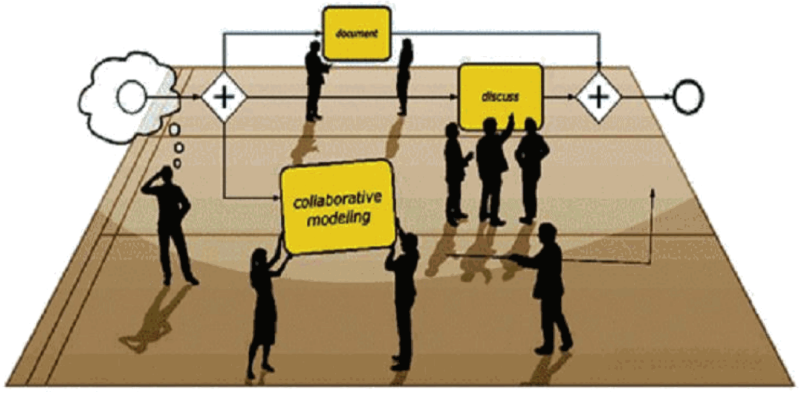
"Large and complex organizations are a tangible manifestation of advanced technology, more than machinery itself." (J.K. Galbraith)

BPMN Modeling and Simulation

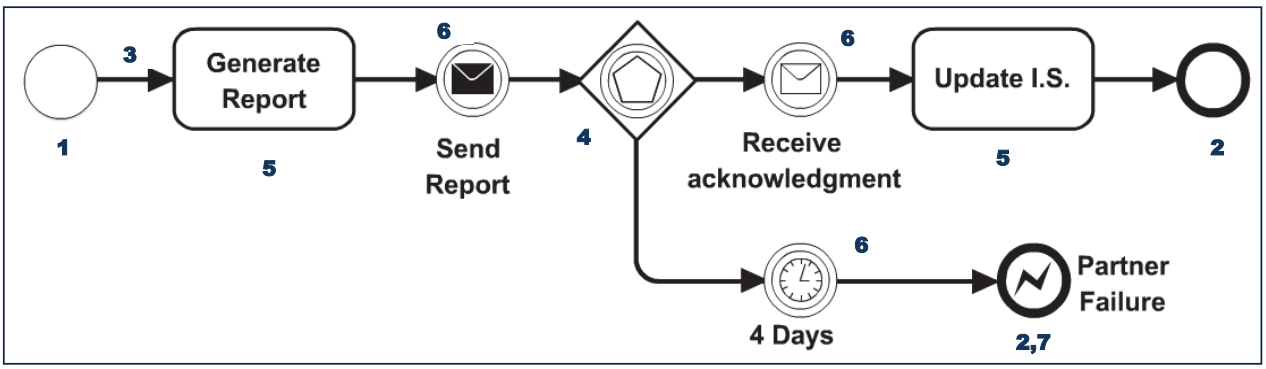


Lectures

Mario G. Cimino, Department of Information Engineering, Center for Logistics Systems

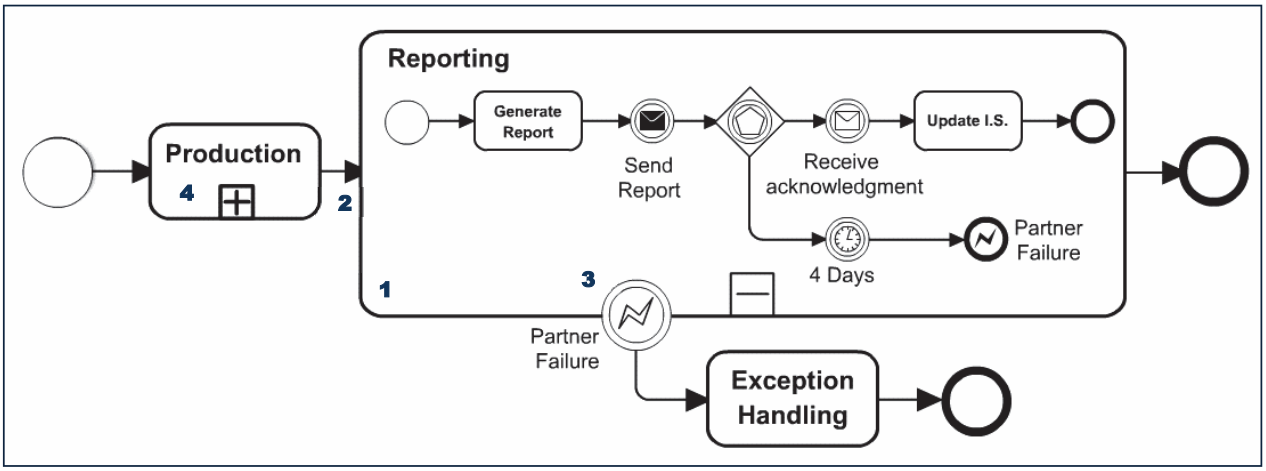
Business Process Model and Notation (BPMN 2.0)

- *BPMN* (OMG, 2005) is specialized in representing the behavior of processes concerning the **control flow**, with the concept of a *token* traversing the process structure.
- A *Start Event*[1] generates a token that will be consumed by an *End Event*[2]. The path of tokens is managed by a network of *Sequence Flow*[3], *Gateway*[4], *Activity*[5] and *Intermediate Event*[6], within the process.
- *Race pattern*: there is a race between two intermediate events[6] after the event-based gateway[4], i.e., "receive acknowledgment" and "four days elapsed". When the message is not received before the 4 days cutoff date, the execution is diverted from normal execution flow in order to raise a 'Partner Failure' error (throw semantic)[7].



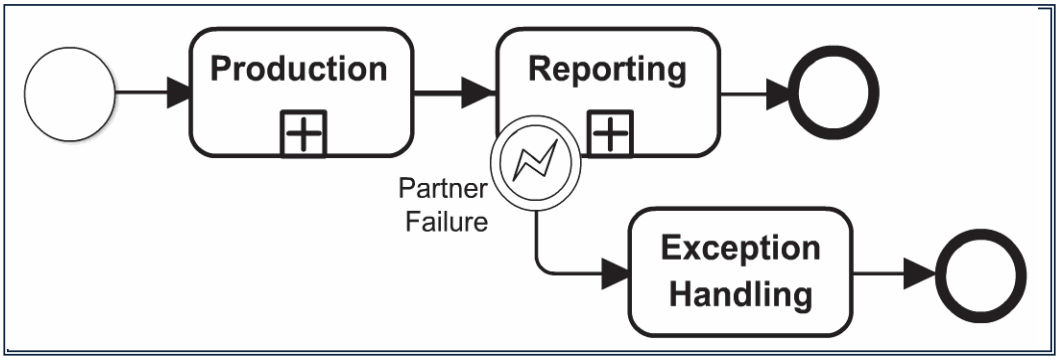
http://www.iet.unipi.it/m.cimino/bpm/res/mov01

- *Interruption pattern*: the previous model is reused and embedded as the 'Reporting' sub-process (expanded notation)[1]. An intermediate event ('catch' semantic) has been added to the boundary of that sub-process. The model contains also a collapsed sub-process (details are not visible), denoted by a "plus" sign[4].
- The 'Partner failure' event ('catch' semantic) gets activated when the execution points reaches the 'Reporting' sub-process[2], and gets un-activated when the 'Reporting' sub-process completes successfully (i.e., an end event is reached).



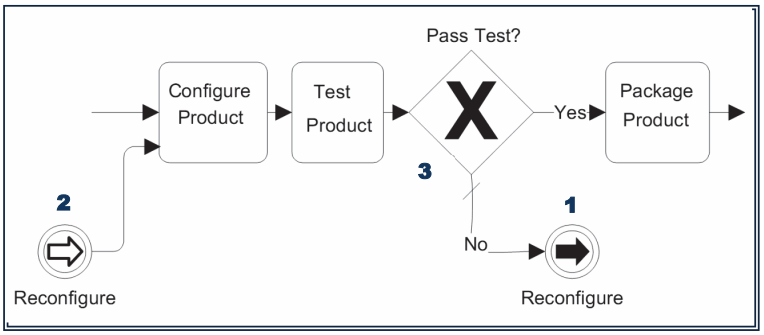
http://www.iet.unipi.it/m.cimino/bpm/res/mov02

- Note how the partner failure event ('send' semantic) raised by the reporting sub-process is caught by the 'Partner failure' event[3] ('receive' semantic) and how the execution flow get diverted from that point.
- Note that if a sub-process of a diagram is expanded in the diagram, the elements inside the sub-process cannot be connected to elements outside the sub-process.
- BPMN allows the *structured modeling* of processes, i.e., views at different levels of abstraction: from the level "0" (the least amount of detail) the processes are decomposed into sub-processes, up to activities (that are atomic, the most amount of detail). For instance, in the IBM methodology the analysis stops at the level "3".

