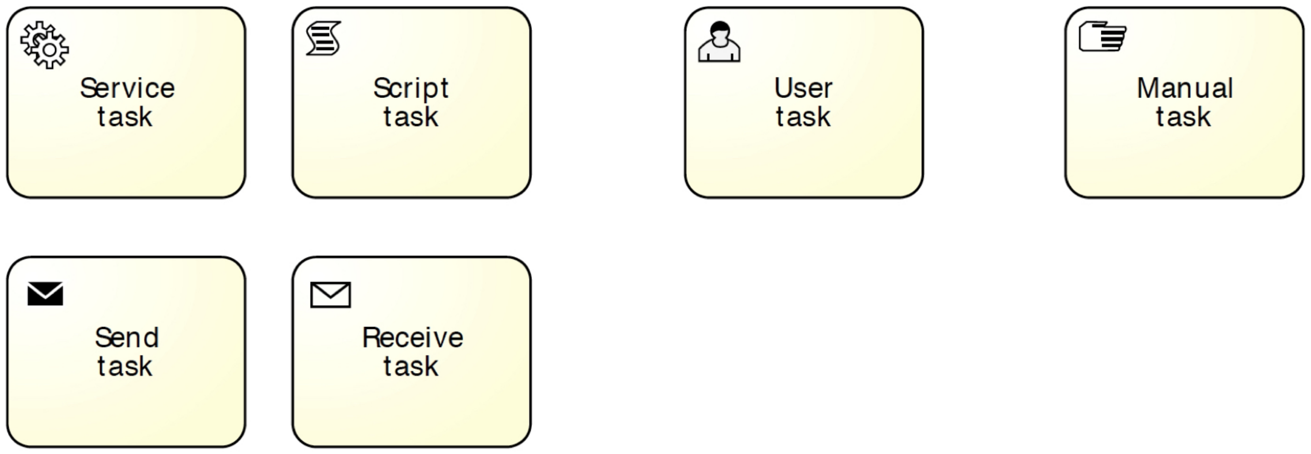
Chapter 10 - Process Implementation with Executable Models

- Five step method to incrementally transform a conceptual process into an executable one, using the BPMN language.

Identify the Automation Boundaries

- **Principle**: not all processes can be automated.
- 1st step: Identify which parts of the process can be coordinated by the BPMS and which parts cannot.
    - Must distinguish between 3 types of tasks:
        - **Automated**: performed by the BPMS or an external service.
        - **Manual**: performed by process participants without any software.
        - **User**: performed by a participant with the assistance of the worklist handler of the BPMS or external task list manager (mix of automated and manual).
    - **Automated and user tasks can be coordinated by a BPMS, but manual tasks cannot**.
- 2nd step: Review the manual tasks and find out if we can hook them up to the BPMS. If we can't, consider if it is convenient to automate the rest of the process without these manual tasks.
    - Ways to hook them:
        - Isolate the task and focus on the automation of the process before and after it.
        - Find a way for the BPMS to be notified when the manual task has started or completed.

All these types of tasks can be modeled in BPMN



Automated tasks

User task

Manual task

•Automated tasks are subtyped in BPMN:

- **Script** (script marker), if the task executes some code (the script) internally to the BPMS. This task can be used when the functionality is simple and does not require access to an external application ex: "Open file" "Select best quote"
- **Service** (wheels marker), if the task is executed by an external application, which exposes its functionality via a service interface ex: "Check stock availability"
- **Send** (filled envelope marker), if the task sends a message to an external service ex: "Request Raw Materials"
- **Receive** (empty envelope marker), if the task waits for a message from an external service ex: "Get shipping address"

Business Rule

Is a specific type of Script Task.

It is also a confusing concept because, as a good practice, Business Rules are related with gateways, not to tasks!

