Event based Gateways: Use of intermediate events

- Event based Gateway
- only one of the alternatives is chosen
- Acceptance
- Rejection
- 3 days
Event based Gateways: Use of Receive tasks

Event based Gateway

only one of the alternatives is chosen

Receive acceptance

receive rejection

receive extension req.
Sub-Conversations

- sub-conversation
  - a higher level conversation which has an expansion (compound conversation)

- sub-conversation expanded
  - a lower level conversation (can be further expanded)
  - message flows

Diagram:

1. Retailer
2. Supplier
3. Delivery negotiations
4. Checkpoint request
5. Confirmation
6. Updated schedule
7. Discuss variations
Call Conversations

- call conversation
  - call global conversation
    - reusable
    - atomic
  - call collaboration
    - expanded

[Diagram showing interactions between customer and supplier, with nodes for ask for quotation, quotation, ask for discounts, discounts, and order.]
More Markers

- Service

- Event subprocesses

- Event subprocess expanded
Event subprocesses

- **Interrupting subprocess**
  - parent process is interrupted
  - an interrupting start event is used
  - boundary of the event is solid

- **Non-interrupting subprocesses**
  - parent process continues after the completion of the subprocess
  - a non-interrupting start event is used
  - boundary of the event is dashed
Use of various types of start events for event sub-processes

- Interrupting timer start event

- Non-interrupting timer start event
Use of various types of start events for event sub-processes

- Interrupting timer start event

- Non-interrupting timer start event
Events

- Something that happens during flow
- e.g. start of an activity, end of an activity, a message that arrives, change in data state
- Event driven processes can be described
- **Start events** indicate where a process will start
- **End events** indicate where a process will end
- **Intermediate events** indicate something happening during a process execution
Events

- Catching Events
  - Some events catch a trigger
  - all start events, some intermediate events

- Throwing Events
  - Some events throw a result
  - all end events, some intermediate events throw a result
  - a thrown result may be caught by another event
    - trigger carries the information from throwing scope into catching scope
Start Events
None Event
- does not have a defined trigger
- only such a process can be called from a call activity
- processes using other types of start event cannot be called by call activities

Message Event
- arrives from a participant and triggers the start of a process
Start Events for top level processes

- **Timer Event**
  - a specific time, or period can be set to trigger the start of a process

- **Conditional**
  - based on conditions such as “arrival rate crosses 120/min threshold”
  - to trigger the event once again, the condition must become false and then true again.
  - it cannot use instance context variables since process instance is not created yet
  - can refer to static attributes in processes, or states of environment entities (how?: not defined in the standard)
Start Events for top level processes

- **Signal Event**
  - arrival of a signal event that is broadcast from another process
  - signal is not a message
  - multiple processes can use the same signal as start event

- **Multiple Start Event**
  - multiple ways of triggering the Process
  - only one is required

- **Multiple Parallel Start Event**
  - multiple events are required to trigger the start
  - all events must be triggered
only one type of start event can be used for sub-processes, which is the None Event.

this is the case for both embedded and called sub-processes

even if a sub-process has other types of start event along with a none event, the other events will not trigger the subprocess (they may however trigger it as a top level process)
Back to event subprocesses...
Start events for event sub-processes

- interrupting and non-interrupting Message event
- interrupting and non-interrupting timer event
- interrupting and non-interrupting escalation event
  - escalation sub-process expedites an activity for which an execution constraint (e.g. deadline) is not satisfied.
- Error start event: interrupting
- compensation start event: interrupting
Start events for event subprocesses:

- interrupting and non-interrupting conditional event
- interrupting and non-interrupting signal event
- interrupting and non-interrupting multiple event
- interrupting and non-interrupting parallel multiple event
End Events
End Events: they generate end event results

- none
  - no defined result
- message
  - message sent to a participant at the end of flow
  - show the participant through a connection
- error
  - named error is generated
  - all active threads in the subprocess are terminated
  - error gets caught by a catch error intermediate event (if it is specified) on the boundary of the nearest enclosing parent activity of this subprocess
End Events: they generate end event results

- escalation
  - triggers an escalation
  - other active threads continue
  - escalation event is caught by an catch escalation intermediate event on the boundary of the surrounding parent

- cancel
  - used in transaction sub-processes
  - it triggers cancel intermediate event attached to the transaction boundary

- compensation
  - indicates that a compensation is necessary

- signal
  - a signal event is broadcasted which can be received by any process that can receive the signal
End Events: they generate end event results

- terminate
  - all activities must be ended immediately
  - no compensation etc.