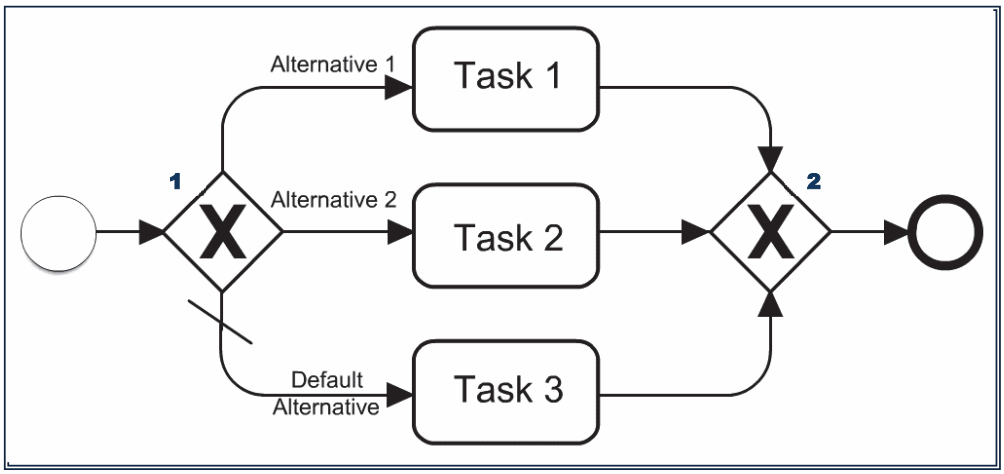


http://www.iet.unipi.it/m.cimino/bpm/res/mov03

- At the level 0 process is a high-level *not executable* process. Level 0 is modeled for the purpose of documenting process behavior at a modeler-defined level of detail. Thus, information needed for execution, such as formal condition expressions, are typically not included. In contrast, an *executable* process is modeled for the purpose of being executed (e.g. a WS-BPEL process).
- BPMN allows process *segmentation*, at a given level, to create different modular segments. For instance, in the IBM methodology it is recommended that the maximum number of process activities per page should be *six*.
- The *off-page connector*, generally used for printing, is an object showing where a Sequence Flow leaves one page and then restarts on the next page. A *Link Intermediate Event* with throw[1] and catch[2] semantic can be used as an Off-Page Connector. In figure, the flow of the exclusive gateway[3] labeled with "No" leads to come back, and then to a cycle.

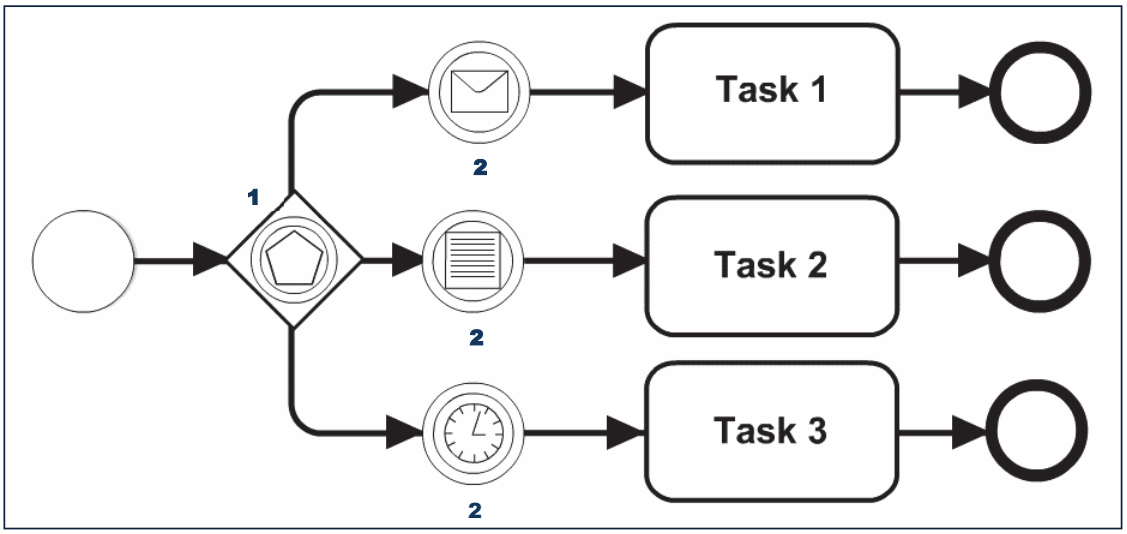


- Gateways are decisions points used to constrain the execution flow, fork an execution point into several or merge several into one. A gateway is represented by a diamond and the kind of a gateway is specified by a marker.
- An *exclusive gateway* can be used as a decision point[1] where several outgoing sequence flows are possible. Such flows are all constrained by a condition allowing only one of them to be used by a token. Such a condition will be evaluated based on the process data. The gateway can be also used as a way to merge[2] several sequence flows into one. The incoming token moves straight through the gateway and goes on.



http://www.iet.unipi.it/m.cimino/bpm/res/mov04

- The *Exclusive Event Based gateway*[1] is similar to the Exclusive Data Based gateway. The only difference is that, instead of evaluating a set of alternatives to determine only one outgoing flow, the event based gateway will start a race between the different events[2] the process might receive, the first one to be received wins the race and that determines which outgoing sequence flow should be used.

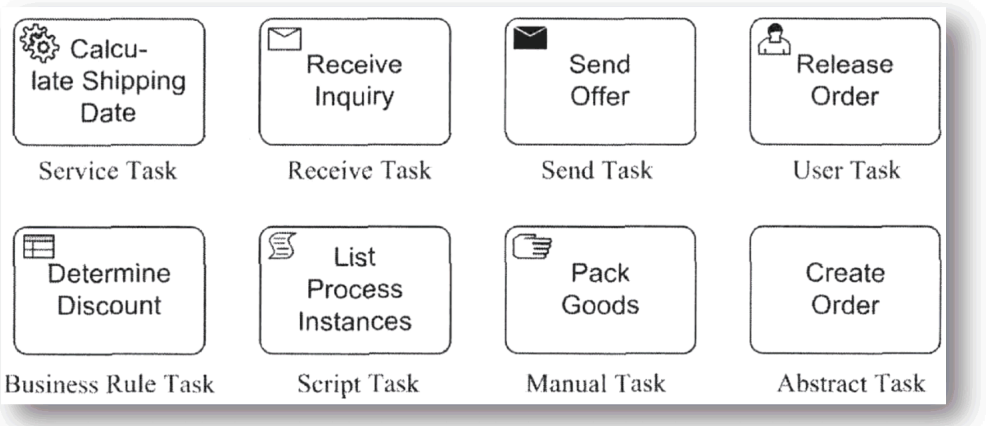


http://www.iet.unipi.it/m.cimino/scom/res/mov05

- Exercise: identify the event before task 2 via the BPMN poster.

Types of tasks and icons

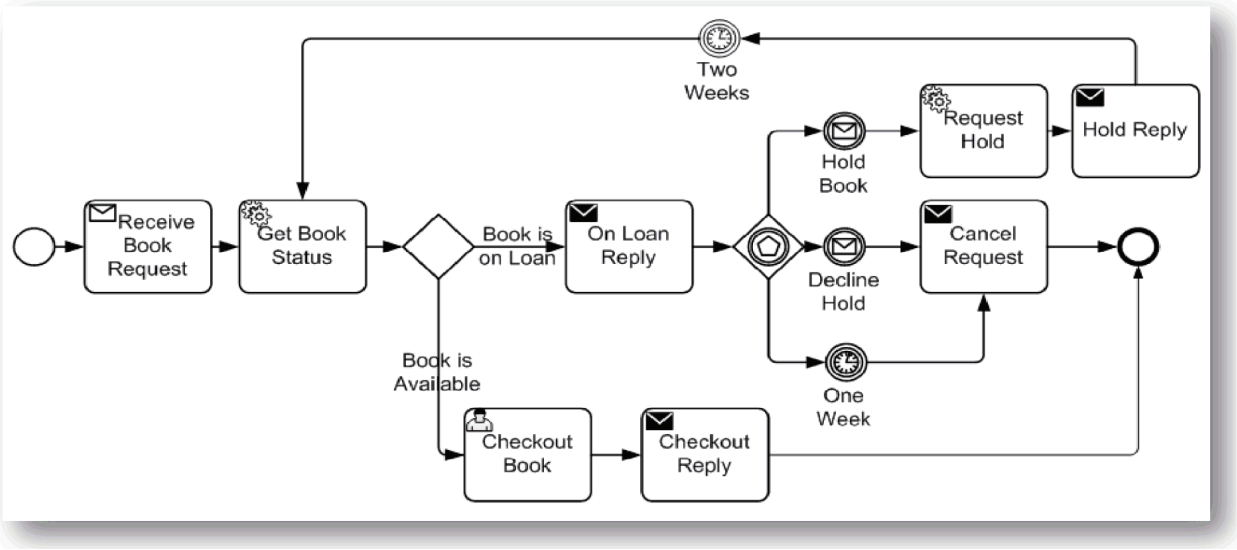
- a *service task* is an automated function processed by an external application;
- a *receive/send* task receives/sends a message;
- a *user task* expects input by a user via a UI;
- in a *business rule task* some business rules are applied (e.g. via a business rule management system launched by the process engine) to produce a result;
- a *script task* contains statements processed directly by the process engine;
- a *manual task* is carried out without IT support;
- in an *abstract task* no type is defined.



Readings:
- Section 7.4, T. Allweyer, *BPMN 2.0*, 2010

Library

- Describe in semi-formal natural language the following orchestration, concerning the book lending process managed by a library.



