

# Choreography and Orchestration of Services

Oxford University  
Software Engineering  
Programme  
April 2021



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# Business Process Management

- Hammer & Champy [1993] “A collection of activities that takes one or more kinds of input and creates an output that is of value to the customer.”
- Davenport [1992] “A structured, measured set of activities designed to produce a specific output for a particular customer or market. It implies a strong emphasis on how work is done within an organization, in contrast to a product focus’s emphasis on what.”



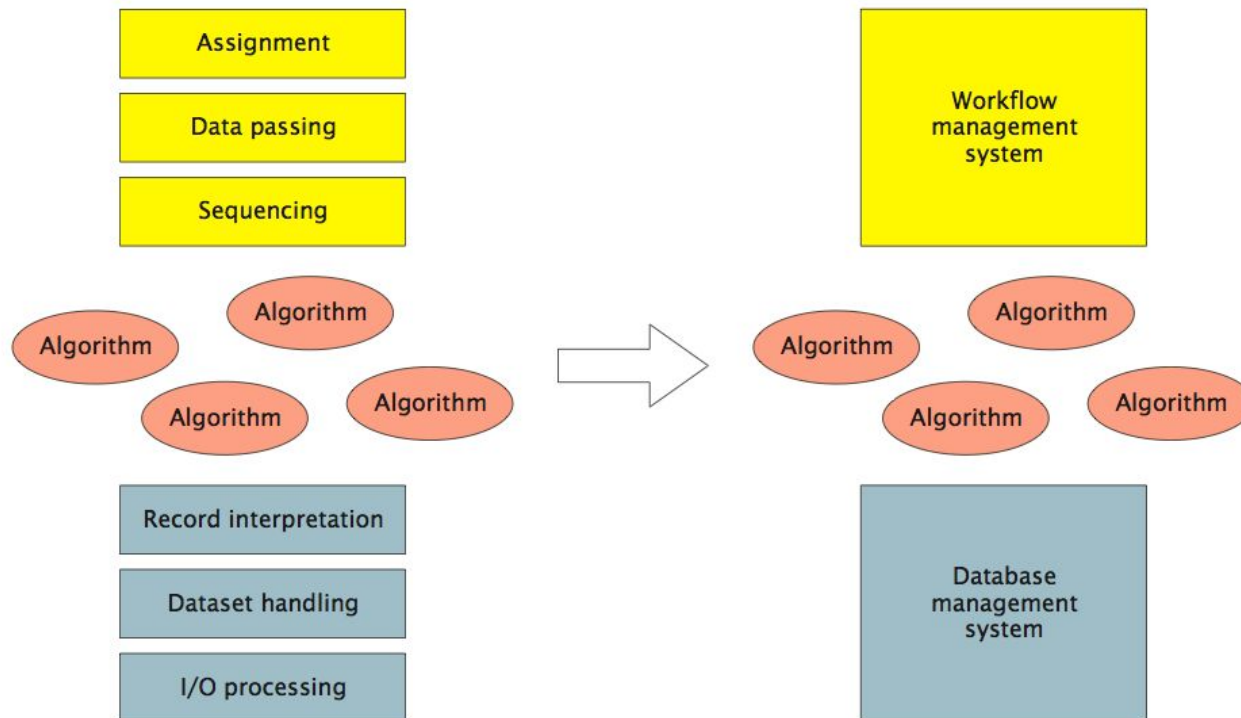
# Composition

- Services provide *platform- and language-independent* access to *software components*
- But these components are *isolated*: they need to be *assembled* into *service-oriented architectures*
- Ideally, they should be recursively *composable* to form composite services in their own right
- *Workflow* languages for scripting or ‘glue’ between individual services
- BPMN, WSCI, WSFL, XLANG, BPEL. . .
- beyond mere *business protocol specifications* like RosettaNet, which are essentially paper specifications so can’t be automated and won’t scale



# Removal of Dependencies (Leymann and Roller)

- DBMS provides independence from data *representation*; workflow provides independence from control or data *flow*.



# Motivations

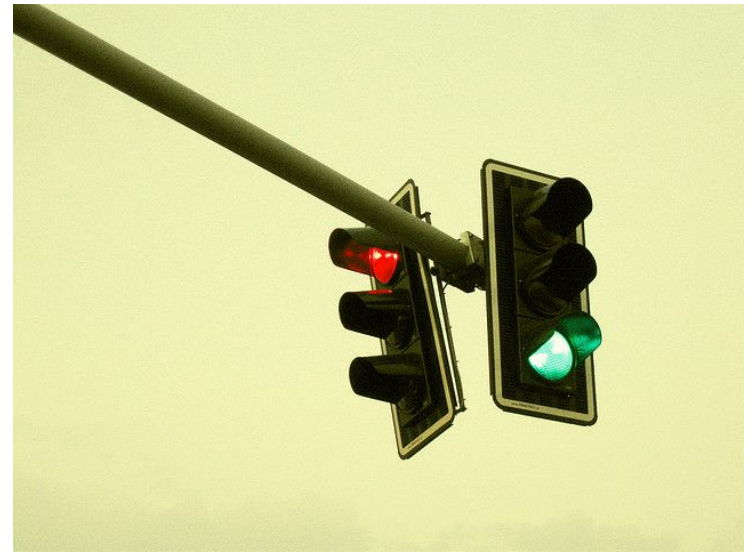
- Model Business Processes
  - Understand what happens?
  - Who is responsible?
  - What is involved?
- Simulate
  - Improve and model
- Execute
  - Automate processes
  - Improve them more quickly
- Monitor
  - Get a real-time health status of processes



# Orchestration vs Choreography



<http://www.flickr.com/photos/herrolm/>



<http://www.flickr.com/photos/tasuki/>



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License  
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

# Orchestration vs Choreography

- *Orchestration*
  - Describes procedure
  - instructs participants globally – imperative; centralized
  - typically deterministic: ‘must’
- *Choreography*
  - Describes protocol
  - Constraints on interaction, but participants act locally – declarative; no ‘current state’
  - Usually non-deterministic: ‘may’
- Orchestra has a conductor, Ballet does not



