# Activities

A **Task** is a unit of work, the job to be performed. When marked with a ⊞ symbol it indicates a **Sub-Process**, an activity that can be refined.

**Task**

A **Transaction** is a set of activities that logically belong together; it might follow a specified transaction protocol.

**Transaction**

An **Event Sub-Process** is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can interrupt the higher level process context or run in parallel (non-interrupting) depending on the start event.

**Event Sub-Process**

A **Call Activity** is a wrapper for a globally defined Sub-Process or Task that is reused in the current process.
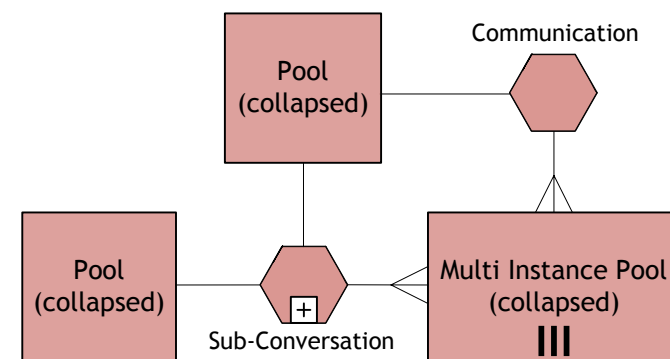
**Call Activity**

## Activity Markers
Markers indicate execution behavior of activities:

- ⊞ Sub-Process Marker
- ↻ Loop Marker
- ||| Parallel MI Marker
- ≡ Sequential MI Marker
- ∼ Ad Hoc Marker
- ◁◁ Compensation Marker

## Task Types
Types specify the nature of the action to be performed:

- Send Task
- Receive Task
- User Task
- Manual Task
- Business Rule Task
- Service Task
- Script Task

**Sequence Flow** defines the execution order of activities.

**Default Flow** is the default branch to be chosen if all other conditions evaluate to false.

**Conditional Flow** has a condition assigned that defines whether or not the flow is used.

# Gateways

**Exclusive Gateway**
When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.

**Event-based Gateway**
Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.

**Parallel Gateway**
When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.

**Inclusive Gateway**
When splitting, one or more branches are activated. All active incoming branches must complete before merging.

**Exclusive Event-based Gateway (instantiate)**
Each occurrence of a subsequent event starts a new process instance.

**Complex Gateway**
Complex merging and branching behavior that is not captured by other gateways.

**Parallel Event-based Gateway (instantiate)**
The occurrence of all subsequent events starts a new process instance.

# Conversations

A **Communication** defines a set of logically related message exchanges. When marked with a ⊞ symbol it indicates a Sub-Conversation, a compound conversation element.

A **Conversation Link** connects Communications and Participants.