BP Modeling: semi-formal specification and scenarios

Semi-formal notation

1 The System receives a book request (receive task)
2 The System gets the book status (service task)
3.1 If the book is available:
3.1.1 The librarian checks the book out (user task)
3.1.2 The System sends a checkout-reply message (send task) → **4**

3.2 If the book is on loan:
3.2.1 The System sends an on-loan-reply message (send task)
3.2.2 The system waits for an event:

3.2.2.1 When a decline-hold message is received by the system, or
When a week lapsed without answer:
3.2.2.1.1 The System cancels the request and sends a related message → **4**

3.2.2.2 When a hold-book message is received by the System:
3.2.2.2.1 The System holds the request (service task)
3.2.2.2.2 The System sends a hold-reply message (send task)
3.2.2.2.3 The System waits for two weeks
3.2.2.2.4 → **2**

4 End

Questions

1. When a hold-book message is sent, no other emails are due by the Customer to have the book in hand.
   ❑ true ❑ false

2. When the book is on-loan, more than 5 total messages (sent or received) are due to have the book in hand, provided that it is made available by two weeks.
   ❑ true ❑ false

3. When the book is still on-loan after two weeks since the hold-book message was sent, it is possible to rethink the initial choice by sending a decline-hold message.
   ❑ true ❑ false

4. While the book is on-loan, the Customer can control the system in causing a get-book-status every about three weeks instead of two weeks.
   ❑ true ❑ false

Answers

1. **False**: the hold-book message must be repeated every two or three weeks to keep the book request. If not, the request will be canceled.

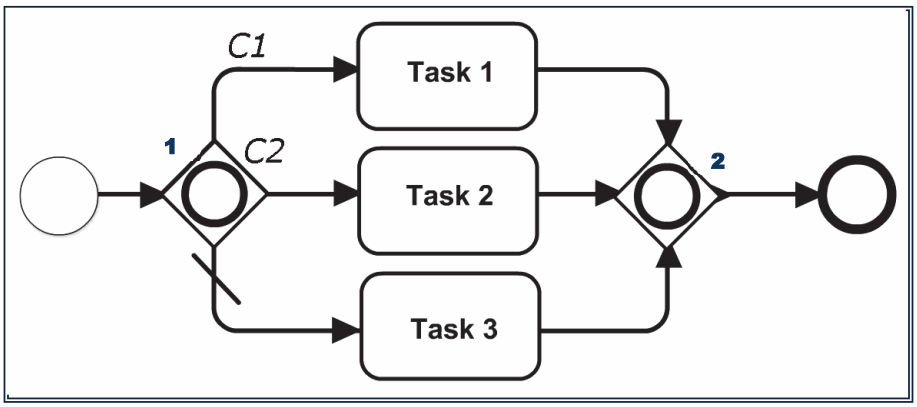
2. **False**: the number of messages is exactly 5. ↓ receive book request, ↑ on-loan reply, ↓ hold book, ↑ hold reply, ↑ checkout reply.

3. **True**: after two week, the System will send another on-loan-reply message, and the Customer can answer with a decline-hold message.

4. **True**: After the on-loan-reply message, the Customer can wait a little less than one week to send the hold-book message. This way, the loop for a get-book-status lasts a little less than three weeks: two weeks + a little less than one week.

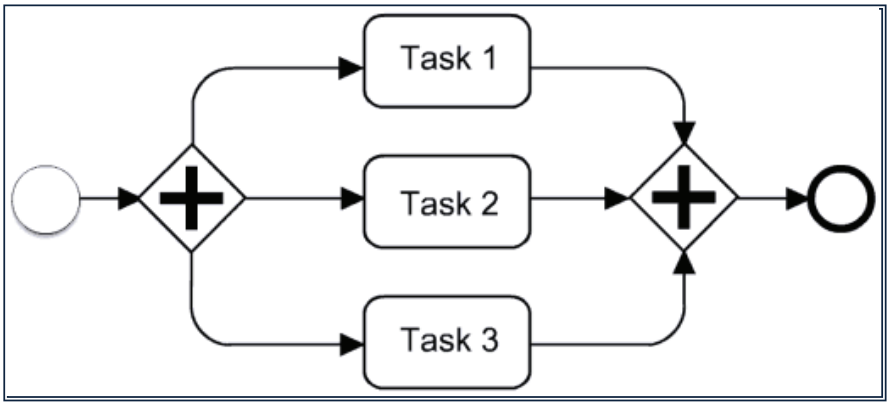
Gateways

- An *Inclusive Gateway*[1] can be used as a decision point where several outgoing sequence flows are possible, they are all constrained by conditions, each outgoing sequence flow with a condition evaluated as being true will be followed. Effectively it might spawn several execution points.
- Used as a merge[2] the Inclusive Gateway will synchronize all the execution points produced upstream but at most one for each incoming Sequence Flow



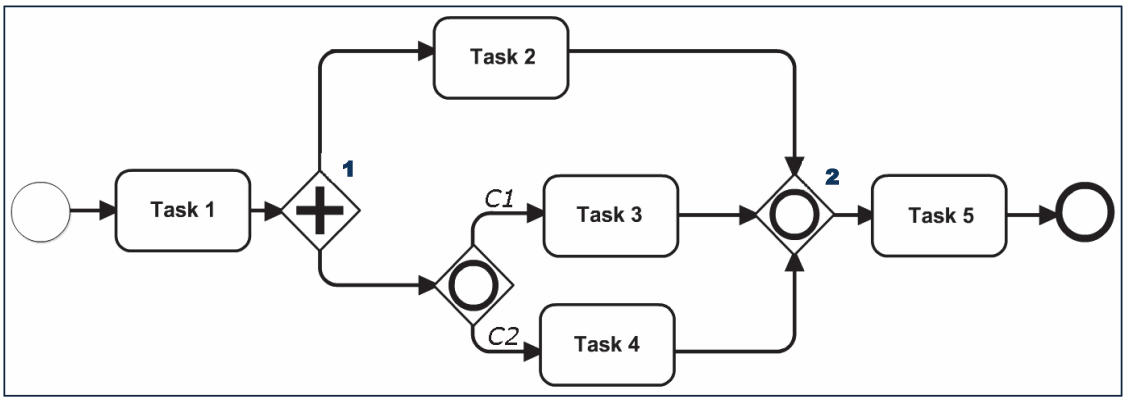
http://www.iet.unipi.it/m.cimino/scom/res/mov06

- A gateway's exit can also be marked with a small diagonal slash as *default* sequence flow. It will be selected automatically, (only) if no condition of the other sequence flows is true. This ensures the actual selection of at least one sequence flow.
- A *Parallel Gateway*[1] provides a mechanism to fork and synchronize flows. There are no conditions associated to this gateway.



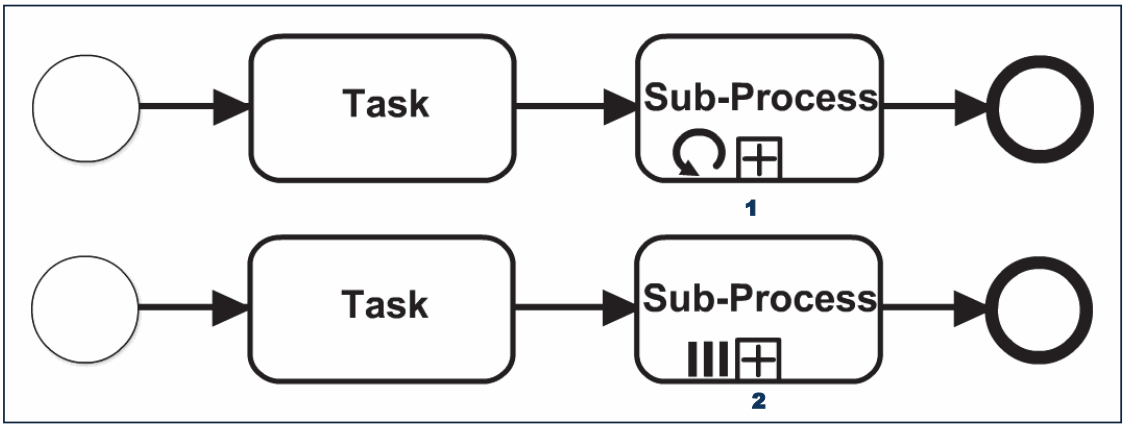
http://www.iet.unipi.it/m.cimino/scom/res/mov07

- In the following model, notice how the first inclusive gateway produces two tokens because both conditions ‘C1’ and ‘C2’ are evaluated as being true. The second inclusive gateway[2] will not only synchronize the token produced by the upstream inclusive gateway, but also the one coming from the upstream parallel gateway
- Exercise: what happens if the second inclusive gateway is replaced by a parallel gateway? Consider a scenario in which *C1* (or *C2*) is false.



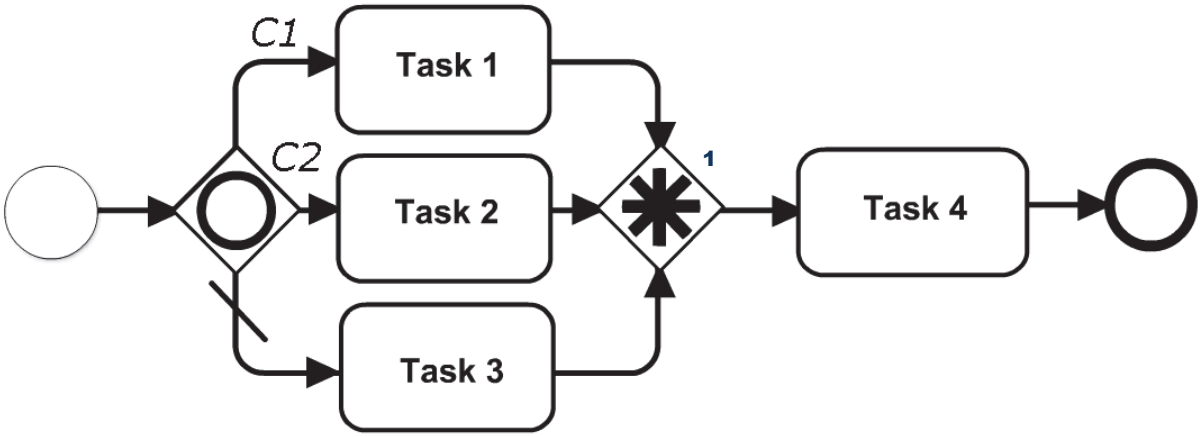
http://www.iet.unipi.it/m.cimino/scom/res/mov08

- A *Loop activity* is an activity repeated sequentially a number of times. It is characterized by a ‘Sequential Loop’ marker[1] on the Task / Sub-Process. As a result, the sub-process will be instantiated several times sequentially. The number of instances to execute might be: (i) defined at design time, (ii) affected at runtime from some process data, (iii) computed at runtime.
- A *multi-instance activity* is an activity repeated in any order, sequentially or in parallel, whose number is defined in advance. It is characterized by a “Parallel Loop” marker[2] on the Sub-Process.



http://www.iet.unipi.it/m.cimino/scom/res/mov09
http://www.iet.unipi.it/m.cimino/scom/res/mov10

- The *Complex Gateway* addresses complex cases which would require the combination of several other gateways. To avoid this, the behavior of the complex gateway can be scripted using an expression language.