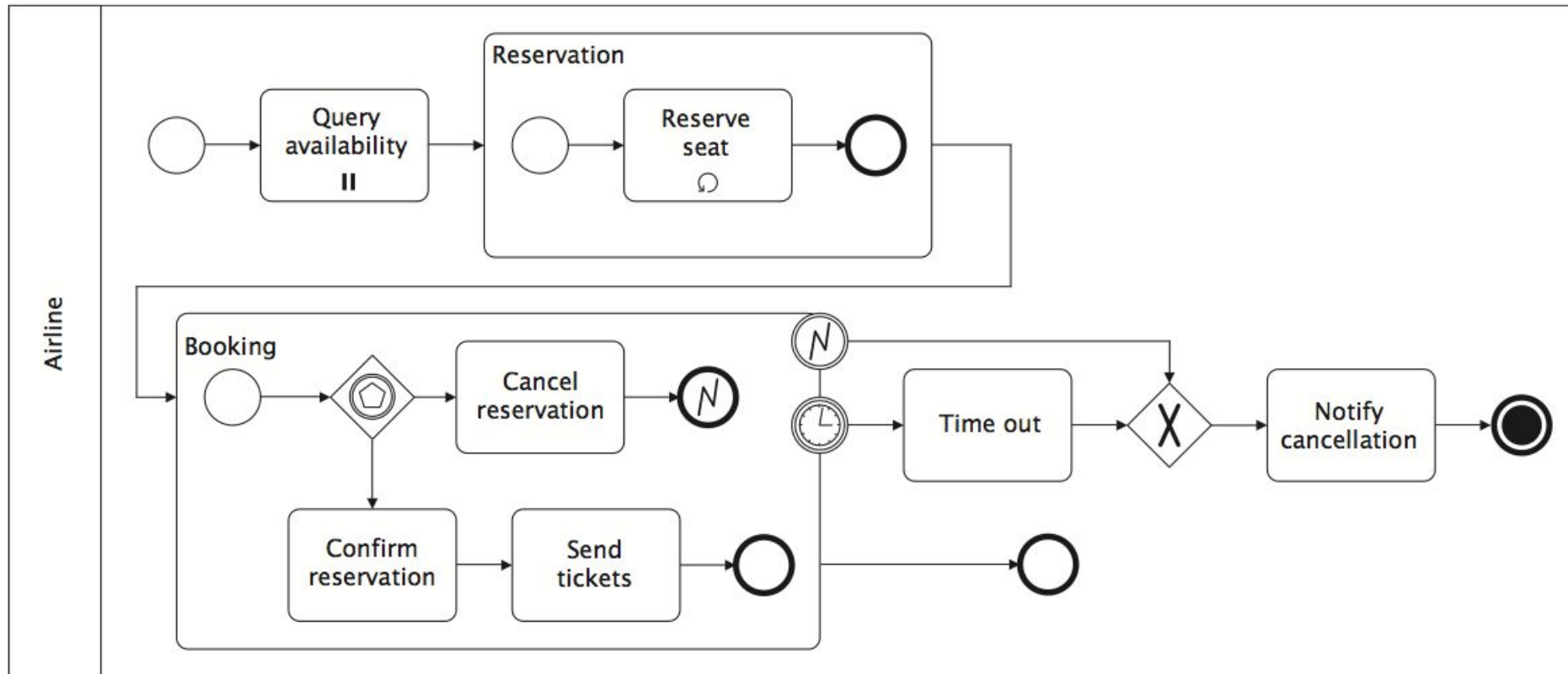
# BPMN 1.1

- Designed to allow process designers to communicate
  - Think UML
- Activities, Gateways, Events
- Control and Data Flow
- Organization modelling (Pools, Swimlanes)

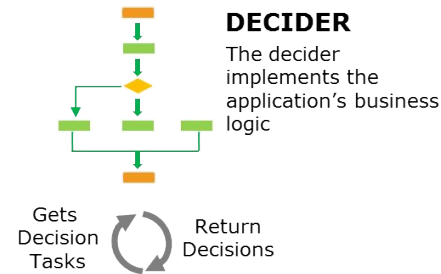




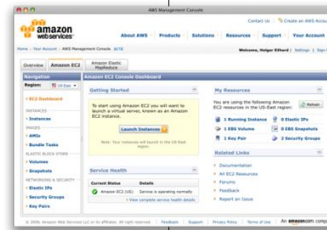
# BPMN Example



# Amazon Simple Workflow Service



## Amazon SWF



- Maintains distributed application state
- Tracks workflow executions
- Ensures consistency of execution history
- Provides visibility into executions
- Holds and dispatches tasks
- Provides control over task distribution
- Retains workflow execution history

<http://aws.amazon.com/swf/>



Cloud



Workers for Activity 1



Mobile



Workers for Activity 2



On Premises



Workers for Activity 3



# Turbine

---

Turbine is a JavaScript workflow engine. It vastly simplifies the development, deployment, and testing of complex web applications through the use of declarative workflows that express your app's program logic in a form that is simple to read and to understand.

## Why Turbine?

---

Turbine is the ideal solution for apps (or parts of apps) with multi-step processes involving many possible branches, sub-flows, or permutations. Examples include:

- Signup forms
- Login forms
- Interactive tours
- Shopping carts
- Checkout flows
- Asset creation (i.e. upload photo -> add filter -> add caption -> tag friends -> share)
- etc.

The programming of these types of apps usually involves a tangled nightmare of conditionals, switches, callbacks, promises, and other strands of spaghetti code.

This tightly coupled code makes it almost impossible to A/B/n test different flows or variations -- any attempt to do so usually makes the problem even worse. It is also very difficult to follow the program logic to trace all the possible flows through the code.

# Business Process Execution Language (BPEL)

- Standardised XML language for executable processes
- Well defined execution
  - No deadlocks
  - Graphs must be acyclic
- Tied to WSDL concepts
- No built in support for human activities (though this has been added)
- No graphical notation



# The main strength of BPEL

(IMO)

- BPEL is a completely executable standalone language
  - PartnerLinks define places where you can call WSDL services
  - Or where other parties can call WSDL Services into the process
- Deployment descriptor + BPEL can be executed without any Java or other language



# The main weaknesses of BPEL

(IMO)

- Too much like a programming language
  - Need WS-HumanTask, BPEL4People and script or Java extensions to make it useful for real processes
- No swimlanes (explained in a minute)
- No common visual notation



# BPMN + BPEL

- In theory:
  - Process experts design and model in BPMN
  - Developers/Implementors implement in BPEL
- No standard bridging/mapping
  - Double the effort