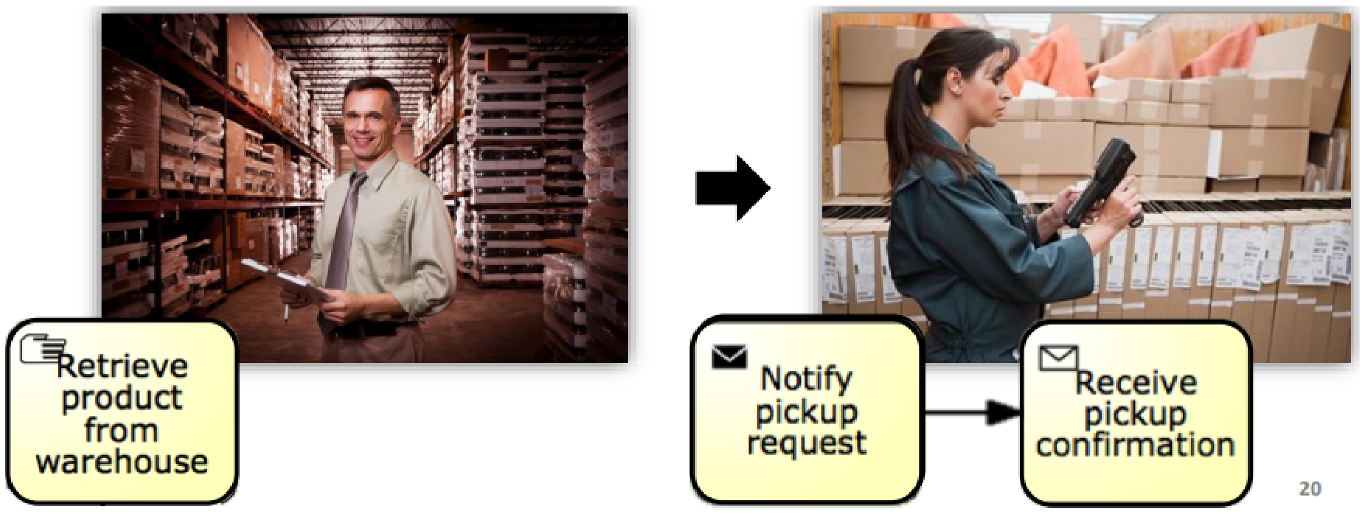
At the end wit address Business Rules

Ex: rule for approving a loan

- These symbols cannot be used in sub-processes.