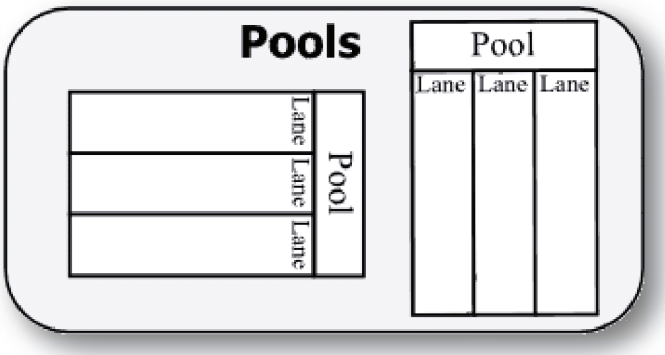


http://www.iet.unipi.it/m.cimino/scom/res/mov11

- As a result the complex gateway can be used to handle every situation. However, a good practice is to avoid it since it makes the process models less readable.

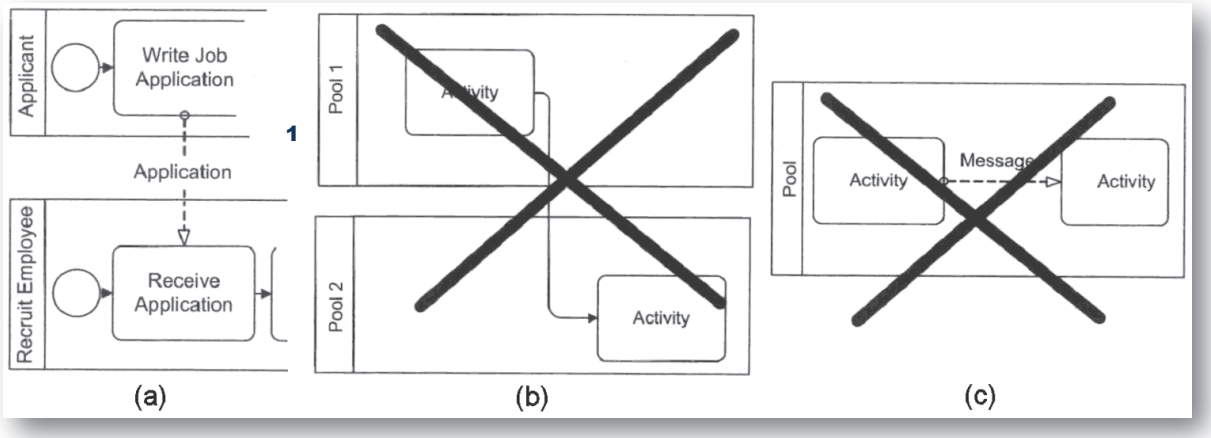
- Modeling with (+) or without (–) gateways?
  + it is not possible to model entirely without gateways
  + gateways are more expressive and, sometimes, necessary
  – if gateways are omitted as far as possible, more compact models can be created
  – gateways with several inputs and several outputs may cause misunderstanding
- A *Pool* represents a participant in a business process. It can be a specific entity (e.g. department) or a role (e.g. assistant manager, doctor, student, vendor). A *Pool* is represented by a box.
- A *Lane* is a sub-partition within a Pool. For instance, having a pool Department, you may have Department Head and General Clerk as lanes. Same as pools, you can use lanes to represent specific entities or roles who are involved in the process.

References:
OMG, *BPMN 2.0 Specification*, pp. 29-41.

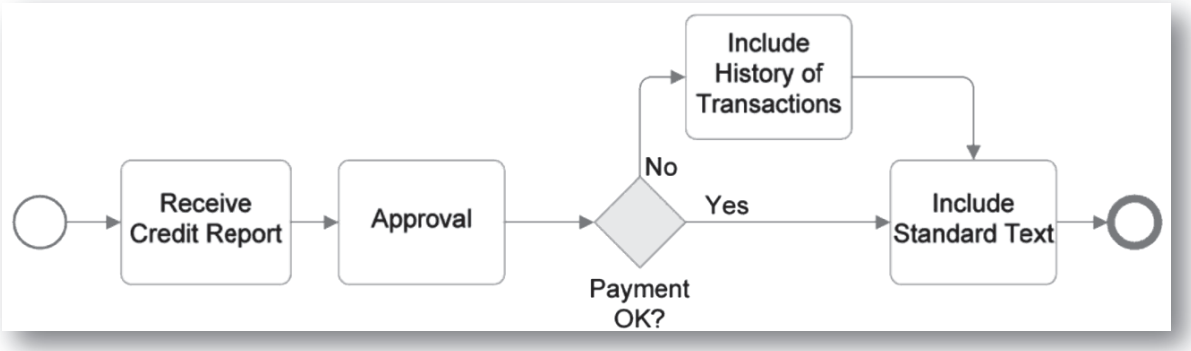


Orchestration and Choreography

- A *private* business process is internal to a pool, i.e., does not interact with external pools.
- A *public* process interacts with other pools via message flows[1], represented as white dashed arrow (a).
- Control flow between pools (b) and message flow within a pool (c) are **forbidden**

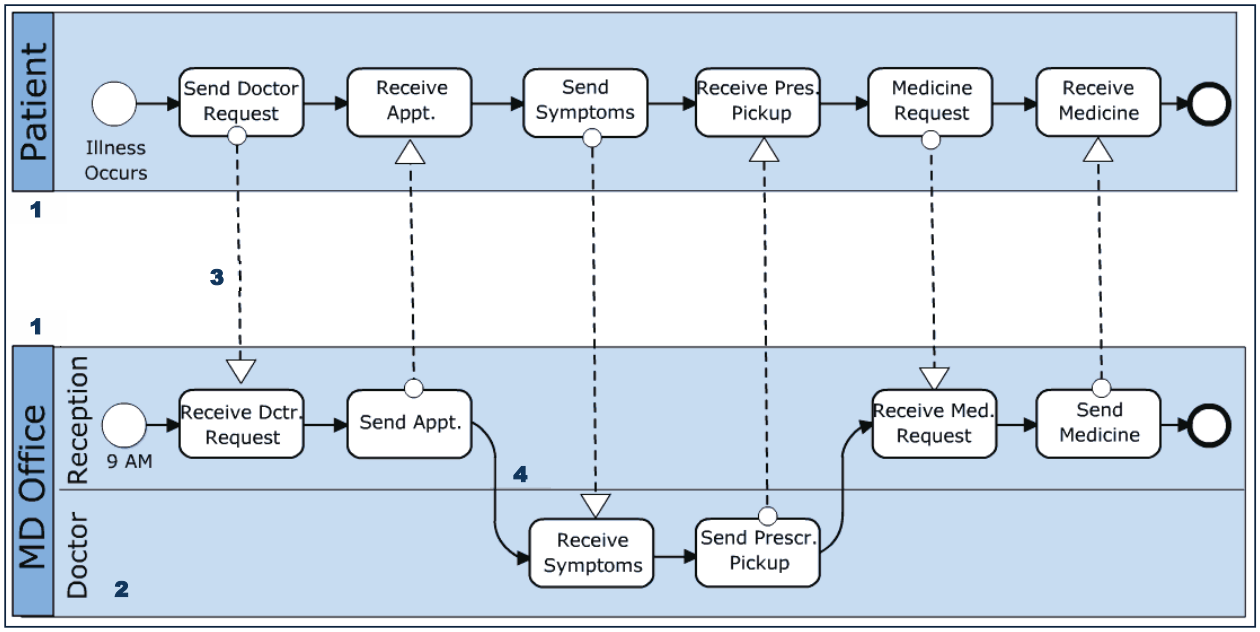


- Considering the centralization/distribution of the control flow, there are two types of BPMN processes: *orchestration* and *collaboration/choreography*.
- *Orchestration*: a process with *central control* of activities (like the control of an orchestra director), modeled as a private process, and containing optionally black-box pools presenting external entities.



- *Collaboration* ("business protocol"): the ordered set of interactions between two or more participants as shown by pools, with their public processes and message flow between them. A collaboration is characterized by *distributed control*. It is also called *choreography*: each individual participant reacts to events generated by the other participants (as in a choreographed dance where dancers react to behaviors of their peers). It can contain optionally some orchestration or just black-box pools.

