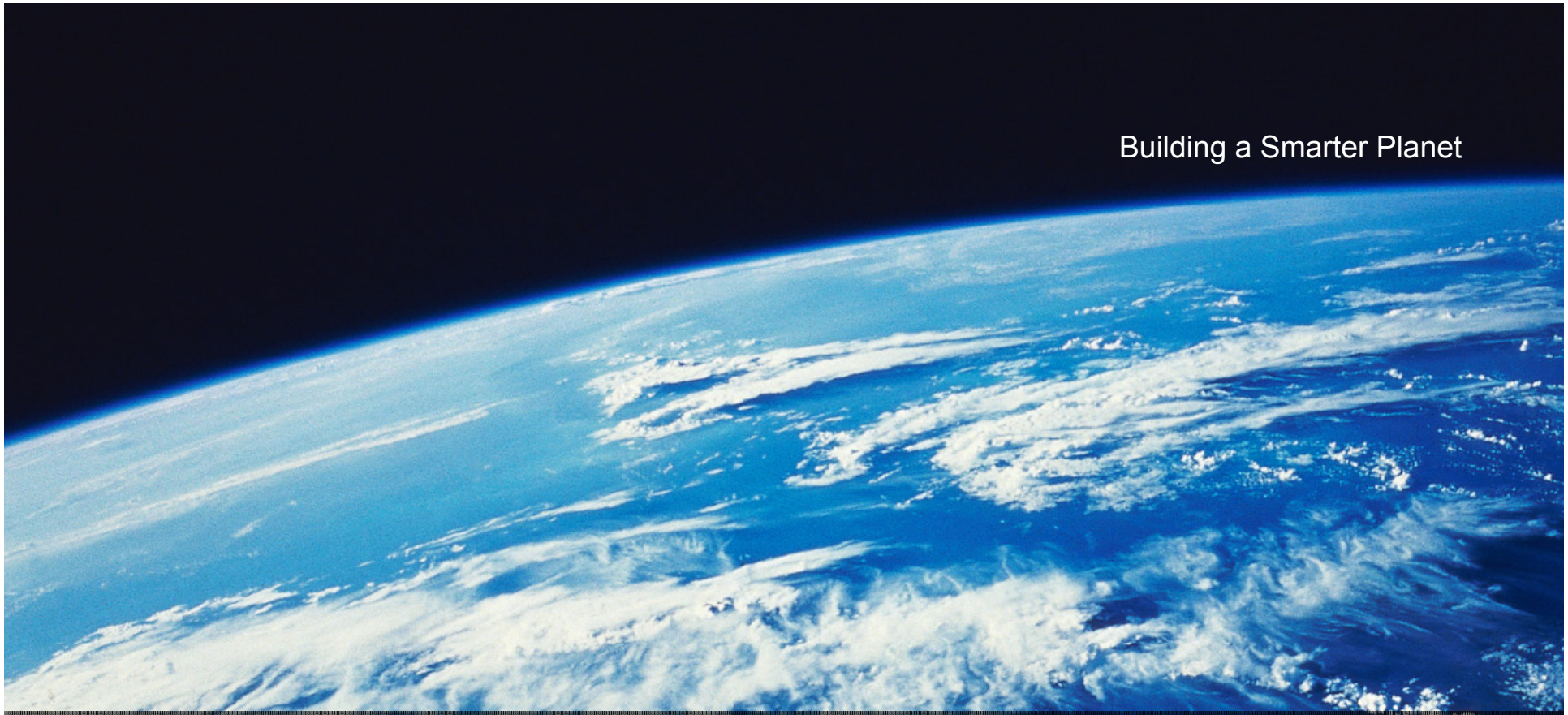


SFM-09WS – 9th International School on Formal Methods for the Design of Computer, Communication and Software Systems – Web Services – June 1-6, 2009 – Bertinoro

Web Service Orchestration and WS-BPEL 2.0

Dieter König – IBM Senior Technical Staff Member – dieterkoenig@de.ibm.com

IBM

Abstract

- Business Processes not only play a key role in Business-to-Business (B2B) and Enterprise Application Integration (EAI) scenarios by exposing the appropriate invocation and interaction patterns but they are the fundamental basis for building heterogeneous and distributed applications (workflow-based applications).
- Web Services Business Process Execution Language (WS-BPEL) provides the language to specify business processes that are composed of Web services as well as exposed as Web services. Business Processes specified via WS-BPEL are portable; they can be carried out by every WS-BPEL compliant execution environment.

