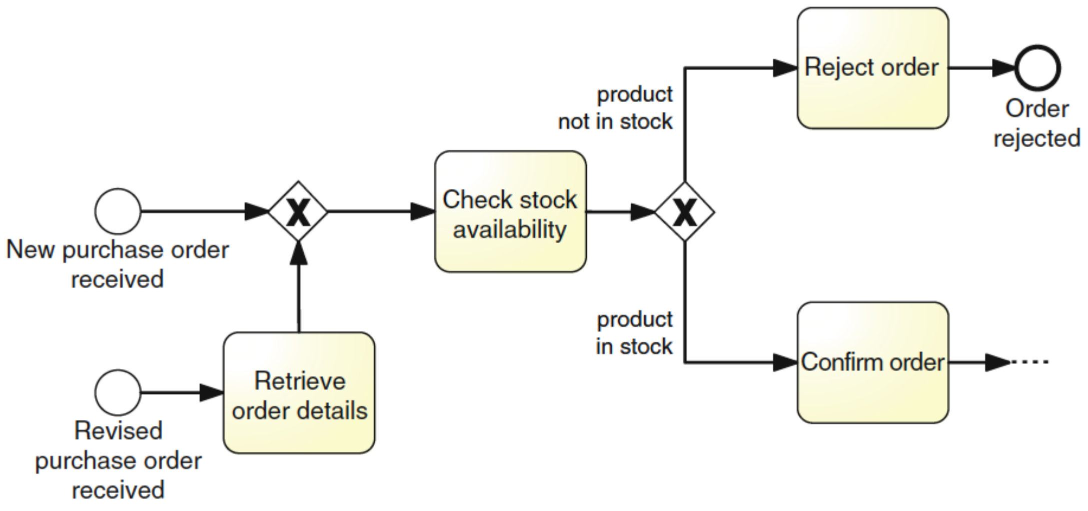


Fig. 3.7 A variant of the order-to-cash process with two different triggers

Business Process Modelling Tools

- **Pen & Paper**: to make intial sketch, not good for systematic knowledge and sharing across an organization.
- **Haptic**: tools that rely on physical objects like post-its or sticky notes, these tools stimulate the engagement of process stakeholders.
- **Single-User**: microsoft visio
- **Multi-tenant**: available to multiple users, typically within the same organization.

Inclusive Decisions

- Sometimes we need to take one or more branches after a decision activity.

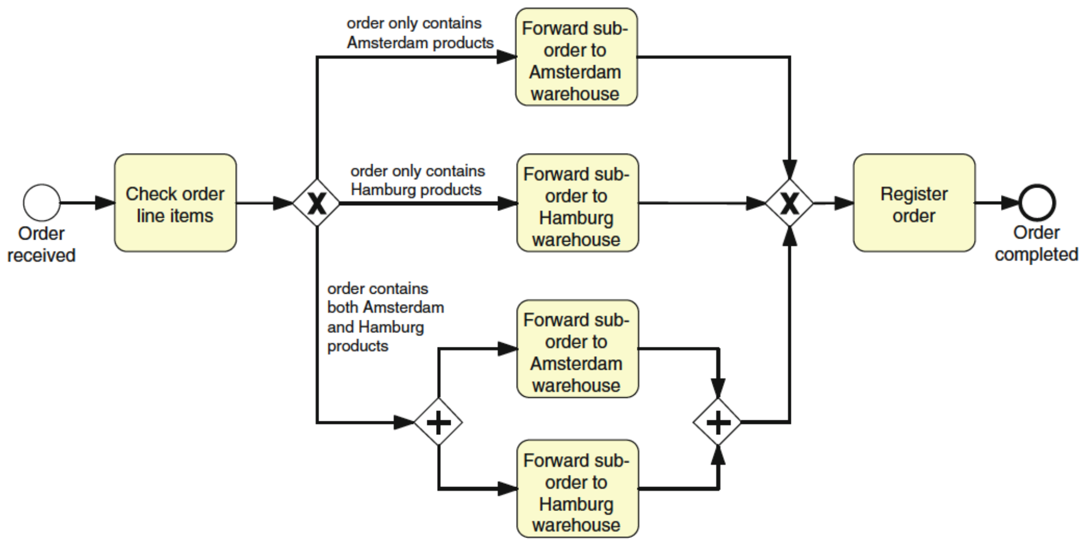

order only contains Amsterdam products
Forward sub-order to Amsterdam warehouse
Order received
Check order line items
order only contains Hamburg products
Forward sub-order to Hamburg warehouse
Register order
Order completed
order contains both Amsterdam and Hamburg products
Forward sub-order to Amsterdam warehouse
Forward sub-order to Hamburg warehouse