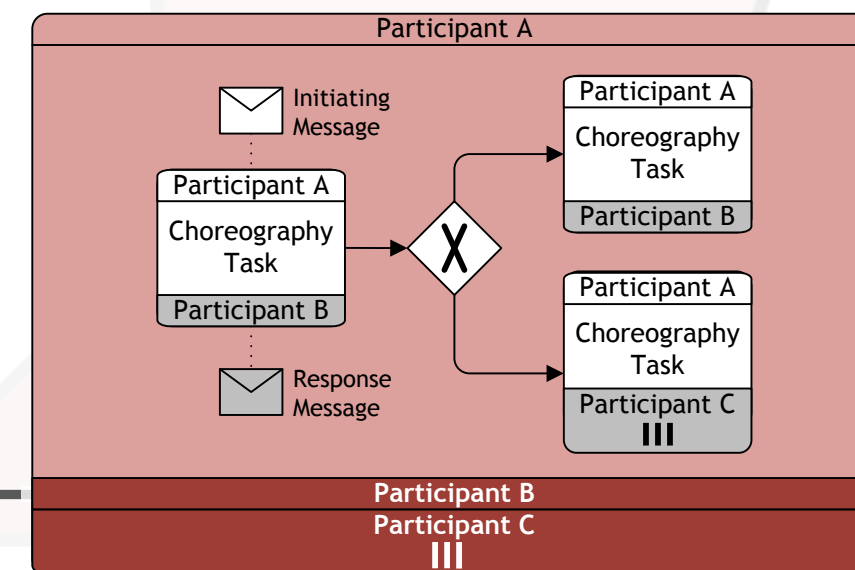A **Forked Conversation Link** connects Communications and multiple Participants.
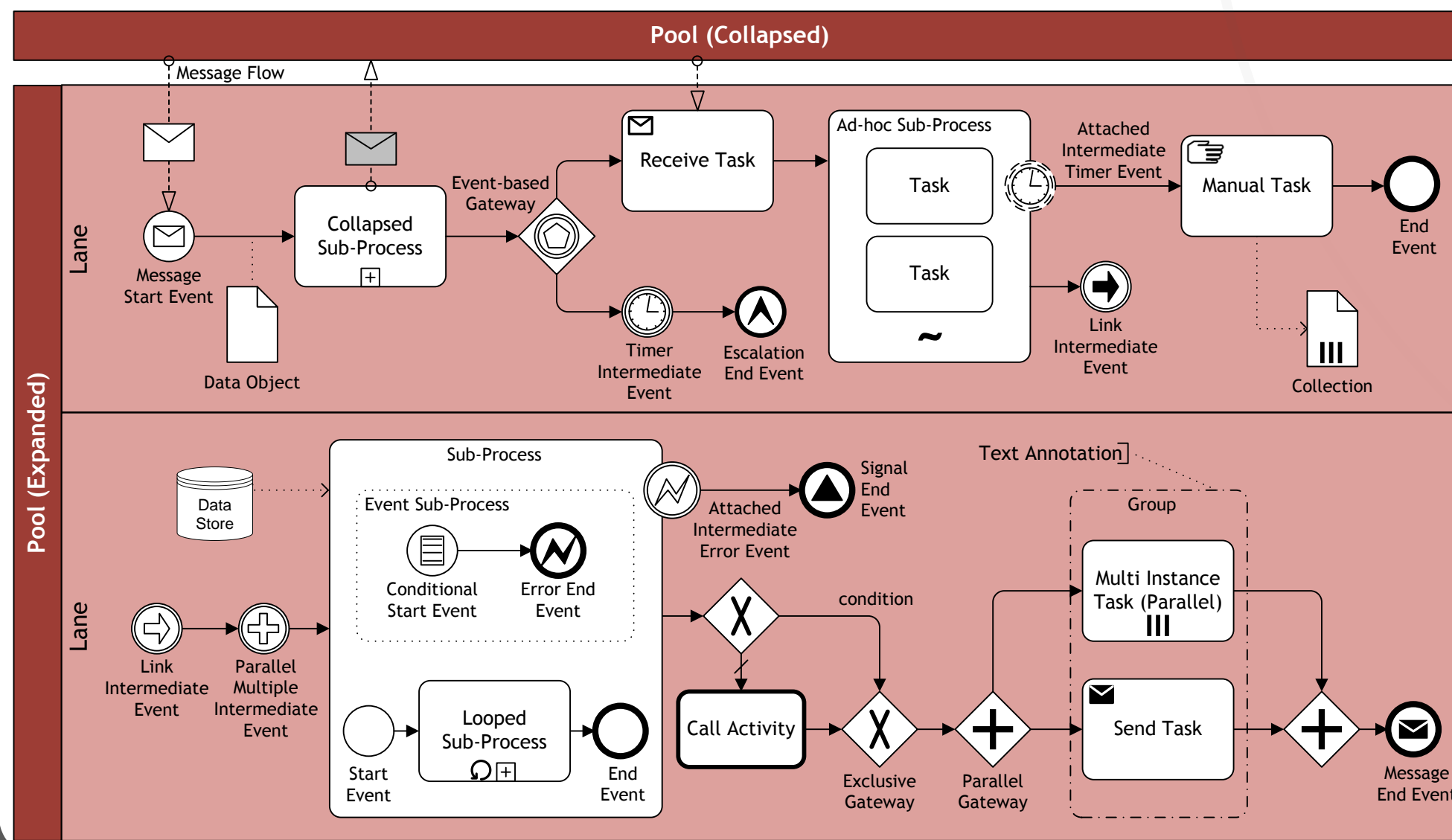
## Conversation Diagram

Pool (collapsed) — Communication

Pool (collapsed) — Sub-Conversation ⊞ — Multi Instance Pool (collapsed) |||

# Choreographies

**Participant A**
Choreography Task
Participant B

**Participant A**
Choreography Sub-Process ⊞
Participant B
Participant C

A **Choreography Task** represents an Interaction (Message Exchange) between two Participants.

**Multiple Participants Marker** denotes a set of Participants of the same kind.

A **Choreography Sub-Process** contains a refined choreography with several Interactions.
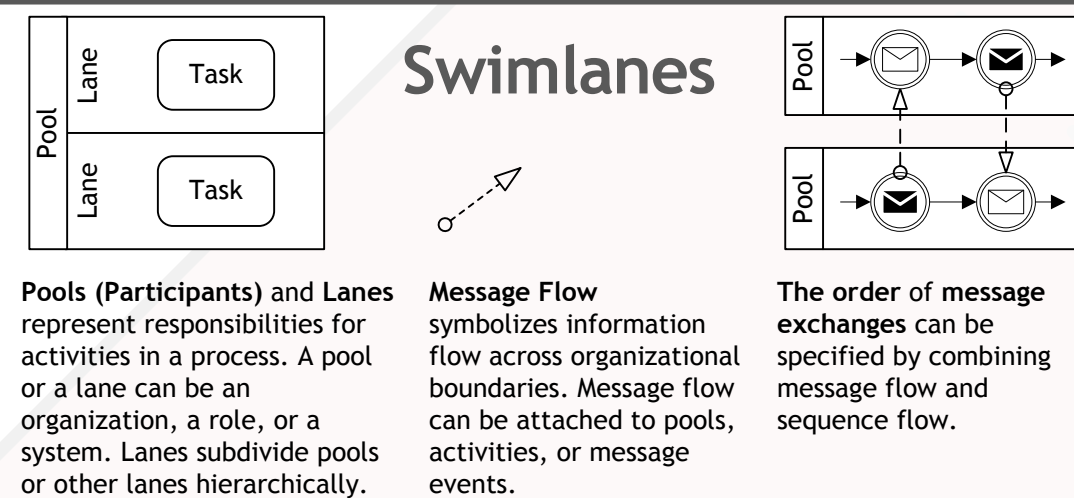
## Choreography Diagram