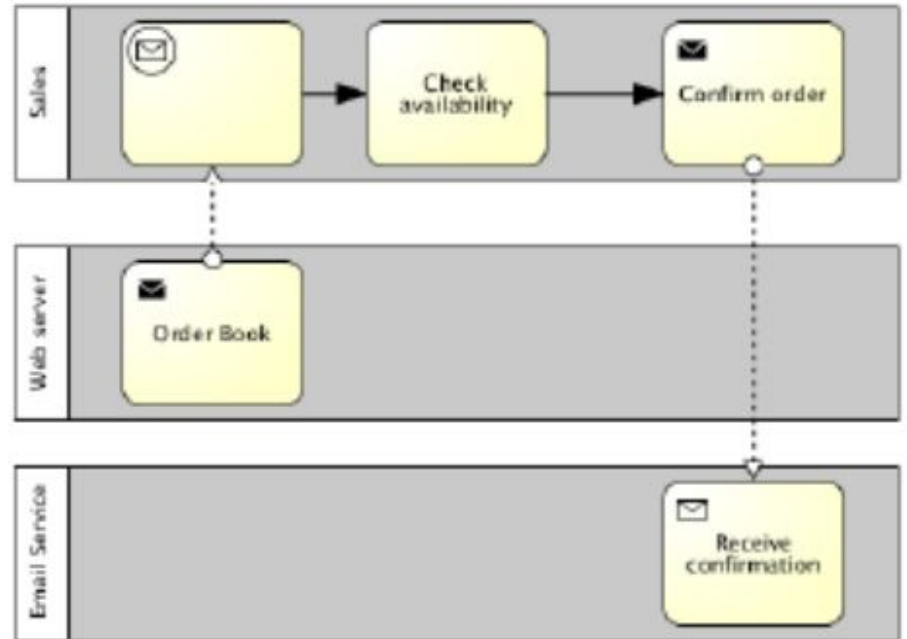
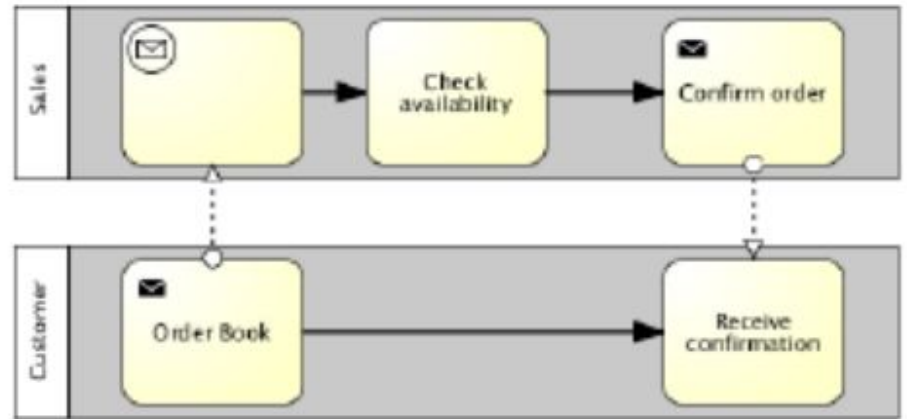
# BPMN 2.0

- A notation for a subset of BPEL
- Execution semantics for BPMN
- Notational support for choreography
- The best of both worlds?
- Cons:
  - Need to write external logic in another language to implement a process