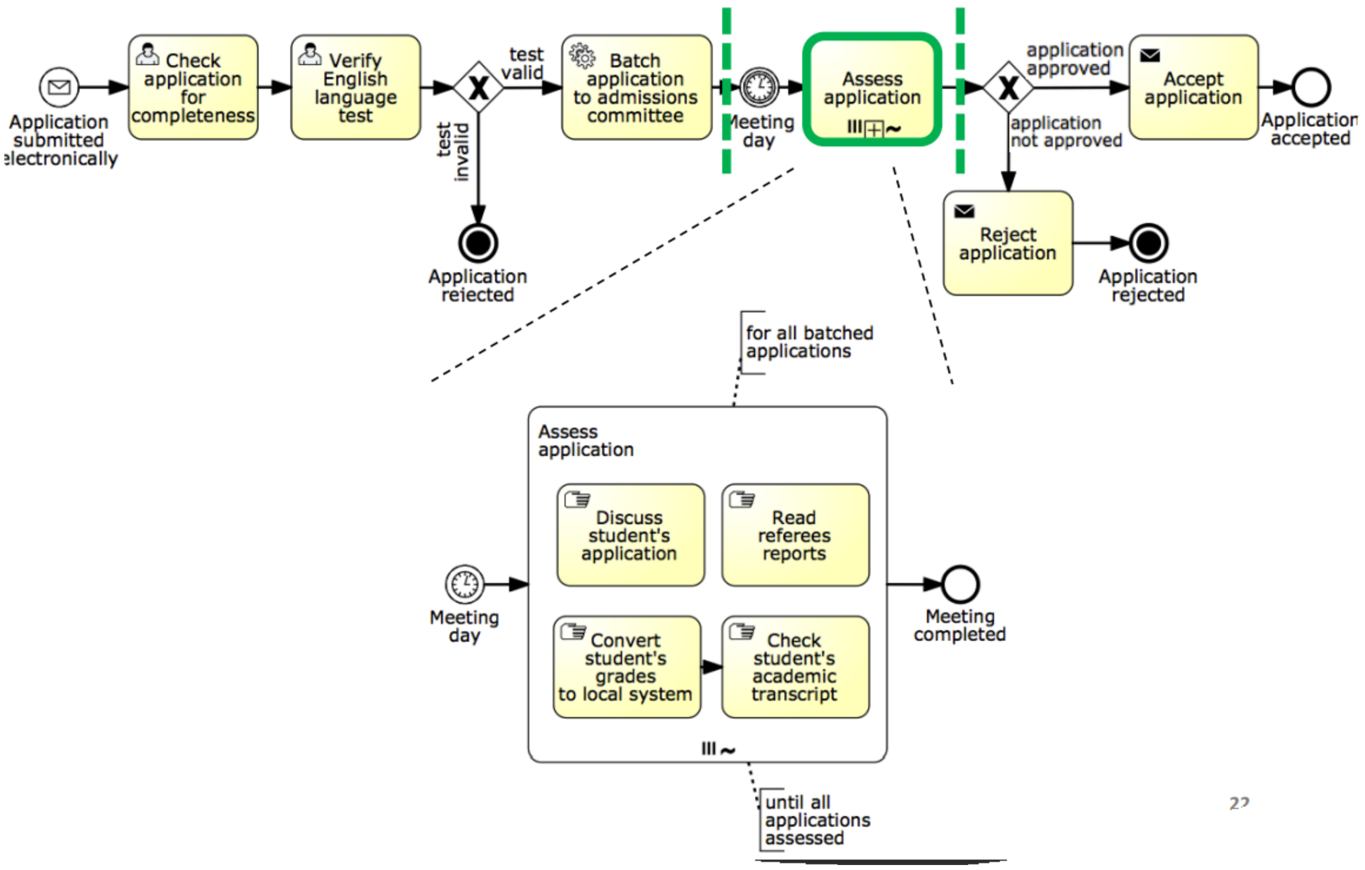
Review Manual Tasks

- **Principle**: If the task cannot be seen by the BPMS, it does not exist. We either find a way to support manual tasks via technology or we isolate these tasks and automate the rest of the process.
    - **Implement as User Task**: if the manual task person can notify the BPMS of their task completion via the worklist handler, then turn it into a user task.
    - **Implement as Automated Task**: if the manual task person can use technology of the BPMS to notify the engine of a work item completion, then turn it into a automated task.



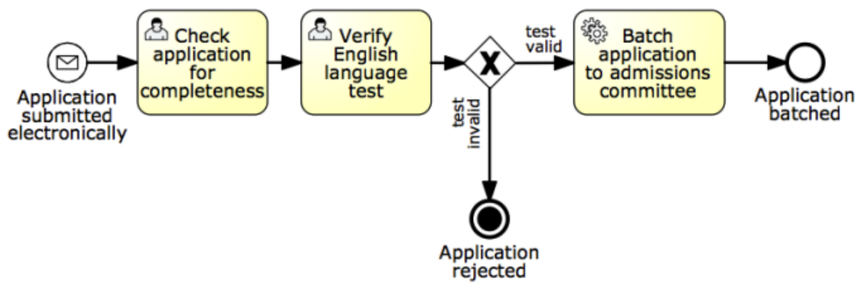
- If not possible, isolate them as automate the rest.