Dieter König

- Dieter König is a Senior Technical Staff Member with IBM Software Group's laboratory in Böblingen and architect for workflow components in WebSphere Process Server. He joined the laboratory in 1988.
- Dieter was a member of the OASIS WS-BPEL Technical Committee, which created the industry standard for the Web Services Business Process Execution Language (WS-BPEL), and is now an active member of OASIS Technical Committees for BPEL4People and Service Component Architecture.
- He has published many articles and has given talks at conferences about Web services and workflow technology, and is co-author of three books about Web services and Business Process Management.
- Dieter holds a Dipl. inform. degree in computer science from the University of Bonn, Germany.

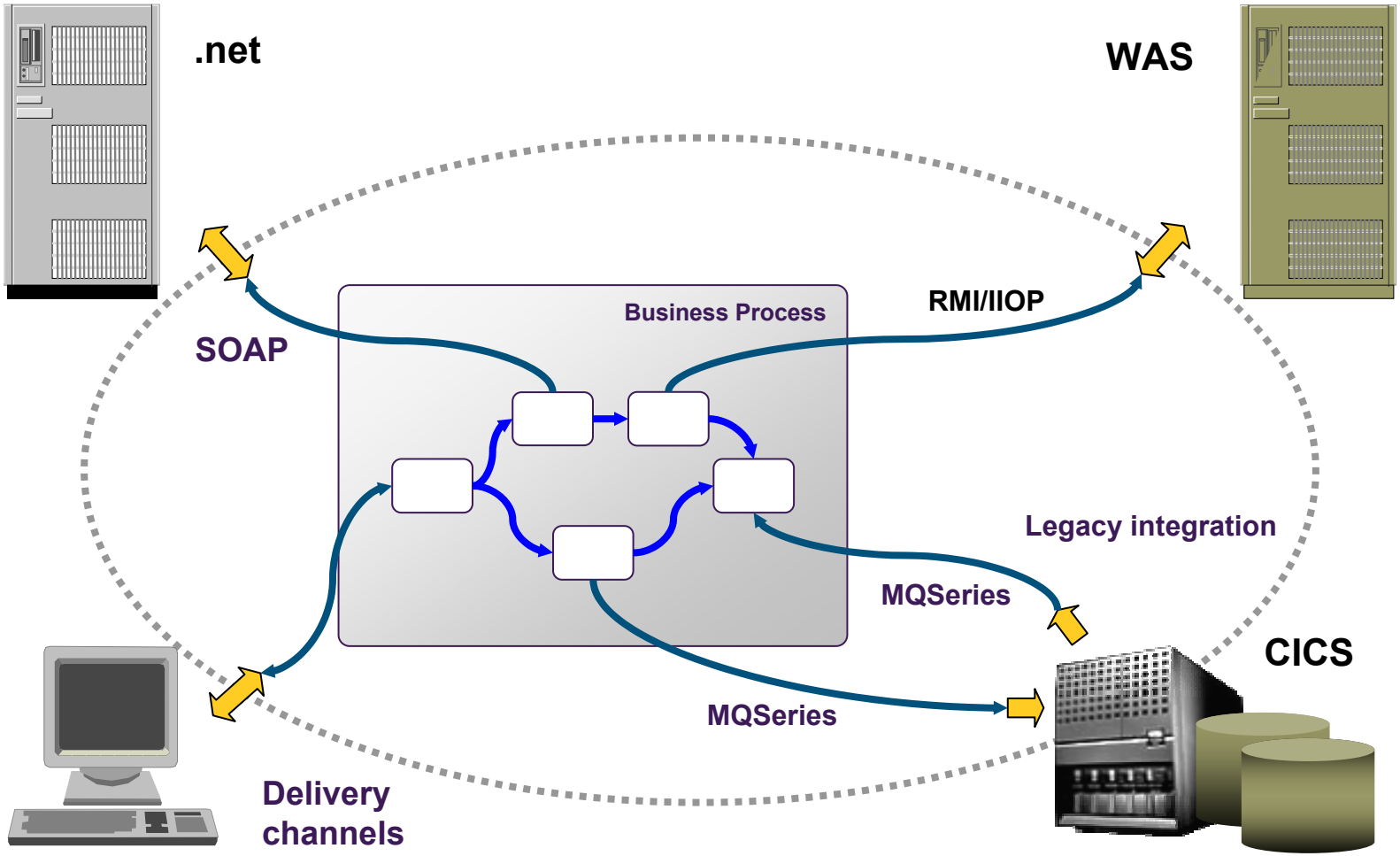
Web Service Orchestration and WS-BPEL

- Part I – Web Service Orchestration
  - Motivation and Overview
  - Business Process Modeling Styles
- Part II – The WS-BPEL 2.0 Standard
  - WS-BPEL 2.0 Concepts and Language Elements
- Part III – WS-BPEL 2.0 Extensions
  - WS-BPEL Extension for People (BPEL4People)
  - WS-BPEL Extension for Subprocesses (BPEL-SPE)
  - WS-BPEL Extension for Java (BPELJ)
- Part IV – Additional Related Standards
  - Service Component Architecture (SCA)
  - Business Process Modeling Notation (BPMN)