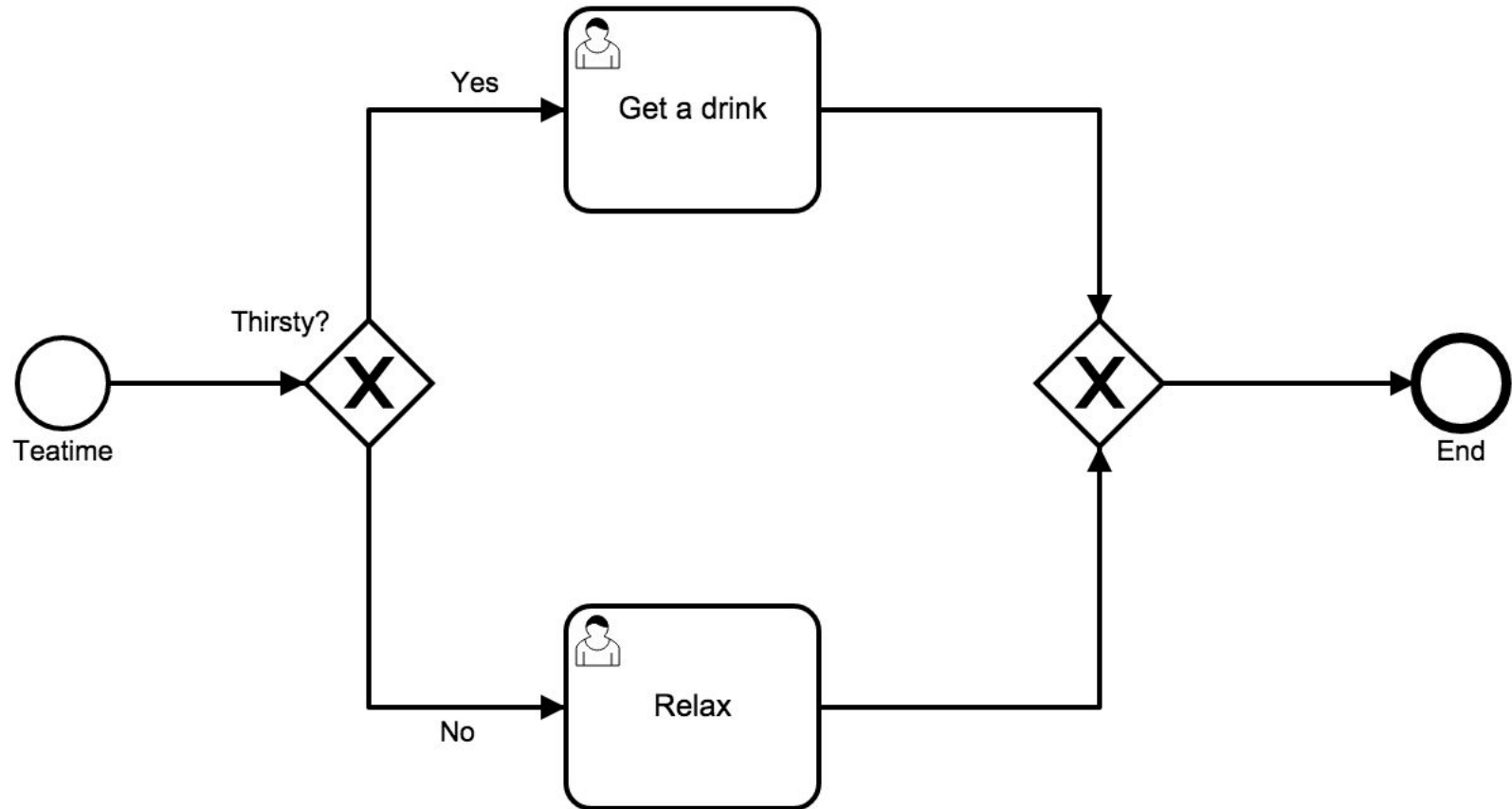




# BPMN 2.0

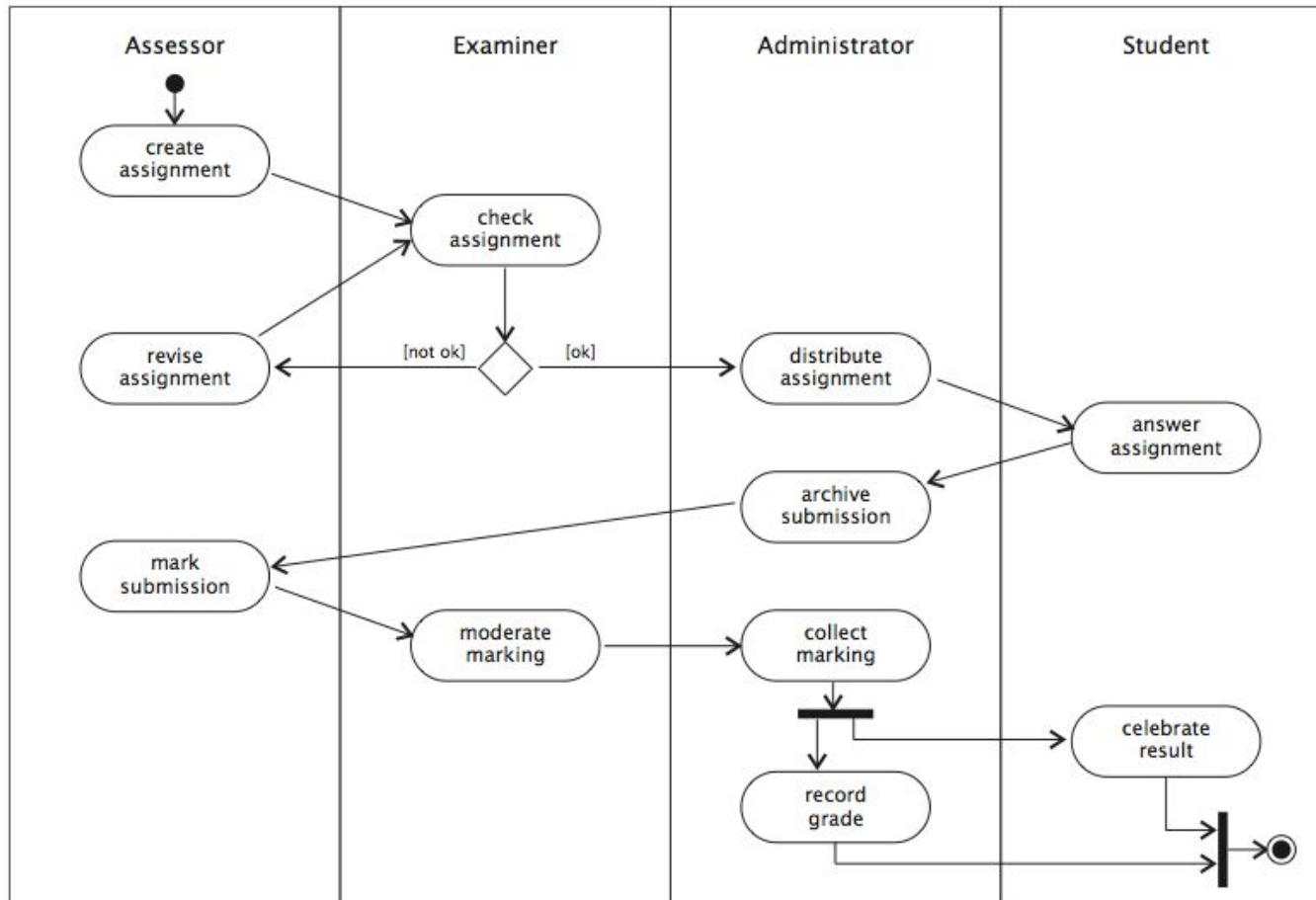


# BPMN 2.0 Basics

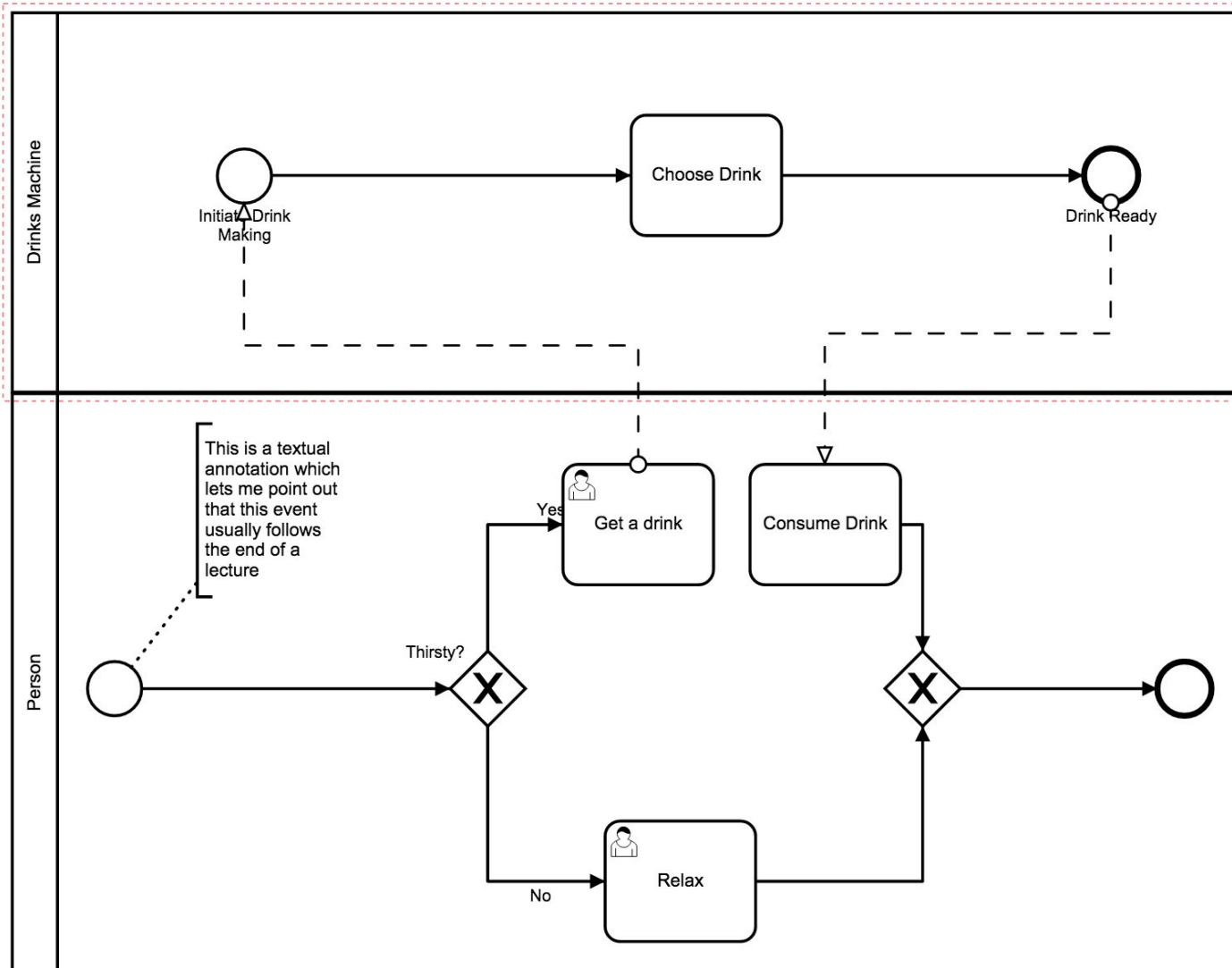


# Swimlanes:

partition an activity diagram into the responsibilities of different entities

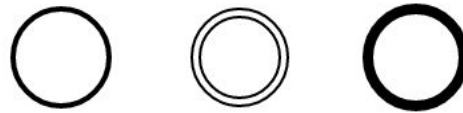


# Swimlanes represent different participants

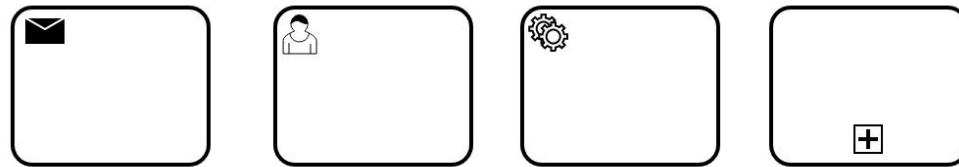