Alternative: isolate manual tasks

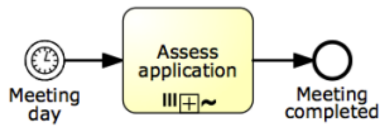

Application submitted electronically
Check application for completeness
Verify English language test
test valid
test invalid
Application rejected
Batch application to admissions committee
Meeting day
Assess application
application approved
application not approved
Accept application
Application accepted
Reject application
Application rejected
for all batched applications
Assess application
Discuss student's application
Read referees reports
Convert student's grades to local system
Check student's academic transcript
Meeting day
Meeting completed
until all applications assessed

22

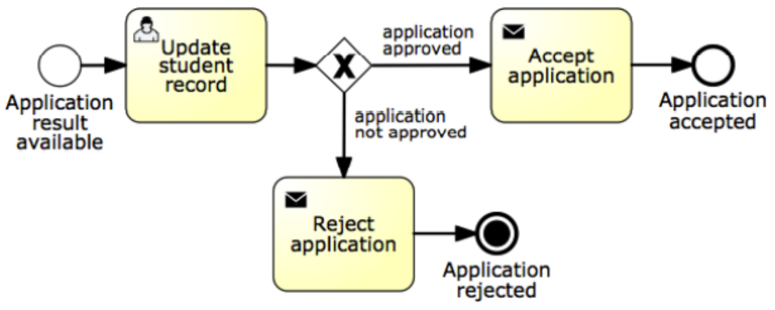
Segment 1


Application submitted electronically
Check application for completeness
Verify English language test
test valid
test invalid
Batch application to admissions committee
Application batched
Application rejected

Segment 2



Segment 3



23

Complete the Process Model

- **Principle 1**: exceptions are the rule.
  - Consider the incomplete paths (not just the sunny day scenario).
  - Rules of Thumb:
    - If we send something to another party, what happens if they do not respond? What happens if the response comes late? What happens if they do not respond the way we expect?
    - For each task: Can it go wrong and what happens if it goes wrong?
    - For each external party: Have we captured all messages or queries they might send us? (use CRUD)
- **Principle 2**: no data implies no decisions and no task handover.
  - Specify all electronic data objects
  - For each task determine which business objects it creates, reads, updates and deletes (CRUD).
  - For each decision, determine which objects it needs.

Adjust Task Granularity

- **Principle**: BPMSs add value if they coordinate handovers of work between resources.
  - **Aggregation** of two consecutive tasks assigned to the same performer/resource. (to become more coarse-grained and less too fine grained)
  - **Decomposition** of two tasks if it requires more than one performer/resource. (to become more fine grained and less too coarse-grained)
- We don't apply these to ad hoc sub-processes, for that is done with CMMN, a complementary language to BPMN. It is a more flexible language than BPMN.