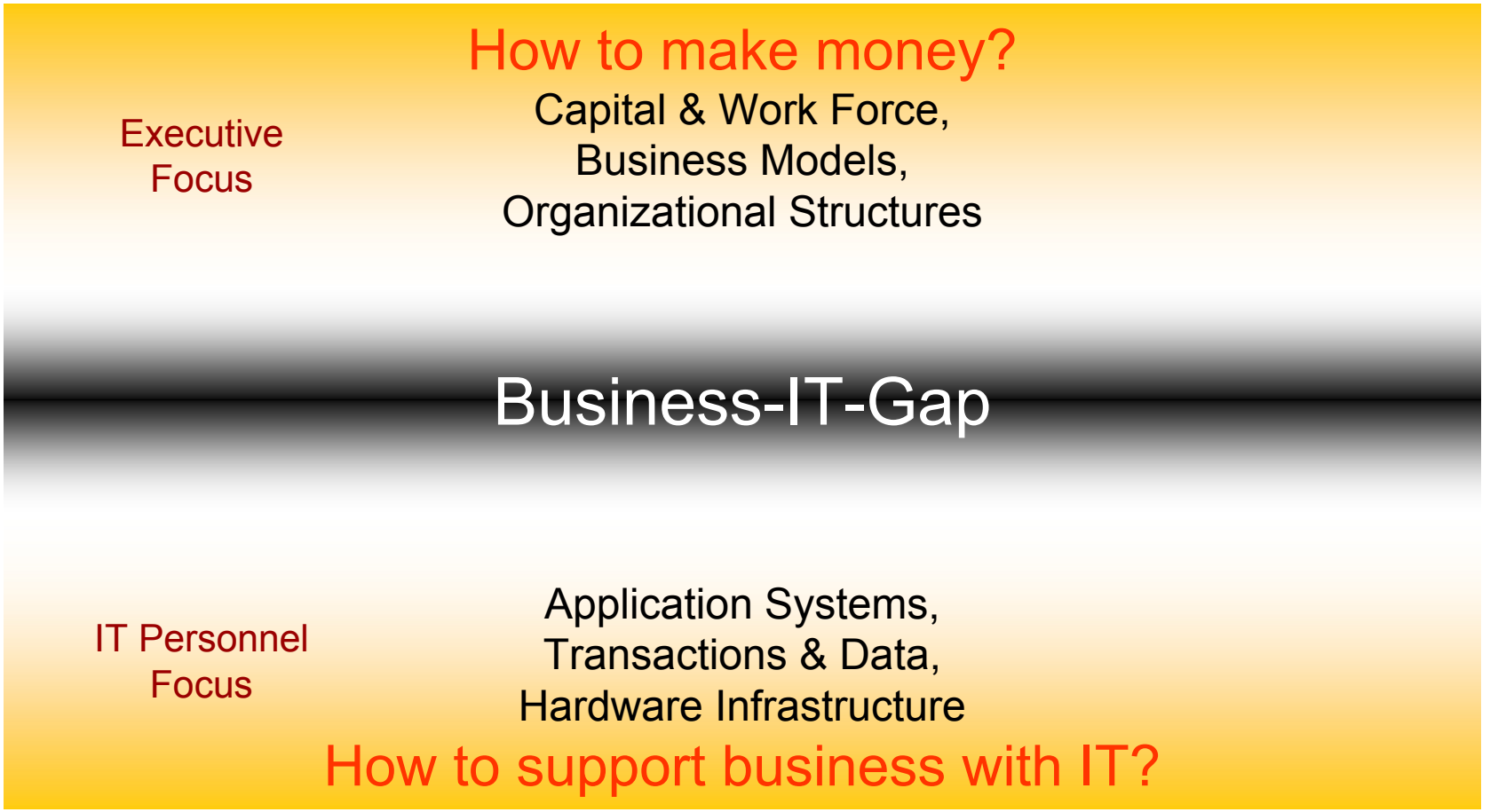
Motivation

- Integration continues to be a key problem facing businesses
  - Intra-enterprise integration (Enterprise Application Integration)
  - Integrating with partners (Business-to-Business Integration)
- Web services → move towards service-oriented computing
  - Applications are viewed as “services”
  - Loosely coupled, dynamic interactions
  - Heterogeneous platforms
  - No single party has complete control
- Service composition
  - How do you compose services in this domain?