# BPMN Basic Constructs

- Events



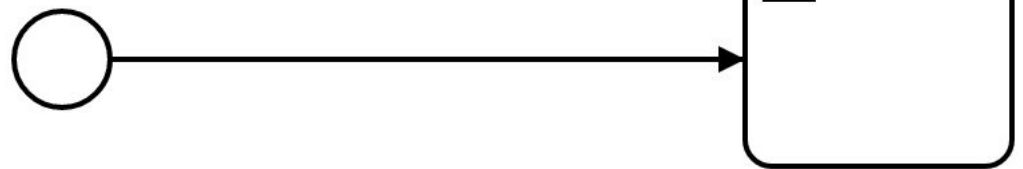
- Activities



- Gateways

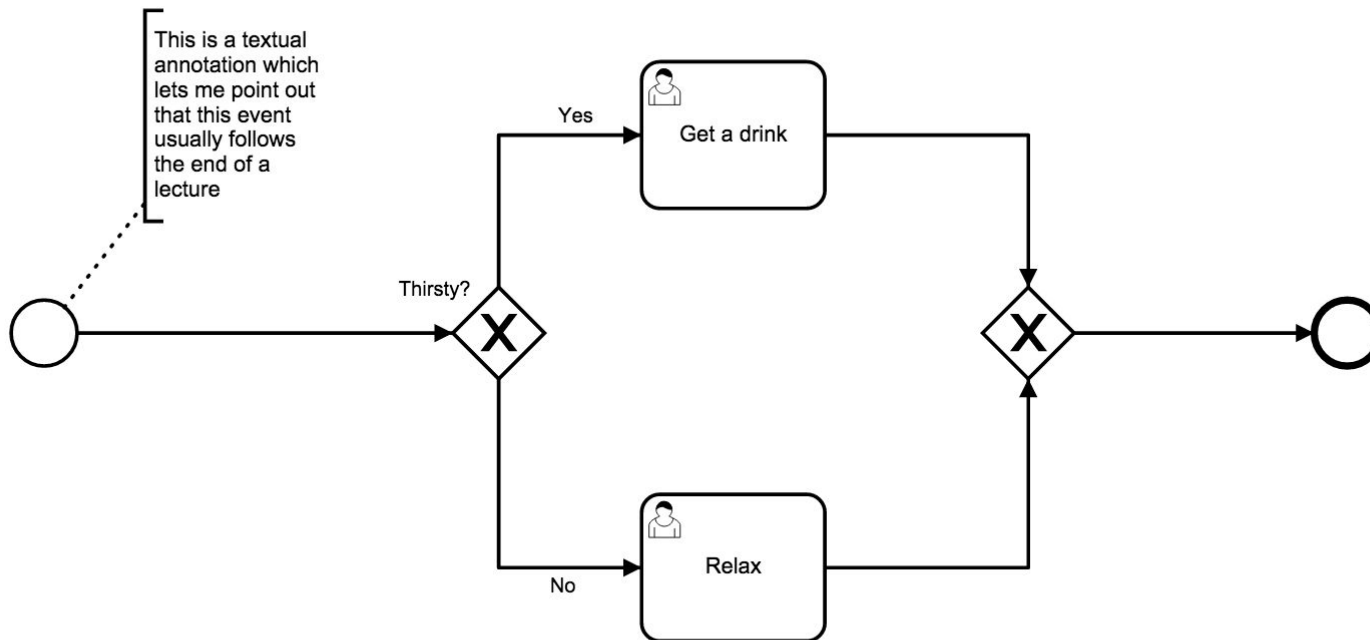


- Sequence Flow








# Text Annotations

- How you document your processes



# Start Events

- Start Event 
  - Message Start 
  - Timer Start 
  - Conditional Start 
  - Signal Start 