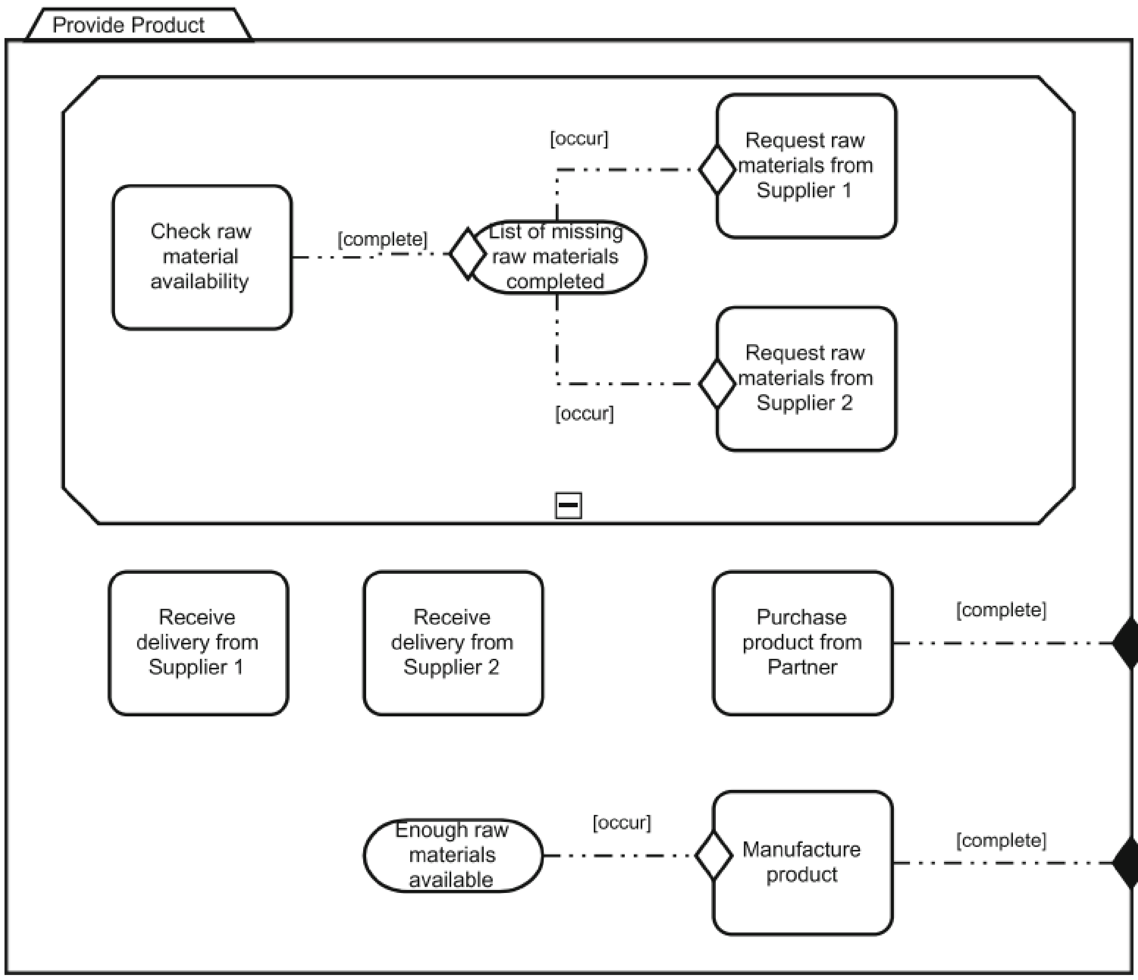

Provide Product
Check raw material availability
[complete]
List of missing raw materials completed
[occur]
Request raw materials from Supplier 1
[occur]
Request raw materials from Supplier 2
Receive delivery from Supplier 1
Receive delivery from Supplier 2
Purchase product from Partner
[complete]
Enough raw materials available
[occur]
Manufacture product
[complete]

Fig. 10.5 Excerpt of an order-to-cash process model (from out-of-stock product to product provided) captured in CMMN

- At the end of this step we obtain a **to-be executed** process model. However this model is still technology agnostic.