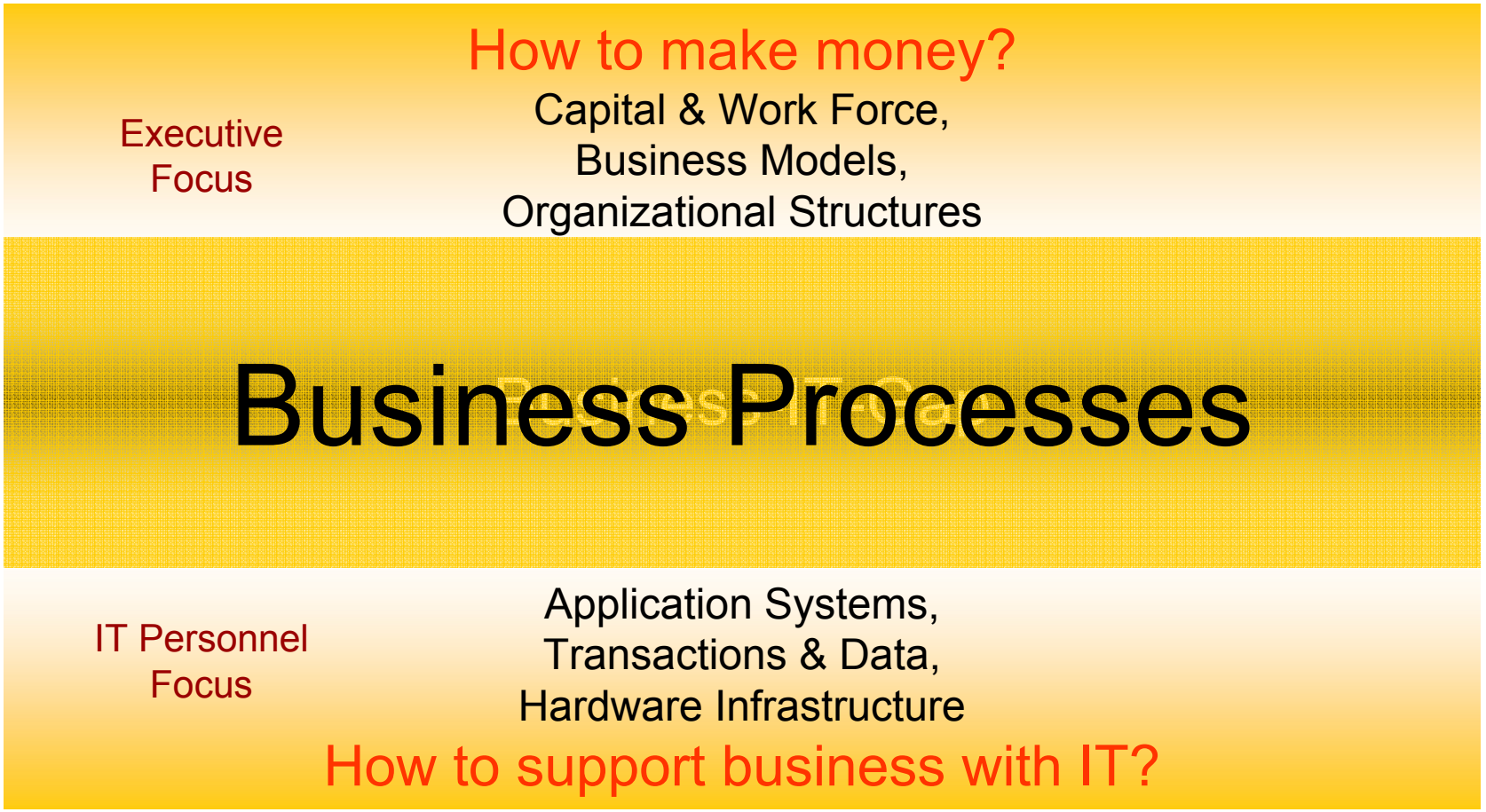
Application Integration



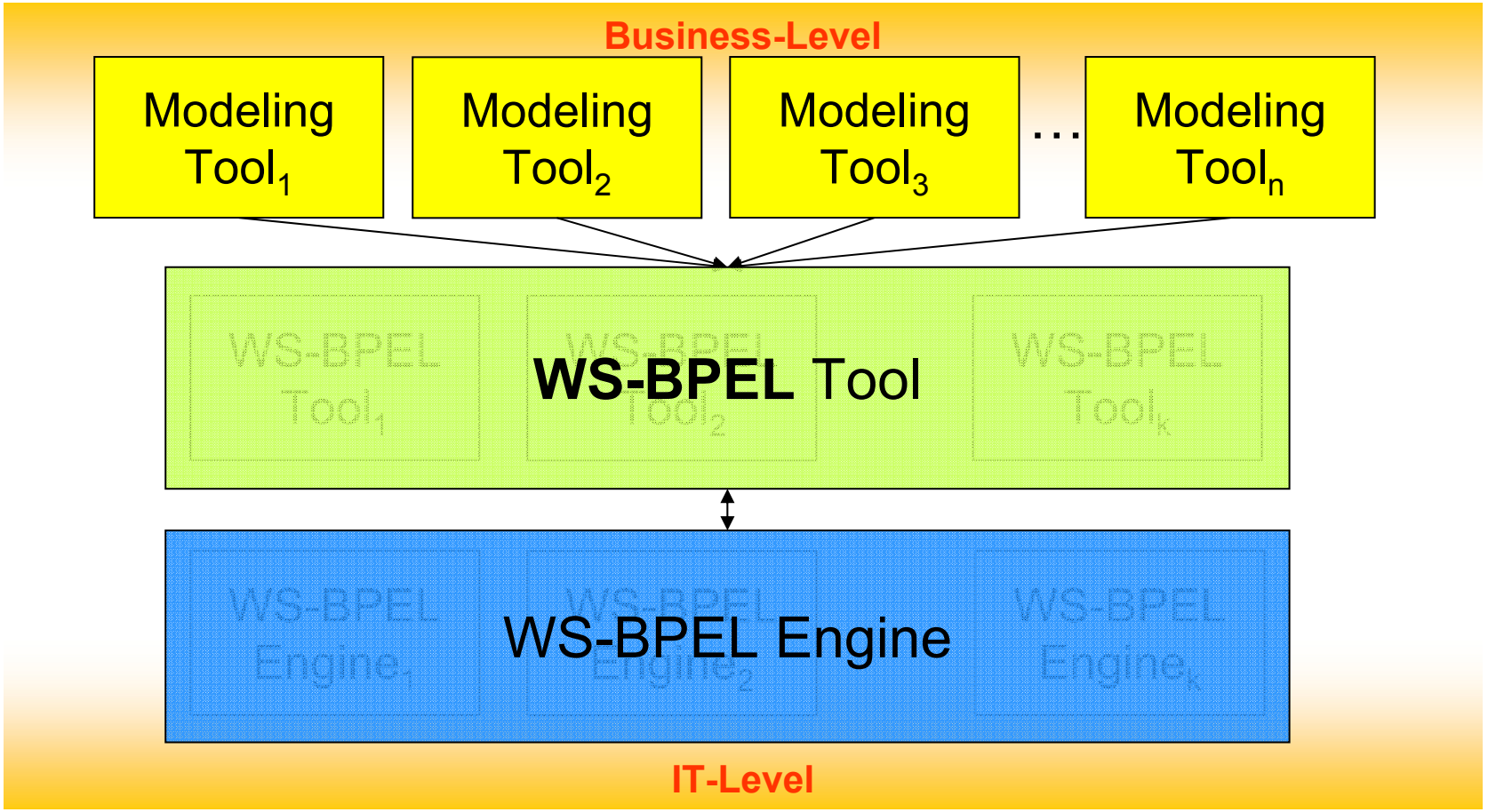
A Well-Known Problem



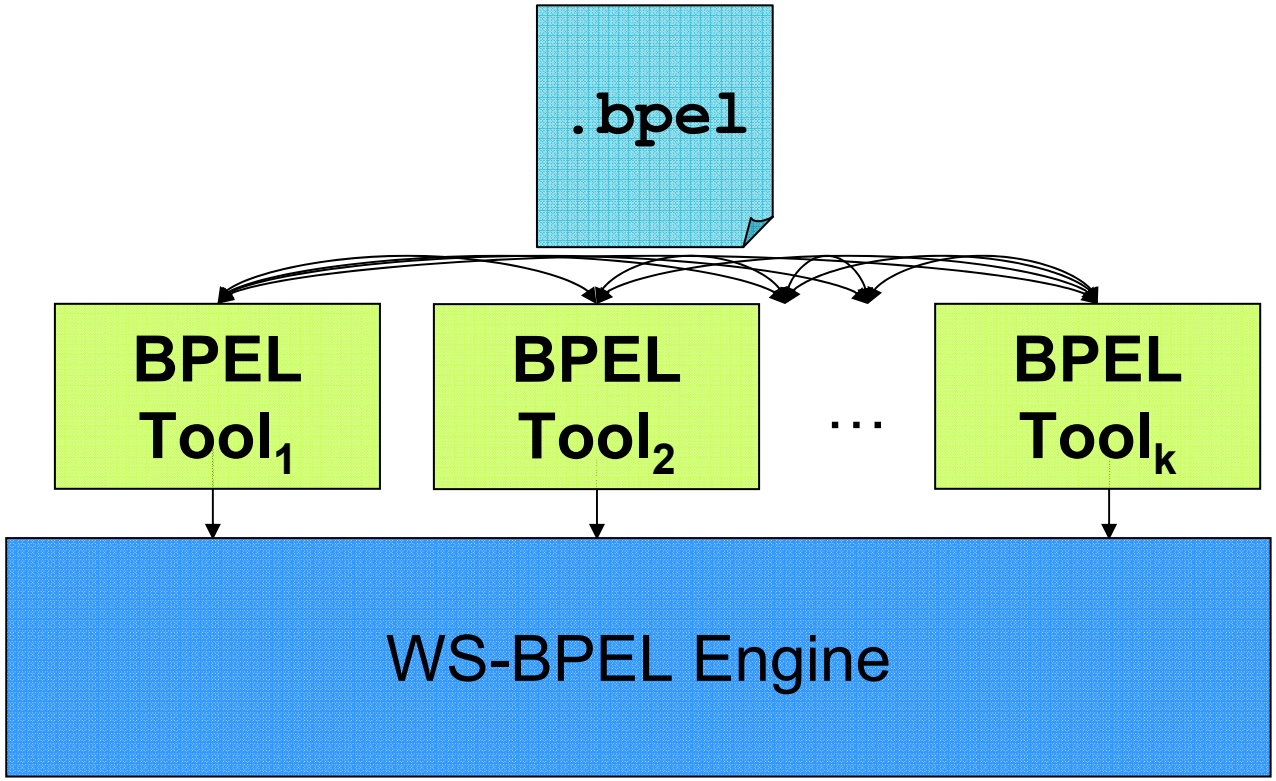
Key Aspect of a Solution



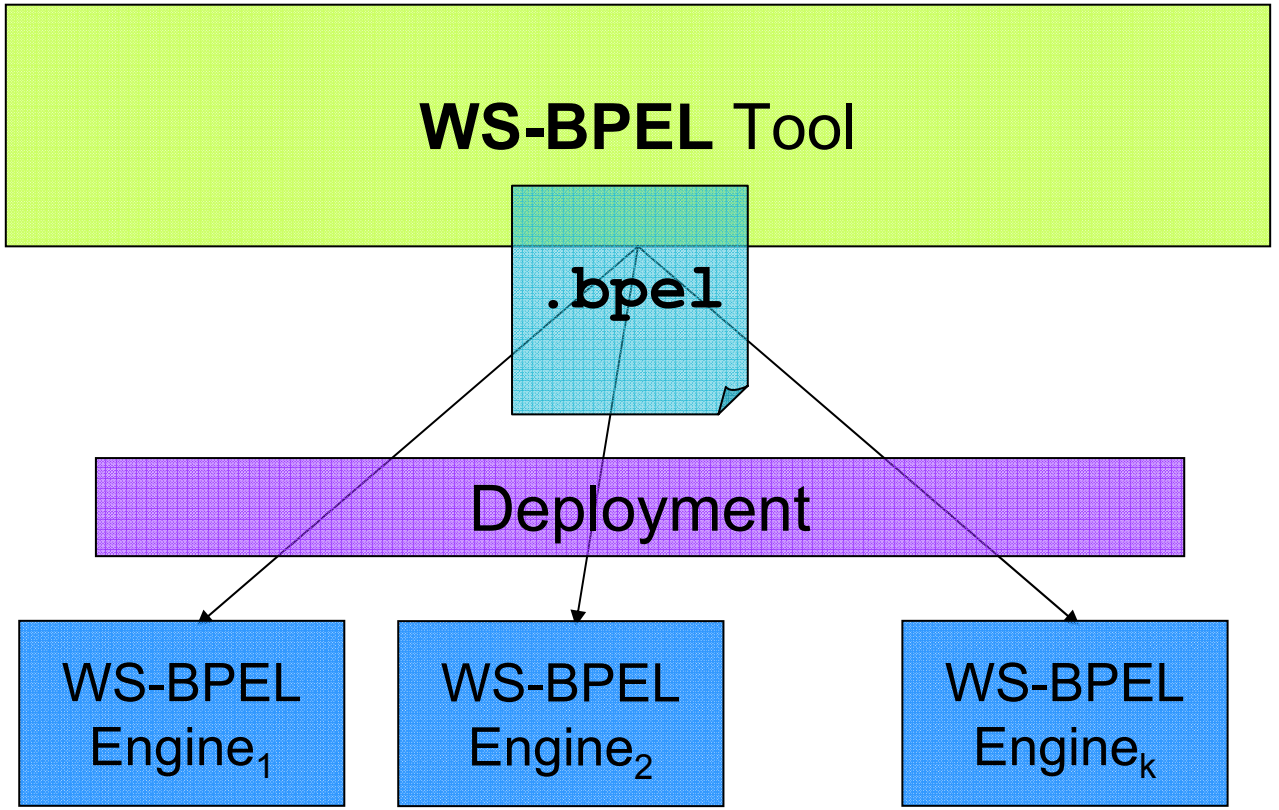
Benefits of WS-BPEL



Tool Interoperability



Common Operational Semantics



Why WS-BPEL?

- WSDL defined Web services have a stateless interaction model
  - Messages are exchanged using
    - Synchronous invocation
    - Uncorrelated asynchronous invocations
- Most “real-world” business processes require a more robust interaction model
  - Messages exchanged in a two-way, peer-to-peer conversation lasting minutes, hours, days, etc.
- WS-BPEL provides the ability to express stateful, long-running interactions

Two-Level Programming Model

- Programming in the large
  - Non-programmers implementing processes
    - Flow logic deals with combining functions in order to solve a more complex problem (such as processing an order)
- Programming in the small
  - Programmers implementing low-level services
    - Function logic deals with a discrete fine-grained task (such as retrieving an order document or updating a customer record)

Web Service Orchestration and WS-BPEL

- Part I – Web Service Orchestration
  - Motivation and Overview
  - Business Process Modeling Styles
- Part II – The WS-BPEL 2.0 Standard
  - WS-BPEL 2.0 Concepts and Language Elements
- Part III – WS-BPEL 2.0 Extensions
  - WS-BPEL Extension for People (BPEL4People)
  - WS-BPEL Extension for Subprocesses (BPEL-SPE)
  - WS-BPEL Extension for Java (BPELJ)
- Part IV – Additional Related Standards
  - Service Component Architecture (SCA)
  - Business Process Modeling Notation (BPMN)