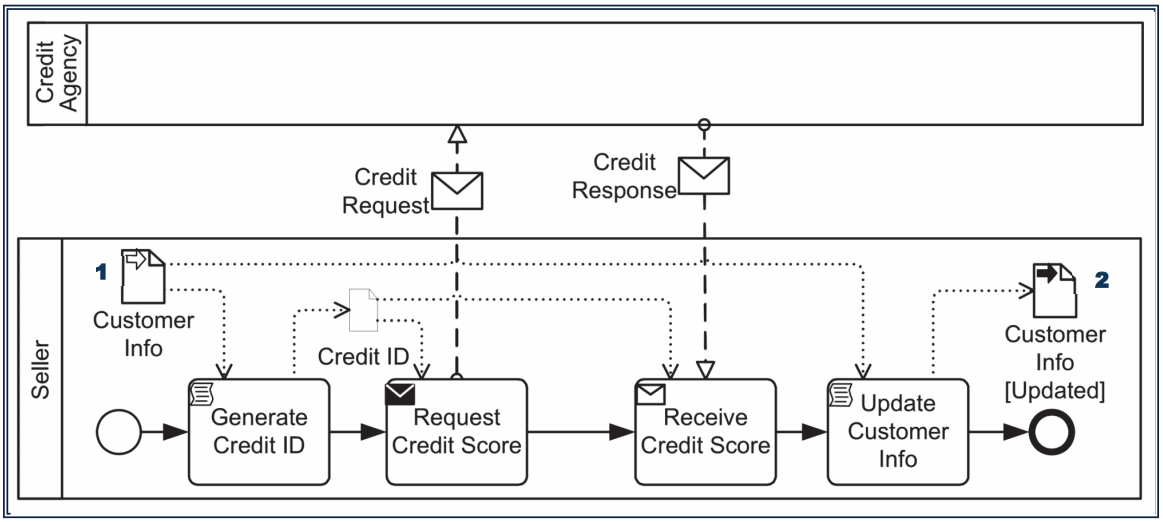
- Collaboration between a patient and a medical office. The processes within pools are abstracted to hide complexity or private information to partners. Note: pool (1), lane (2), message flow (3) and control flow (4). Here, choreography is made by the message flows, whereas orchestration is made by sequence flow, events, tasks.



http://www.iet.unipi.it/m.cimino/scom/res/mov12

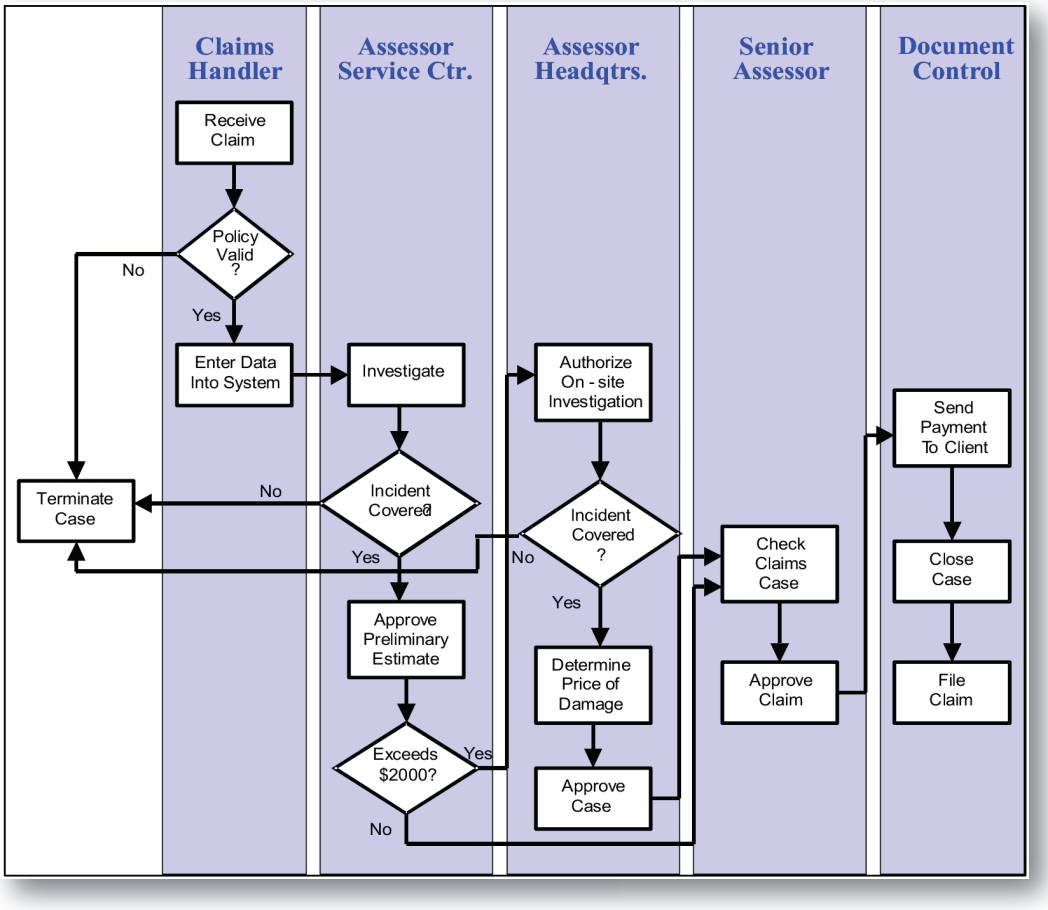
Data objects

- BPMN allows the explicit modeling of data transfer, via *data objects*, *messages* and *data store*. Data objects only exist within a process, whereas data store represent persistent data. UML class diagrams or technical terms diagrams could be used to refine data objects in process models → process view is integrated with data view.
- A directed data association is drawn as a dotted line to model which activity outputs and takes as input a certain data object. Data input[1] and data output[2] can represent a form of dependency between activities.



Modeling from informal diagrams: a claims process at an insurance company

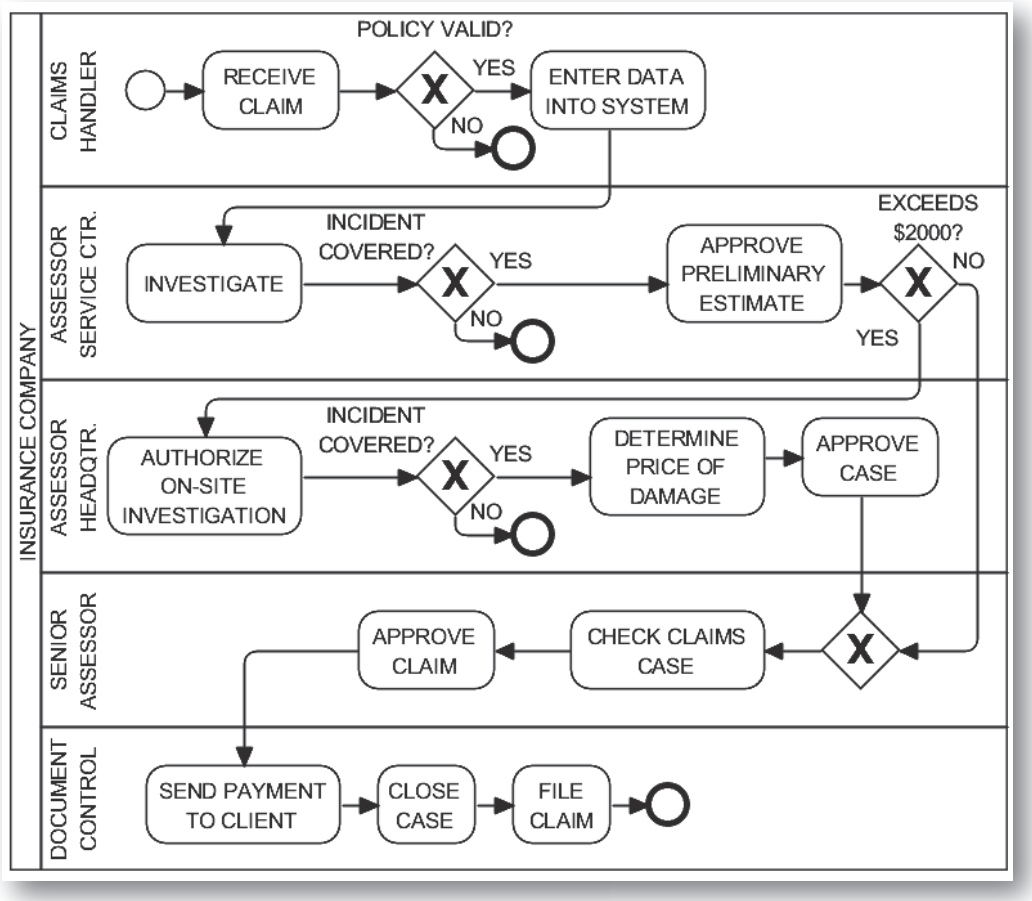
- A personal claims department in an insurance company handles claims made by their clients. The figure on the next page is a non-BPMN process map depicting the personal claims process in terms of swimlanes.
- The first lane corresponds to work done by a claims handler (CH) located at the client's local service center. Upon arrival of a claim, the assessor determines if the client has a valid policy. If no (5% of all cases), then the case is terminated; otherwise (95% of all cases), the assessor enters the appropriate information in the system.
- In the second lane, an assessor located at the service center (ASC) receives data from the claims handler. The assessor first determines if the claim is covered by the client's policy. If not (5% of cases), the case is terminated; otherwise (95% of cases), the assessor approves the preliminary estimate of the damage. If the damage exceeds $2,000 (35% of cases), the claim is sent to an assessor at headquarters for approval; otherwise (65% of cases), it is sent directly to a Senior Assessor.