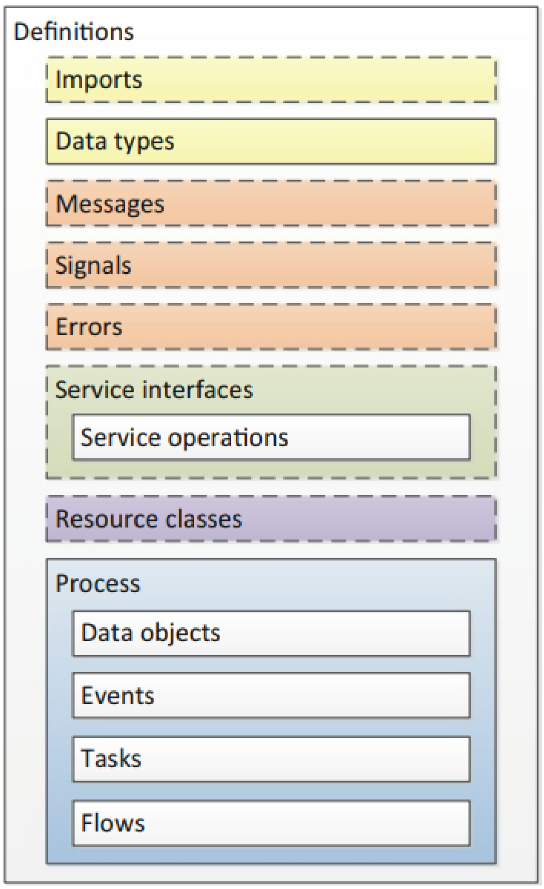
Specify Execution Properties

- **Principle**: Must specify how each model element is effectively implemented by our BPMS of choice.

- Process variables, messages, signals, errors
- Task and event variables and their mappings to process variables
- Service details
- Code snippets
- Participant assignment rules and user interface structure
- Task, event and sequence flow expressions
- BPMS-specific: work queues, forms, connectors…

- BPMN allows for definition of all these, some are mandatory (straight borders) and some are optional (dashed borders).

Fig. 10.6 Structure of the BPMN format



Process Variables

- Allow data exchange between process elements.
- Must assign a **data type** to it: can be **simple** or **complex**.

    - Simple is the types we know like ints, strings, doubles etc...
    - Complex can be imported from other documents. Ex: a purchase order or an invoice that can have multiple simple attributes.
- Ex: The object "Stock availability" can be a integer (units of a product).
- Messages also must have a type. To assign one, simply look at the message flow and define the type. Ex: if the message is labeled "purchase order" it is of the type “purchaseOrderType”.
- For signals, the data type describes the content of the signal being broadcasted.
- For errors, the data type defines the info that is being carried with the error, must also assign a **error code** to each error (this is to connect the throwing and catching of errors).
- Must also identify **internal variables** that are data inputs (capture data required for the task to be executed) and outputs (capture data required for the task to be completed) in BPMN, each one must match the type of the input/output object.
- Catching events only have data output, to store the content of the event being caught. Throwing events only have data input, to store content of the event being thrown (both match data types with what they receive).

Data Mappings

- Must define mappings between data objects and task data inputs/outputs via data associations.

Service tasks

- Must specify how to communicate with the external application that will execute the task.
- All the BPMS needs to know is the service interface the service task can use. The interface contains one or more service operations, each describing a way of interacting with the service.
    - **In-out/Synchronous operations**: the service expects a request message and replies with an answer once the operation has been completed or there was an error. Ex: "Check Availability"
        - Drawback: stops the process until an answer is received.
    - **In-only/Asynchronous operations**: the service expects a request message but will not reply with an answer. Ex: "Archive Order".

Send and Receive Tasks, Message and Signal Events

- Receive tasks can also receive responses of an asynchronous service.