# Some Intermediate Events



Basic Intermediate Event



Message Catch



Message Throw



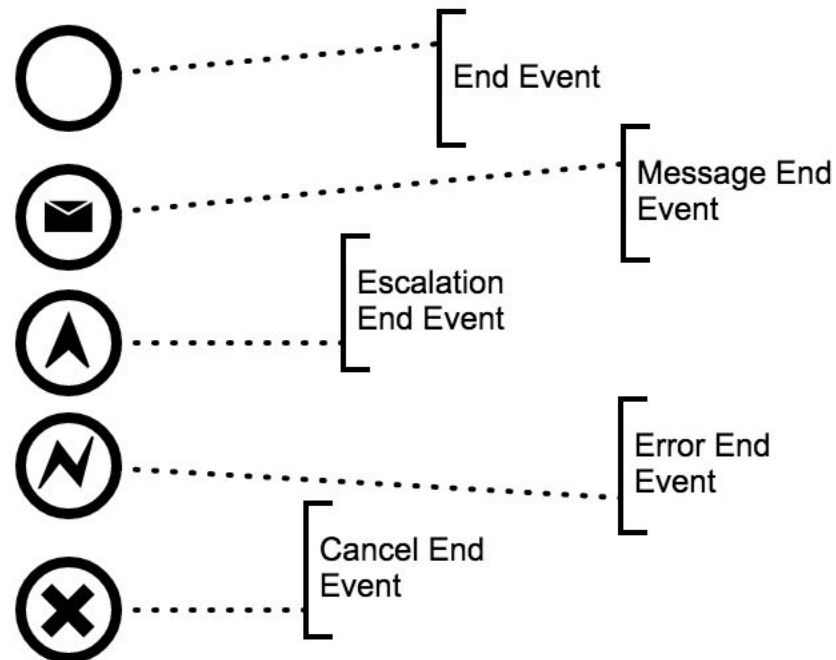
Timer



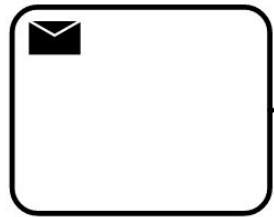
Escalation



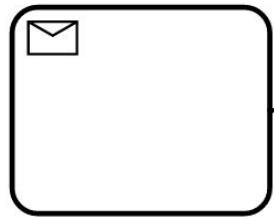
# Some End Events



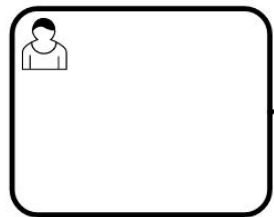
# Activities



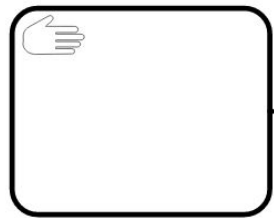
Send Task  
- sends a message



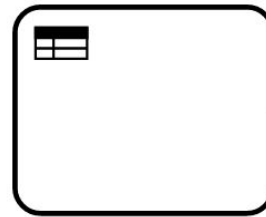
Receive Task  
- receives a message



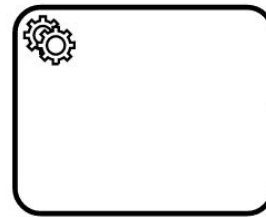
User Task  
- requires input  
or action from a  
user, mediated  
by the engine



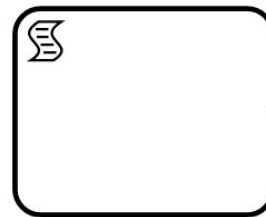
Manual Task  
- completely manual task (out  
side the engine)



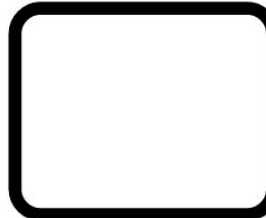
Business Rule Task  
- implemented by a BR engine



Service Task  
- implemented by a service



Script Task  
- implemented by a script



Call Activity-  
calls another  
process

# Service Task

- Call a service
  - Unlike BPEL there is no direct way of capturing the actual service
  - Typically need to write some code (e.g. Java) to capture that