- Lane 3 corresponds to the assessor at headquarters (AHQ). The assessor first authorizes the on-site investigation of the accident. If the investigation determines that the incident is not covered by the client's policy (2% of cases), then the case is terminated; otherwise (98% of cases), a final price is determined and the case is approved.
- In lane 4, the senior assessor (SA) receives the claim, checks it, completes it, and provides the final approval. Once the claim is approved, it is sent to documentation control.
- Documentation control (DC), in lane 5, is in charge of processing the payment to the client, closing the case, and, finally, filing the claim.

✓ Create a BPMN model

✓ Given 100 starting tokens, determine the number of ending tokens for each scenario (path), considering the aforementioned branching proportion (percentage of cases) for each gateway.

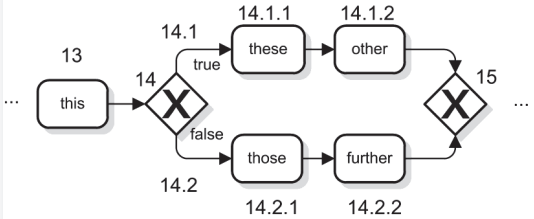


BP Modeling: semi-formal specification and scenarios 2 of 14

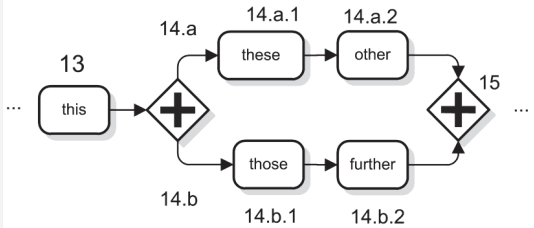
- The semi-formal notation is made of short sentences, numbered so as to allow the specification of the control flows with BPMN 2. It resembles detailed use cases in UML.

MAPPING FROM SEMI-FORMAL TEXTUAL DESCRIPTION TO BPMN MODEL (AND VICE-VERSA)

13. this
14. *these*
15. *that*



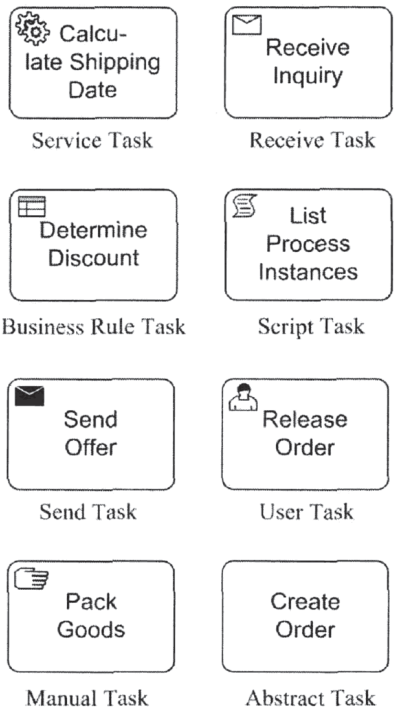
13. *this*
14.1. If it is true ...
14.1.1. *these*
14.1.2. *other* → 15
14.2. If *It is false…*
14.2.1. *those*
14.2.2. *further*
15. ...



13. *this*
14.a.1. *these*
14.a.2. *other* → 15
14.b.1 *those*
14.b.2 *further*
15. Wait for the end of 14.a and 14.b
...

Note: In case of  : 15. Wait for the end of the ongoing activites in 14.a and 14.b

TASK TYPES

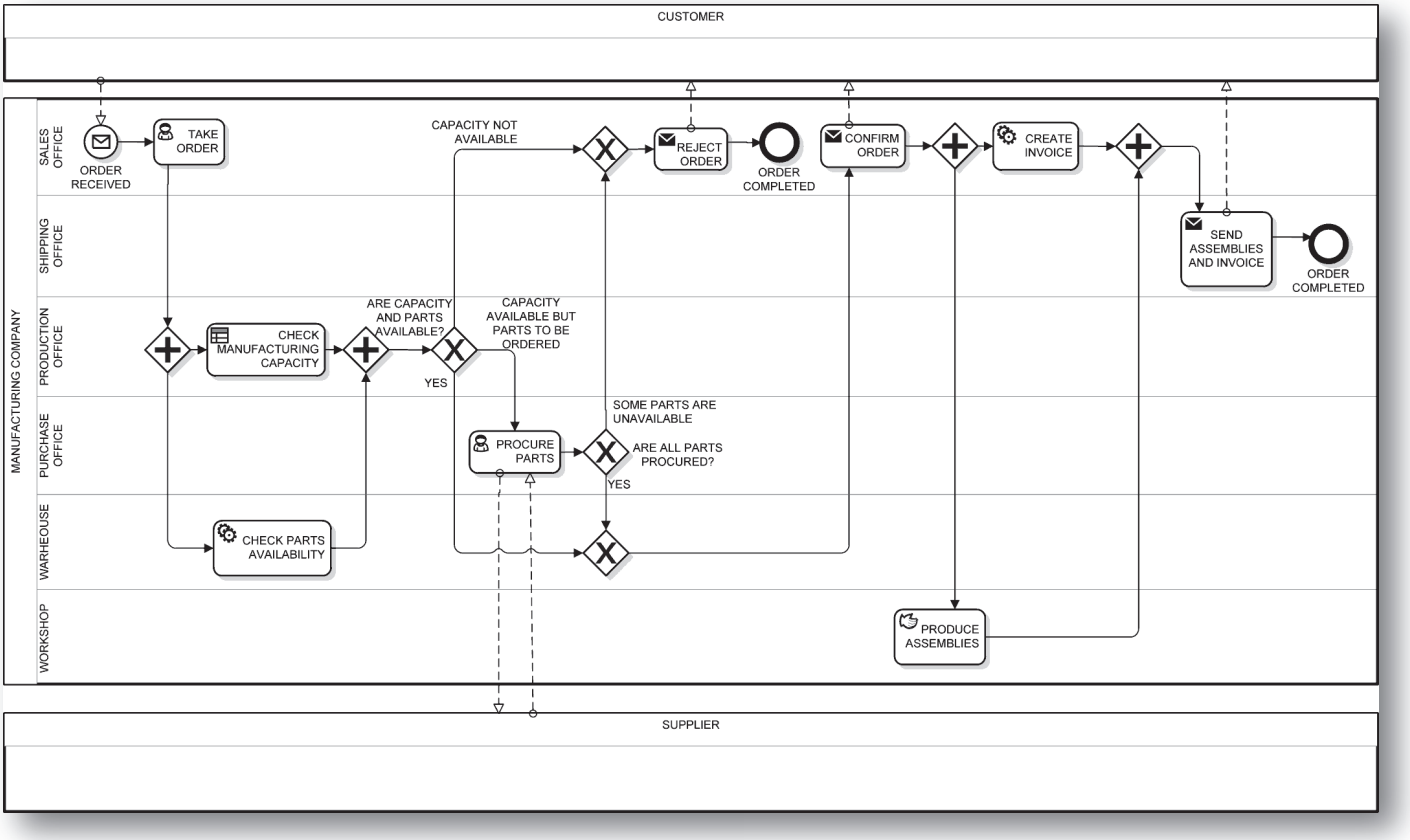


Example: create a handoff diagram

Manage orders in a manufacturing company

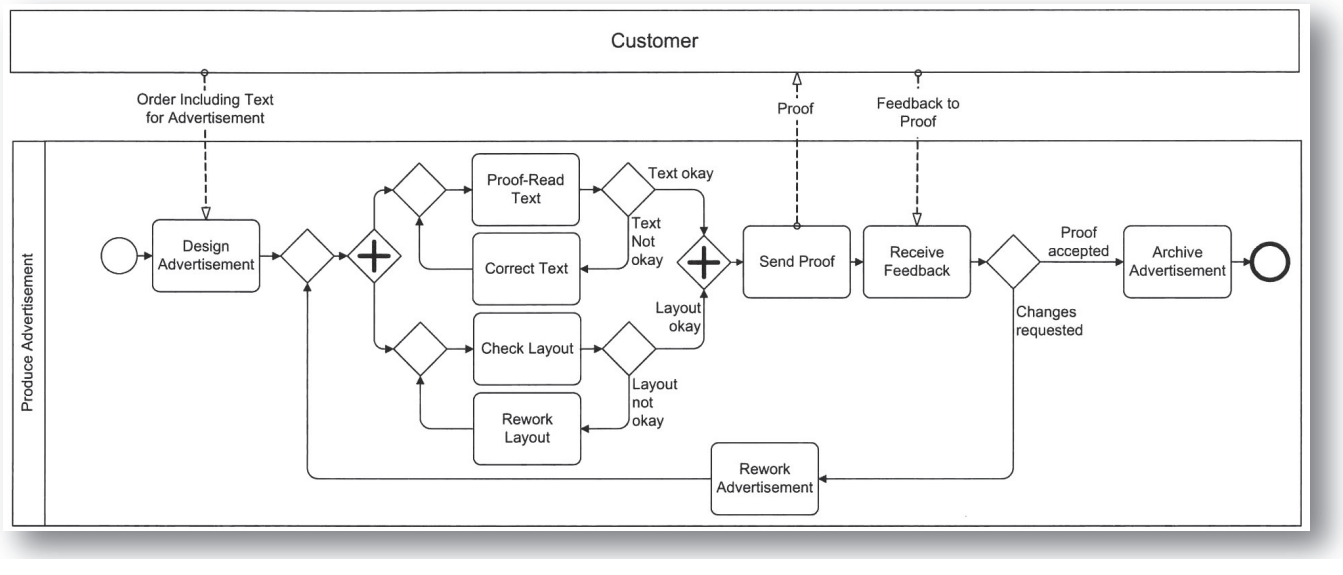
Order processing in a manufacturing company takes place after an order has been received from a **customer**. The order is first taken by the **sales office**. Both manufacturing capacity and parts availability are then checked, via appropriated business rules and an automated information service, by the **production office** and the **warehouse**, respectively. Subsequently, an order rejection is sent to the **customer** by the **sales office** if capacity is not available. Otherwise, if parts are also available, an order confirmation is automatically sent to the **customer** by the **sales office**. Therefore, both assemblies and invoice are produced, by human operatives at the **workshop** and by the **sales office** via an automated information service, respectively. Finally, assemblies and invoice are sent to the **customer** by the **shipping office**. In contrast, if there are parts to be ordered, the **purchase office** procures such parts, purchasing them from **suppliers**, so as to allow the aforementioned process to continue once such parts are available. However, if some parts are still unavailable (they cannot be procured), a rejection is then sent to the customer by the **sales office**.