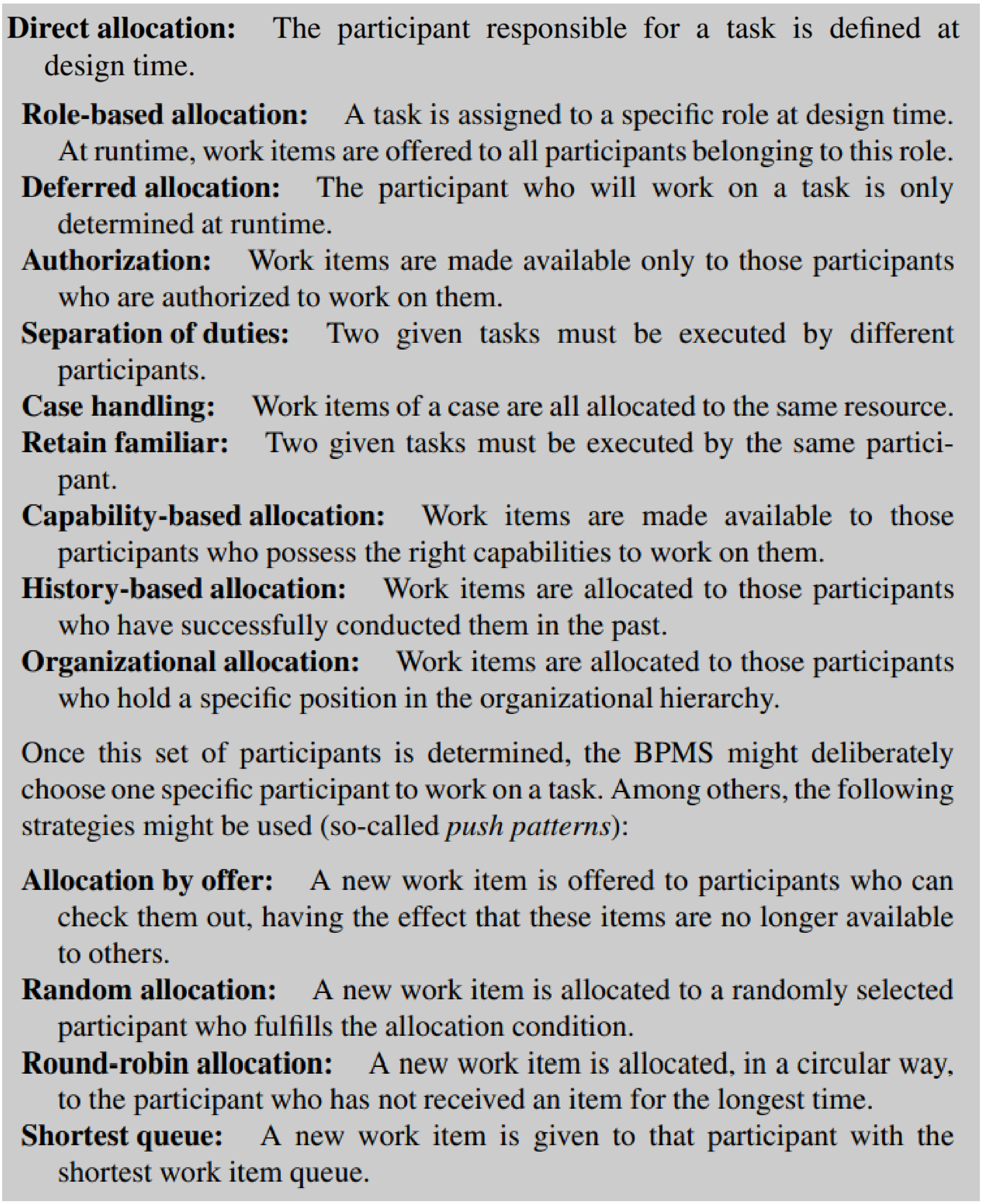
Script Tasks

- Must provide a snippet of code the BPMS will execute.
- The BPMS does not use a specific language, use the one you think is best.
- The task's data inputs stores the parameters of invoking the script and the data outputs store the results of script's execution.

User Tasks

- Must specify rules for allocating work items to process participants at runtime, the technology for communication with other process participants and details of the user interface to use.
- Must define data inputs and outputs (passing information to participant and receiving the results).
- **Potential owners**: process participants that share characteristics like belonging to the same department.

Task Allocation Strategies



Direct allocation: The participant responsible for a task is defined at design time.

Role-based allocation: A task is assigned to a specific role at design time. At runtime, work items are offered to all participants belonging to this role.

Deferred allocation: The participant who will work on a task is only determined at runtime.

Authorization: Work items are made available only to those participants who are authorized to work on them.

Separation of duties: Two given tasks must be executed by different participants.

Case handling: Work items of a case are all allocated to the same resource.

Retain familiar: Two given tasks must be executed by the same participant.

Capability-based allocation: Work items are made available to those participants who possess the right capabilities to work on them.

History-based allocation: Work items are allocated to those participants who have successfully conducted them in the past.