Participant A

Initiating Message

Participant A
Choreography Task
Participant B

Participant A
Choreography Task
Participant B

Participant A
Choreography Task
Participant C |||

Response Message

Participant B
Participant C |||

# Collaboration Diagram

**Pool (Collapsed)**

Message Flow

Lane
Message Start Event — Collapsed Sub-Process ⊞ — Event-based Gateway — Receive Task — Ad-hoc Sub-Process [Task / Task ∼] Attached Intermediate Timer Event — Manual Task — End Event

Data Object

Timer Intermediate Event — Escalation End Event — Link Intermediate Event — Collection |||

Pool (Expanded)

Data Store

Sub-Process
Event Sub-Process
Conditional Start Event — Error End Event

Attached Intermediate Error Event — Signal End Event

Text Annotation

Group
Multi Instance Task (Parallel) |||
Send Task

Lane
Link Intermediate Event — Parallel Multiple Intermediate Event

Looped Sub-Process ↻⊞

Start Event — End Event

Call Activity — Exclusive Gateway — condition — Parallel Gateway — Send Task — Parallel Gateway — Message End Event

# Events

| | Start | | | Intermediate | | | End |
|---|---|---|---|---|---|---|---|
| | Top-Level | Event Sub-Process Interrupting | Event Sub-Process Non-Interrupting | Catching | Boundary Interrupting | Boundary Non-Interrupting | Throwing |
| **None:** Untyped events, indicate start point, state changes or final states. | ◯ | | | | | | ◯ |
| **Message:** Receiving and sending messages. | ◉ | ◉ | ◌ | ◉ | ◉ | ◌ | ◉ |
| **Timer:** Cyclic timer events, points in time, time spans or timeouts. | ◷ | ◷ | ◌ | ◷ | ◷ | ◌ | |
| **Escalation:** Escalating to a higher level of responsibility. | | ◮ | ◌ | | ◮ | ◌ | ◭ |
| **Conditional:** Reacting to changed business conditions or integrating business rules. | ▤ | ▤ | ◌ | ▤ | ▤ | ◌ | |
| **Link:** Off-page connectors. Two corresponding link events equal a sequence flow. | | | | ◉ | | | ◉ |
| **Error:** Catching or throwing named errors. | | ◈ | | ◈ | | | ◈ |
| **Cancel:** Reacting to cancelled transactions or triggering cancellation. | | | | ◉ | | | ◉ |
| **Compensation:** Handling or triggering compensation. | | ◁ | | ◁ | | | ◀ ◀ |
| **Signal:** Signalling across different processes. A signal thrown can be caught multiple times. | △ | △ | ◌ | △ | △ | ◌ | ▲ ▲ |
| **Multiple:** Catching one out of a set of events. Throwing all events defined. | ⬠ | ⬠ | ◌ | ⬠ | ⬠ | ◌ | ⬟ ⬟ |
| **Parallel Multiple:** Catching all out of a set of parallel events. | ⊕ | ⊕ | ◌ | ⊕ | ⊕ | ◌ | ⊕ ⊕ |
| **Terminate:** Triggering the immediate termination of a process. | | | | | | | ● |

# Data

**Input** — Task — **Out-put**

A **Data Input** is an external input for the entire process. It can be read by an activity.

A **Data Output** is a variable available as result of the entire process.

A **Data Object** represents information flowing through the process, such as business documents, e-mails, or letters.

A **Collection Data Object** represents a collection of information, e.g., a list of order items.

A **Data Store** is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.

A **Message** is used to depict the contents of a communication between two Participants.

# Swimlanes

Pool — Lane — Task
Pool — Lane — Task

Pool
Pool

**Pools (Participants)** and **Lanes** represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.

**Message Flow** symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events.

**The order** of message exchanges can be specified by combining message flow and sequence flow.

BPM Offensive Berlin

HPI Hasso Plattner Institut — IT Systems Engineering | Universität Potsdam

camunda — the business process company

HUMBOLDT-UNIVERSITÄT ZU BERLIN

inubit — integrating your business and IT

SIGNAVIO — simply professional