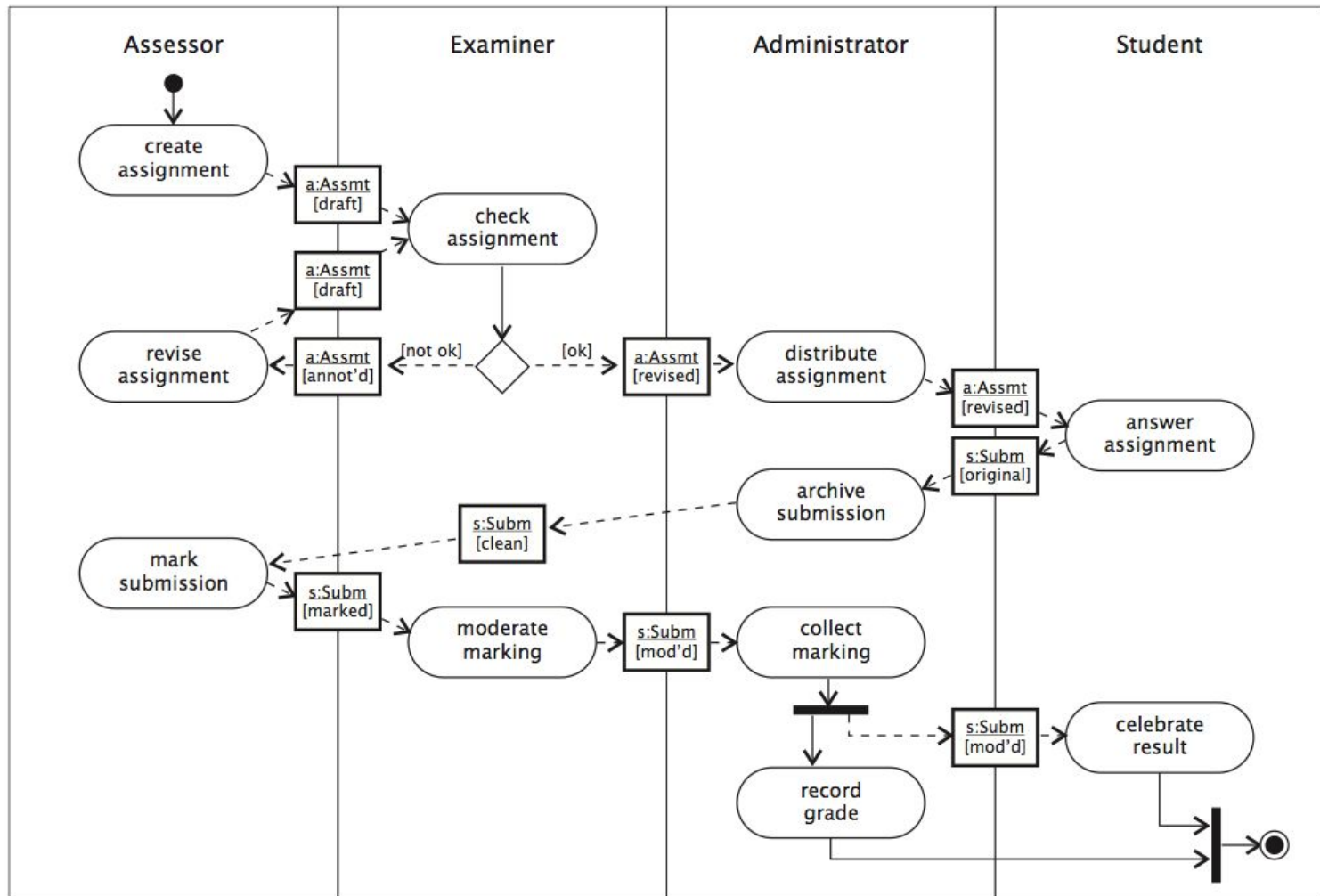


# Data Flow

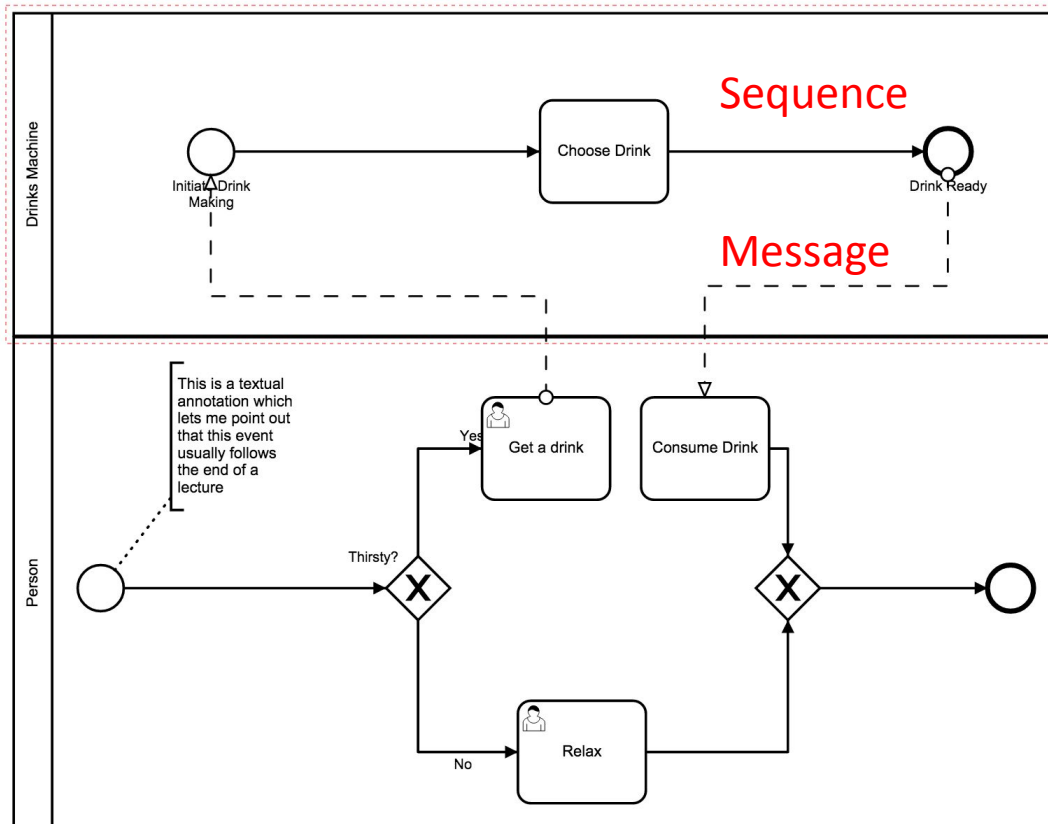
- Transitions between activities represent *control dependencies*: one activity must complete before another can start
- Workflows also have *data dependencies*: one activity produces a result that another requires
- UML activity diagrams allow *object flow* as well as *control flow*
- Dependent data is shown as an object icon (rectangle with underlined name and type)
  - *dependencies* shown as dashed arrows from generating activity to object, and from object to consuming activity(s)
  - same object may occur multiple times in an activity diagram, typically in different *states* (shown in square brackets after object name)



# Example Object Flow



# Flows



Sequence flows are within a Swimlane

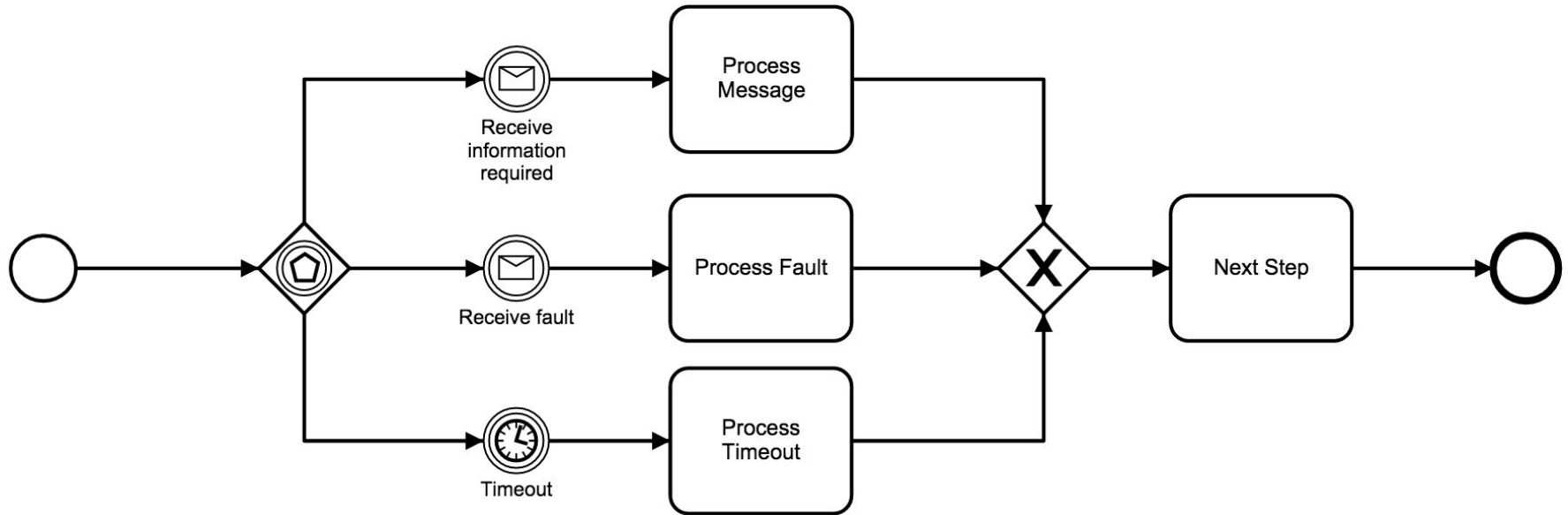
Message flows between swimlanes

# Gateways

- Exclusive Gateway
  - Fork – choose one path (if/else)
  - Join – wait for a single event
- Parallel Gateway
  - Fork – do both / all paths
  - Join – wait for all inputs