The BPMN model of the manufacturing company at the handoff level



Homework: advertising agency

- Describe in semi-formal natural language the following process model, describing how an advertising company produces an advertisement, and which messages it exchanges with the customer.
- Given 100 starting tokens, determine the number of ending tokens for each scenario (path), considering the following branching proportions at each gateway: text not okay (10%), Layout not okay (20%), changes requested (1%).
- At each scenario, assume "happy" cases maximizing completd orders.



1. The Company is ready to receive orders from customers
2. A Customer sends an order to the Company including the text for advertisement
3. The Company receives the order from the Customer
4. The Company designs the Advertisement

5.a.1. The Company proofreads the text
5.a.2.1. If the text is okay
5.a.2.1.1. → 6
5.a.2.2. If the text is not okay
5.a.2.2.1. The Company corrects the text
5.a.2.2.2. → 5.a

5.b.1. The Company checks the layout
5.b.2.1. If the layout is okay
5.b.2.1.1. → 6
5.b.2.2. If the layout is not okay
5.b.2.2.1. The Company reworks the layout
5.b.2.2.2. → 5.b

6. The Company waits until both text and layout are okay
7. The Company sends a proof to the Customer
8. The Customer receives the proof from the Company
9. The Customer sends to the Company a feedback on the proof
10. The Company receives the feedback on the proof from the Customer

11.1. If the Customer requests changes
11.1.1. The Company reworks the advertisement
11.1.2. → 5
11.2. If the Customer accepts the proof
11.2.1. The Company archives the advertisement
11.2.2. End

Questions

Suppose that the tasks of proof-read, check, and feedback must provide the same result when applied to the same content (text and/or layout):

1. The Customer can request changes to the layout, and subsequently to the text.
   ❑ true ❑ false

2. Even if the Company reworks the advertisement by applying exactly all changes requested by the Customer, the Customer can subsequently request new changes.
   ❑ true ❑ false

Questions

Suppose that the tasks of proof-read, check, and feedback must provide the same result when applied to the same content (text and/or layout):

1. The Customer can request changes to the layout, and subsequently to the text.
   ❑ true ❑ false

2. Even if the Company reworks the advertisement by applying exactly all changes requested by the Customer, the Customer can subsequently request new changes.
   ❑ true ❑ false

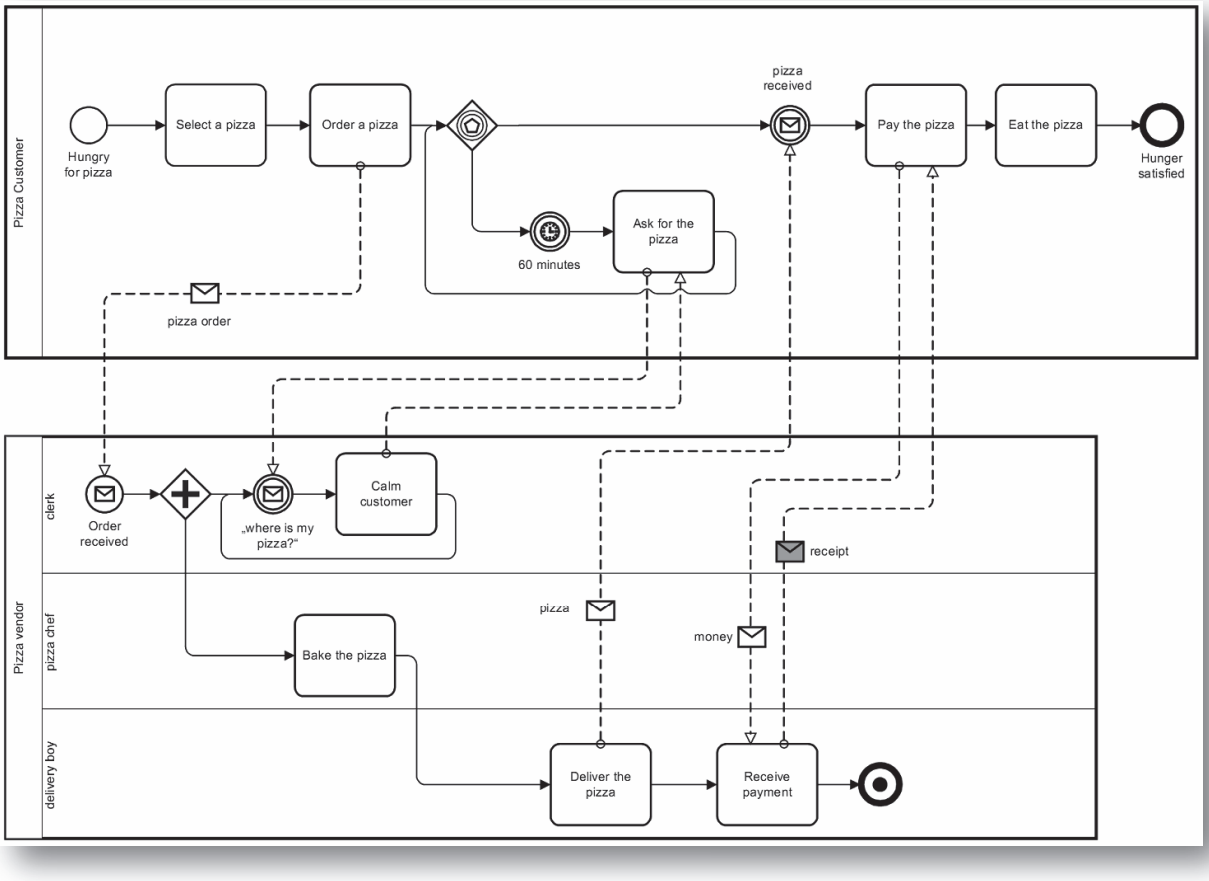
Answers

1. **False**: because the task *rework advertisement* will change the layout only, and then the tasks *proof-read text* and *feedback to proof* will return again *text ok* (as they must provide the same result when applied to the same text).

2. **True:** because the activities of proof-read, check, and feedback can provide a different result when applied to a new content.

Ordering and delivering pizza

❑ The process represents the interaction between a pizza customer and the vendor.



Ordering and delivering pizza

❑ Describe in semi-formal natural language the above BPMN process.

1. The Customer is hungry for pizza
2. The Customer selects a pizza
3. The Customer orders a pizza from the Vendor
4. An order of the Customer has been received by the Clerk

4.a.1. The chef bakes the pizza
4.a.2. The boy delivers the pizza to the Customer
4.a.3. The pizza has been received by the Customer
4.a.4. The Customer pays the pizza to the boy
4.a.5. The boy receives the payment from the Customer
4.a.6. The boy delivers the receipt to the Customer
4.a.7. The Pizza vendor terminates his activities on **4.a** and **4.b** (Vendor End)
4.a.8. The Customer receivers the receipt from the Customer
4.a.9. The Customer eats the pizza
4.a.10. The Customer has been satisfied (Customer End)

4.b.1. When 60 minutes are elapsed:
4.b.1.1. The Customer asks the clerk for the pizza
4.b.1.2. The Clerk has received a request by the Customer for the pizza
4.b.1.3. The Clerk calms the Customer → 4.b.1

Modeling from informal natural language: a Hospital Emergency Center

a. Consider the operation of a Hospital Emergency Center (HEC). The process begins when a patient arrives through the *acceptation* process in the Entrance Room of the HEC, and ends when a patient is either *released* from the HEC or an *arrangement* into the hospital has been established, for further treatment.

b. Patients arriving on their own, after *acceptation* go to an Administrative Room to *sign in*, and subsequently are assessed in the Triage Room in terms of their condition (*triage*).

c. Depending on their condition, patients are classified into different codes (levels): with Red Code (14.8% of all patients), patients are more critical than with Yellow and Green codes.

d. Red Code patients are taken to an Emergency Room. Once in the room, they undergo their *treatment*. Subsequently, they go to an Administrative Room to make the *registration* process and then to be formally *released* or *admitted* into the hospital.

Modeling from informal natural language: a Hospital Emergency Center

e. Yellow and Green Code patients go to an Administrative Room to make the *registration* process and then to an Emergency Room to undergo their *treatment*. Subsequently, they go to an Administrative Room to be formally *released* or *admitted* into the hospital.

f. After admission, the patient must then wait at the Entrance Room for a hospital bed to become available (*arrangement*), so as to be transferred to the hospital room.

g. Patients arriving by ambulance (5.6% of all patients) are directly classified with Red Code after *acceptation* and then immediately taken to an Emergency Room for *treatment*. Subsequently, they are then considered Red code patients.

h. Overall, 90% of all patients (regardless of the assigned code) are released from the HEC, while the remaining 10% are admitted into the hospital for further treatment.

i. In terms of resources, the *treatment* process consists of the following activities: a secondary assessment performed by a **nurse** and a **physician**; laboratory tests, performed by a **patient care technician**; the treatment itself, performed by two **nurses** and a **physician**.