Organizational allocation: Work items are allocated to those participants who hold a specific position in the organizational hierarchy.

Once this set of participants is determined, the BPMS might deliberately choose one specific participant to work on a task. Among others, the following strategies might be used (so-called *push patterns*):

Allocation by offer: A new work item is offered to participants who can check them out, having the effect that these items are no longer available to others.

Random allocation: A new work item is allocated to a randomly selected participant who fulfills the allocation condition.

Round-robin allocation: A new work item is allocated, in a circular way, to the participant who has not received an item for the longest time.

Shortest queue: A new work item is given to that participant with the shortest work item queue.

Task, Event, and Sequence Flow Expressions

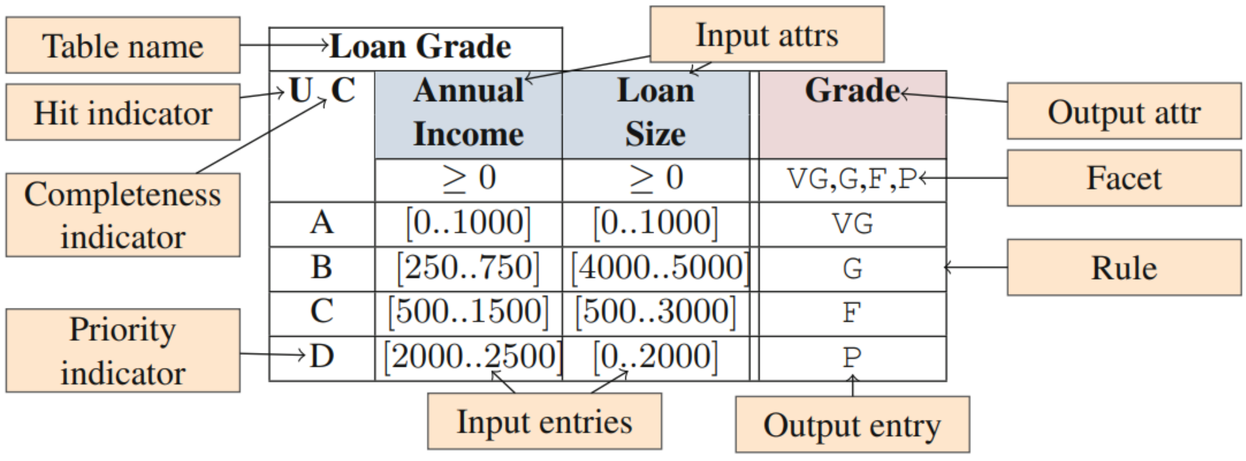
- Must write expressions for the attributes of tasks and events like booleans that indicate a loop condition (ex: "until response approved").
- Expressions written with XPATH.

Implementing Rules with DMN

- Sometimes, the conditions that allow a process instance to be routed towards one or another path in the model can be quite complex.
- OMG has developed the Decision Model and Notation (DMN) standard, which can be used for specifying *business rules*

DMN provides three parts for the specification of business rules:

1. the Decision Requirements Graph (DRG), which describes how data is propagated between different decisions,
2. the Simple Expression Language (S-FEEL) to define how values are extracted from variables, and
3. Decision Tables (DMN tables)



Loan Grade

U,C	Annual Income	Loan Size	Grade
	≥ 0	≥ 0	VG,G,F,P
A	[0..1000]	[0..1000]	VG
B	[250..750]	[4000..5000]	G
C	[500..1500]	[500..3000]	F
D	[2000..2500]	[0..2000]	P

Fig. 10.8 Example of a decision table for loan applications

The Last Mile

- Now we can distinguish 3 types of BPMSs:
  - **Pure BPMN**: systems designed to support BMPN from the first moment they were designed. Ex: Camunda.
  - **Adapted BPMN**: use a BPMN skin but rely on a specific internal representation to execute the process model. Ex: Bizagi.
  - **Non BPMN**: BPMSs that use their own language and semantics. Ex: YAWL.