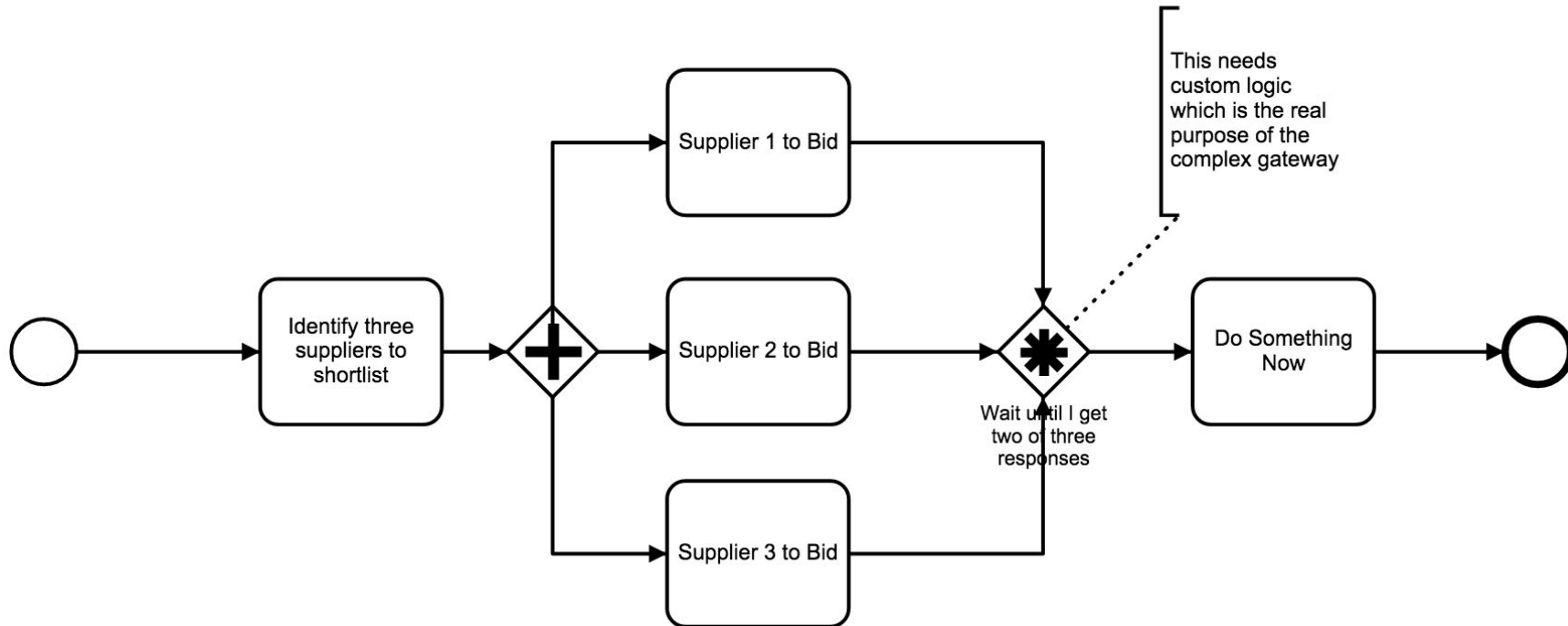
# Event Gateway



*An Event Gateway allows different events to trigger different actions*



# Complex Gateway



# How much BPMN do you need?

How Much Language is Enough?  
Theoretical and Practical Use of the  
Business Process Management Notation  
[http://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=2038665](http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2038665)

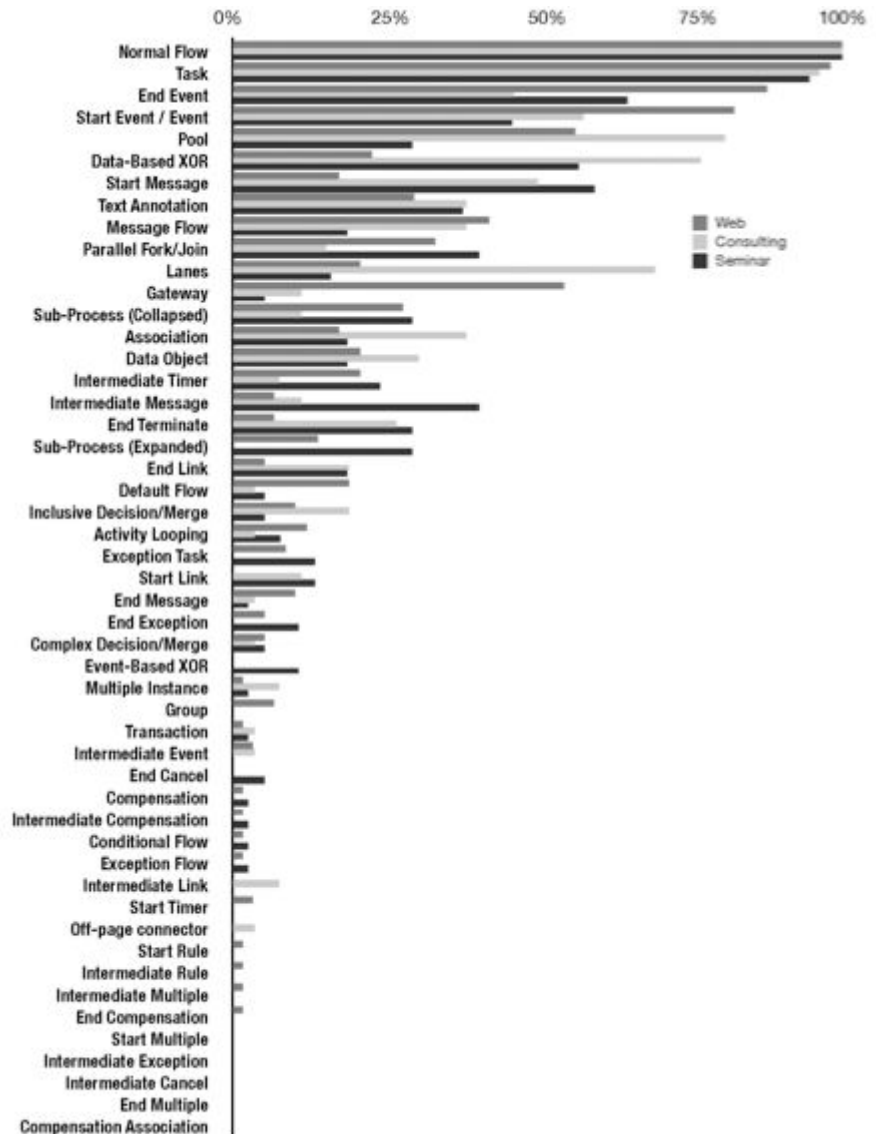


Fig. 1. Occurrence Frequency of BPMN Constructs

# Summary

- Process Management has a strong place in composing SOA systems
  - Externalising dependencies
  - Agility
  - Sharing with the business owners