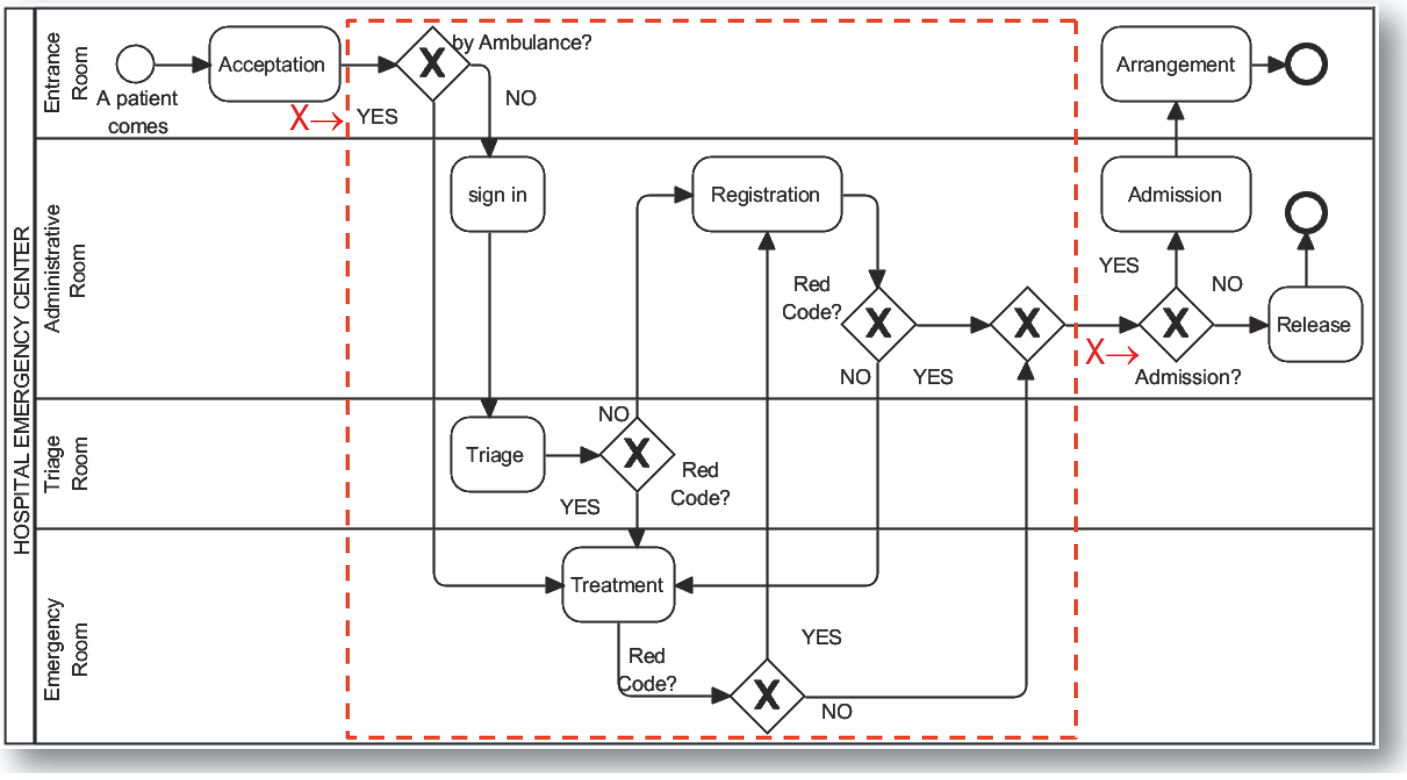
Modeling from informal natural language: a Hospital Emergency Center

i. In terms of resources, the *registration* process consists of: a data collection activity performed by an **administrative clerk**; an additional data collection activity performed by an **administrative clerk**, in case the patient has Worker's Compensation Insurance; a printing of the patient's medical chart for future reference, performed by an **administrative clerk**.

j. The final *release* / *hospital admission* processes consist of: in case of *release*, an **administrative clerk** fills out the release papers; in case of *admission* into the hospital, an **administrative clerk** fills out the patient's admission papers.

- ❑ Create a BPMN model, taking into account only items *a,b,c,d,e,f*, assigning a lane for each underlined term, a task for each italicized term. Do not consider resources (represented in boldface style).
- ❑ Given 100 starting tokens, determine the number of ending tokens for each scenario (path), considering the aforementioned branching proportion (percentage of cases) for each gateway: Ambulance 5.6%, Red code 14.8%, Released 90%

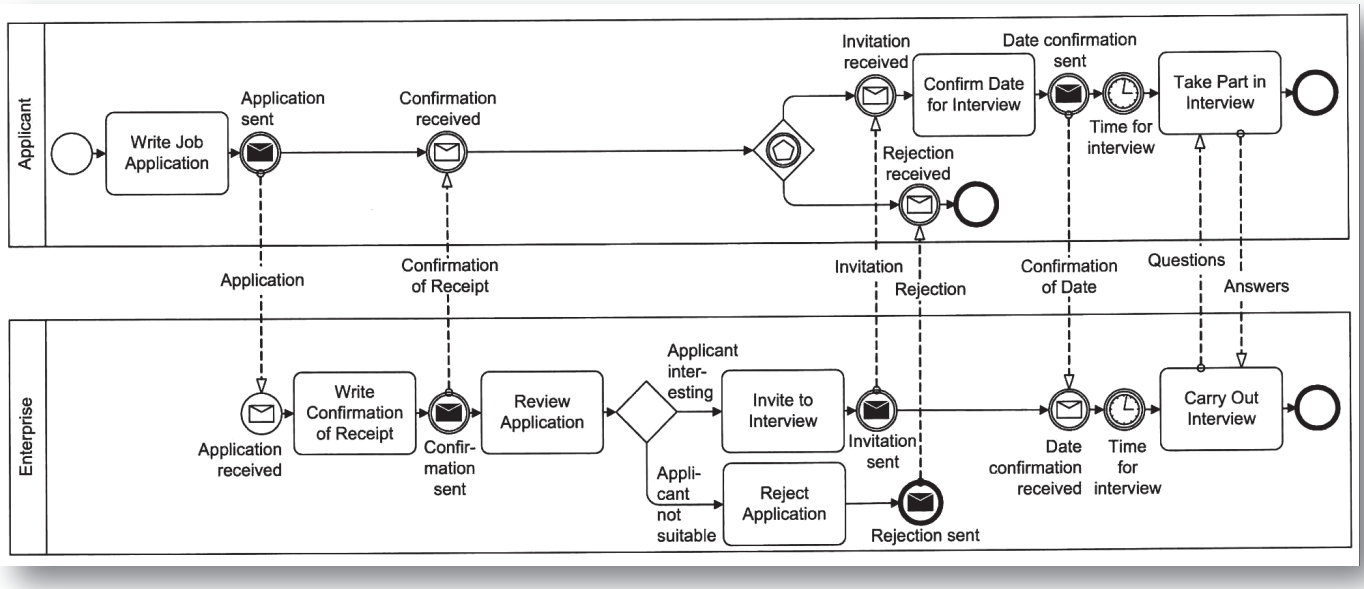
- ❑ Multiple input flows at the tasks *treatment* and *registration* should be avoided by using a XOR-join (not inserted to avoid a high number of gateways).
- ❑ The number X of incoming patients is the same at the exit of the *acceptation* task and at the entry of the "Admission?" gateway, because there is a 1-entry-1-exit block.
- ❑ Relative percentages at each gateway are calculated as shown in the next slide.



Job Application

- Describe in semi-formal natural language the following event-driven interaction between a job applicant and an enterprise.

- Given 100 starting tokens, determine the number of ending tokens for each scenario (path), considering the following branching proportions at each gateway: applicant interesting (10%).



Semi-formal notation

1. An Applicant begins (to search for a job)
2. The Applicant writes a job application
3. The application has been sent by the App to the Enterprise
   (The Applicant waits for receiving a confirmation of receipt)
4. The application has been received by the Enterprise from the Applicant
5. The Enterprise writes the confirmation of receipt
6. The confirmation of receipt has been sent by the Enterprise to the Applicant
7. The confirmation of receipt has been received by the Applicant from the Enterprise
   (The Applicant waits for receiving either an invitation or a rejection)
8. The Enterprise reviews the application

9.1. If the Applicant is not suitable:
9.1.1. The Enterprise rejects the application
9.1.2. A message of rejection has been sent by the Enterprise to the Applicant
9.1.3. The Enterprise ends.
9.1.3. The rejection message of the Enterprise has been received by the Applicant
9.1.4. The Applicant ends.

9.2. If the Applicant is interesting:
9.2.1. The Enterprise Invites the Applicant to interview
9.2.2. The invitation to interview has been sent by the Enterprise to the Applicant
(The Enterprise waits for a confirmation of date)
…

Semi-formal notation

…

9.2.3. The invitation to interview has been received by the Applicant from the Enterprise
9.2.4. The Applicant confirms the date for interview
9.2.5. The confirmation of date has been sent to the Enterprise by the Applicant
9.2.6. The confirmation of date has been received by the Enterprise from the Applicant
9.2.7. Time needed for starting the interview has elapsed
9.2.8. The Applicant takes part in interview, receiving questions and providing answers.
The Enterprise carries out the interview, asking questions and receiving answers.
9.2.9. The Process is finished for the Applicant.
The Process is finished for the Enterprise.