Fig. 3.8 Modeling an inclusive decision: first trial

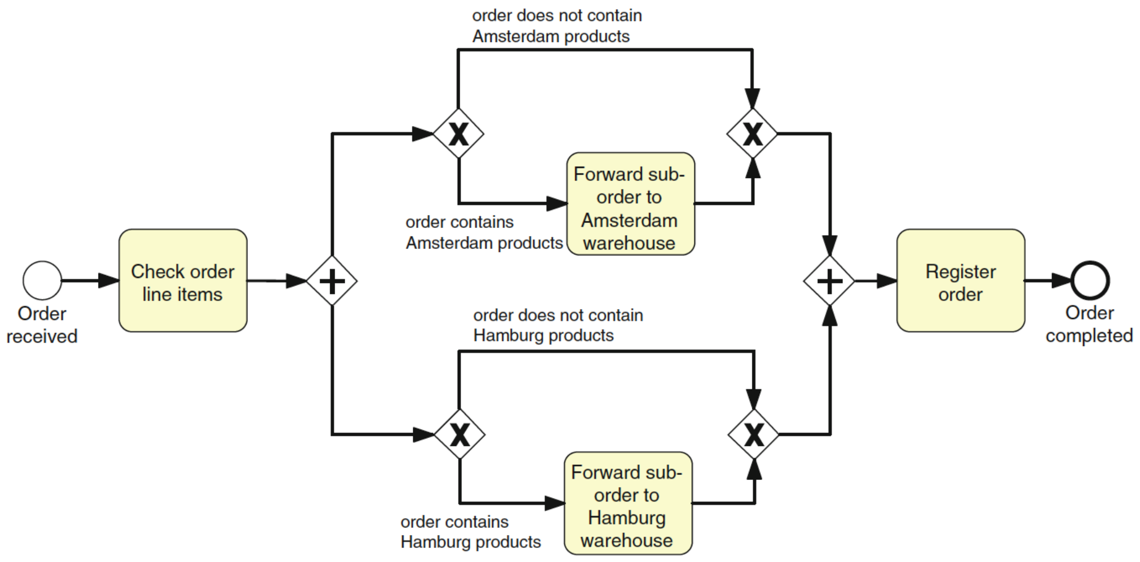
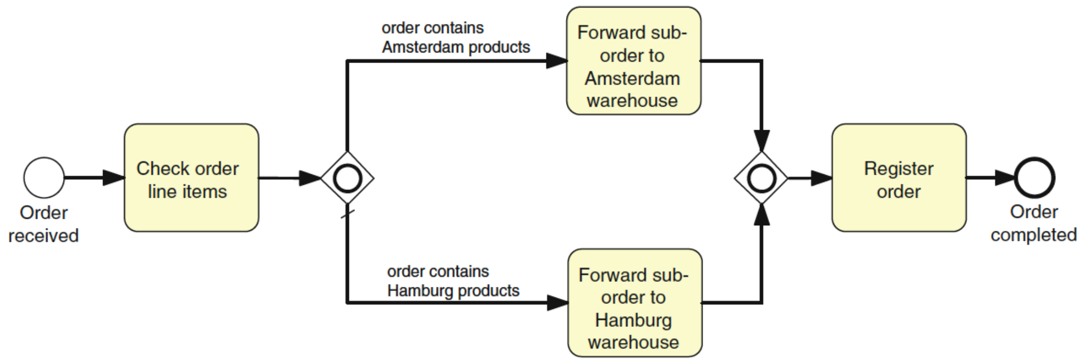

order does not contain Amsterdam products
Forward sub-order to Amsterdam warehouse
order contains Amsterdam products
Order received
Check order line items
Register order
Order completed
order does not contain Hamburg products
order contains Hamburg products
Forward sub-order to Hamburg warehouse

Fig. 3.9 Modeling an inclusive decision: second trial

- Both these diagrams have problems, we will use an **OR gateway** instead.
    - **OR split**: takes the input token and generates a number of tokens equivalent to the number of output conditions that are true, this number can be at least one and at most the total number of outgoing branches, can also have a default flow (taken only when all other conditions are false).