- multiple
  - means multiple consequences of ending
  - all of them will occur
Intermediate Events
Intermediate Events

- Event happens somewhere in between start and end of a process
- It does not directly end or start a process

Purposes:
- To show where messages are expected
- To show where messages are sent
- To show delays
- Generate exceptions and disrupt normal flow
- Compensation: place an intermediate event on the boundary of a task/subprocess, and use an outgoing flow from there
Placement of Intermediate events

- In the flow
  - To catch event trigger
    - Token stays at event till the trigger occurs (e.g. Message recd.) and then the token moves down the outgoing sequence flow
  - To throw event trigger
    - The trigger of event immediately happens (e.g. message gets sent) and then the token moves down the outgoing sequence flow

- On the boundary
  - To catch the event trigger
  - Circles are drown using double thin line
Intermediate Events in normal flow

- None
  - No specific trigger, indicates a point in flow
  - Cannot be used on boundary

- message
  - Catch, Throw messages
  - With catch, process flow continues
  - With throw, the exception handling path is followed

- error” Not used as intermediate event

- Timer Catch
  - Acts as delay

- Escalation
  - throws an escalation
Intermediate Events in normal flow

- compensation
  - Throws compensation
  - if the activity is identified, and it successfully completed, it will be compensated. The activity must be visible from compensation event
    - compensation intermediate event is contained in normal flow at the same level of subprocess
    - compensation intermediate event is contained in a compensation event subprocess which is contained in subprocess containing the activity.
  - if no activity is identified, all successfully completed activities visible to the compensation event will be compensated in reverse flow direction
    - those which occur in the same subprocess as that of the compensation event
    - those that occur in the same subprocess that contains the event subprocess in which the compensation event occurs
  - to be compensated, an activity must have
    - a boundary compensation OR
    - a compensation event sub-process
Intermediate Events in normal flow

- **signal**
  - Communication across pools, diagrams
  - Catch and throw type
  - Received by an activity only when attached to boundary
- **Conditional**
  - Catch event when a condition becomes true
Intermediate Events in normal flow

- **Multiple**
  - Catch and throw
  - Catch when attached to boundary
    - Only one of the assigned triggers is required
  - When used for throwing, all assigned triggers are thrown

- **Parallel Multiple**
  - It can only catch the triggers
  - All assigned triggers are required for it to trigger

- **Link**
  - Mechanism for connecting two sections of a process
  - Valid in normal flow only (not used on the boundary)
Intermediate events on the boundaries

- On the boundary of an activity, an intermediate event can only catch a trigger
- Interrupting event interrupts the flow, and exception path is followed
- Non-interrupting event resumes the flow
- Both interrupting and non-interrupting
  - Message, timer, escalation, conditional, signal, multiple, parallel multiple
- Only interrupting
  - Error, cancel, compensation
intermediate events on activity boundaries

- compensation
  - catches compensation
  - the event will be triggered by a thrown compensation targeting this activity
  - when event is triggered, the associated compensation activity is performed
  - compensation is triggered only after the activity is completed, thus they do not interrupt an activity. so the aspect of interruption vs. non-interruption is not applicable. (they cannot interrupt the activity)
intermediate events on activity boundaries

- **Message**
  - arrives from a participant and triggers the event
  - after the trigger, the flow changes to exception flow
  - can be interrupting the activity or it can also be non-interrupting
intermediate events on activity boundaries

- Timer
- Escalation: assumed not to abort the activity, but an interrupting version exists
- Error
  - always interrupts the activity
- Cancel
  - used with a transaction subprocess
  - triggered when a cancel end event is reached within the transaction subprocess
  - it always interrupts the activity
Intermediate events on activity boundaries

- **Conditional**
  - based on a conditional expression

- **Signal**
  - non-error condition

- **Multiple**
  - only one of the assigned triggers is required

- **Parallel Multiple**
  - all the assigned triggers are required
Examples
An example

- Book Hotel Room
- Cancel hotel
- Credit card payment
- Notify the failure To customer
Linking events
Strategies to forward thrown result from throwing events into catching events

- Publication
- Direct resolution
- Propogation
- Cancellation
- Compensation