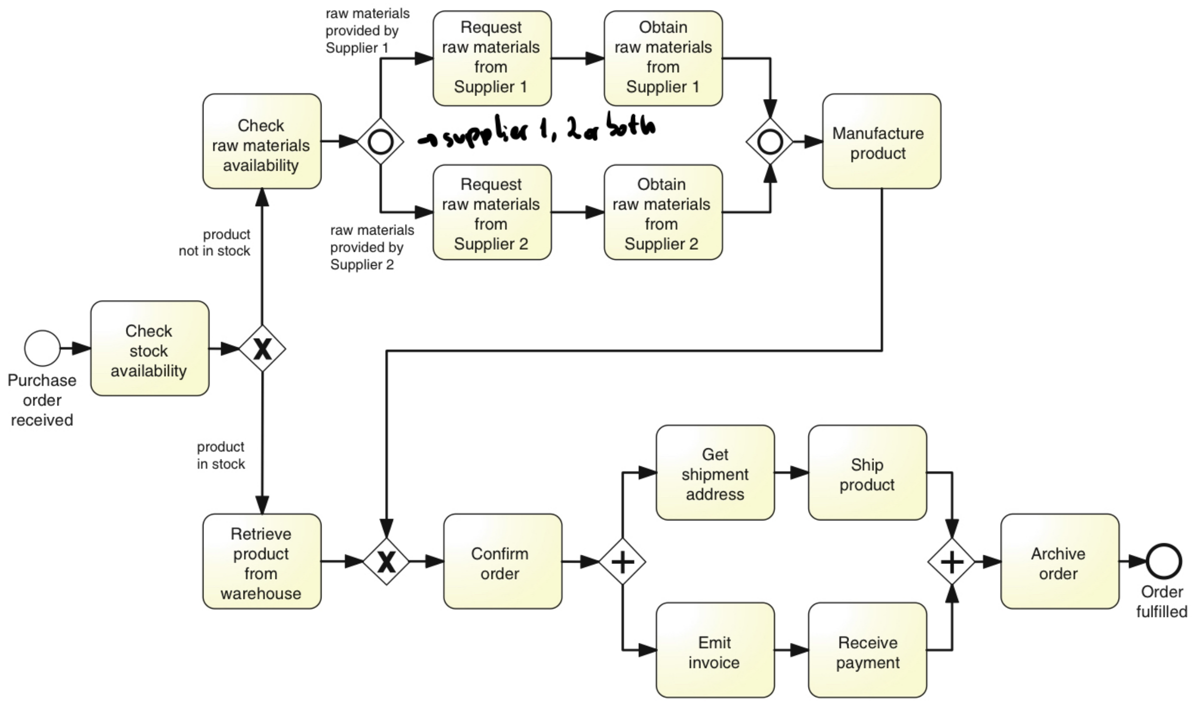
- **OR join**: will wait for all the other active branches to complete.



Order received
Check order line items
order contains Amsterdam products
Forward sub-order to Amsterdam warehouse
order contains Hamburg products
Forward sub-order to Hamburg warehouse
Register order
Order completed

Fig. 3.10 Modeling an inclusive decision with the OR gateway

- When should we use an OR join?
  - When we need to synchronize control from a preceeding OR split.
  - There are other non obvious ways to use but we avoid those.



Purchase order received
Check stock availability
product not in stock
product in stock
Check raw materials availability
raw materials provided by Supplier 1
raw materials provided by Supplier 2
Request raw materials from Supplier 1
Obtain raw materials from Supplier 1
→ supplier 1, 2 or both
Request raw materials from Supplier 2
Obtain raw materials from Supplier 2
Manufacture product
Retrieve product from warehouse
Confirm order
Get shipment address
Ship product
Emit invoice
Receive payment
Archive order
Order fulfilled

Fig. 3.12 The order-to-cash process model with product manufacturing

Rework and Repetition