Categorizing Business Processes in BPM

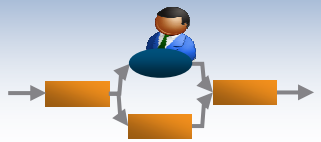
- Anytime:
  - Business processes are at the core of BPM. Therefore, specific characteristics of business processes need to be at the core of any categorization scheme.
- At present:
  - The primary foundation for our studies are business process automation projects
    - Integration-centric and human-centric BPM
    - Based on WebSphere Process Server
    - Sometimes in conjunction with other WebSphere Dynamic Process Edition products
  - The degree of automation of a business process is used as the #1 categorization factor for the twelve business process styles introduced in the next slides
- In the bright future:
  - “Agility enablers” are becoming increasingly important, and will be considered in future updates of this categorization
    - Analytics, rules, service selection, policies: All available through WebSphere BPM.
    - Active content: Available through integration with content management systems (IBM Content Manager, FileNet, WebSphere Portal Content Management, …). Included in this categorization.
    - Events: Available in WebSphere Process Server, WebSphere Enterprise Service Bus, WebSphere Business Events, … Partially included in this categorization.

Twelve "Magic" Business Process Styles

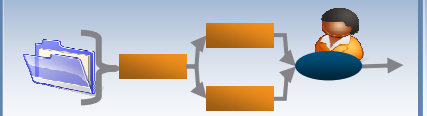
Business Processes with a High Degree of Automation

Fully automated processes ("lights off" workflow, including micro flows)

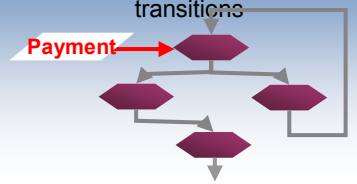
Combination of automation with human tasks for decision making or exception handling



Content-related workflows dealing with documents of various types



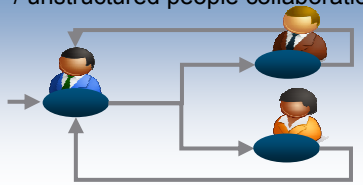
State machines requiring event driven actions and state transitions



Business Processes with a High Degree of Human Interaction

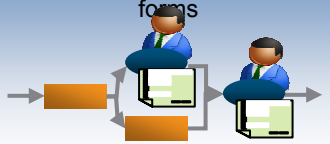
Human-centric workflows supporting work assignment and collaboration

Ad-hoc human tasks for dynamic / unstructured people collaboration



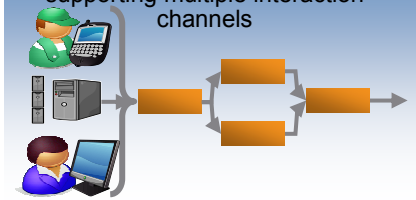
Page flows controlling user interaction

Forms flows present information to the user through electronic forms

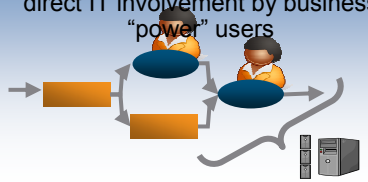


Business Processes that are Independent of the Degree of Automation

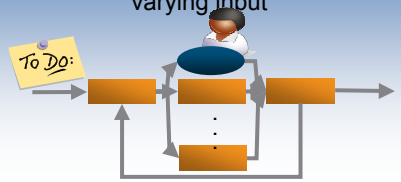
Multi-channel enabled processes supporting multiple interaction channels



Business user workflows are defined and deployed without direct IT involvement by business "power" users



Meta flows – processes that dynamically interpret and adapt to varying input

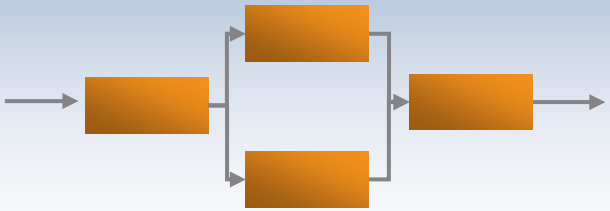


…plus combinations from multiple categories

Business Process Style: Fully Automated Flows

Category: High Degree of Automation

Fully automated processes
("lights off" workflow, including micro flows)

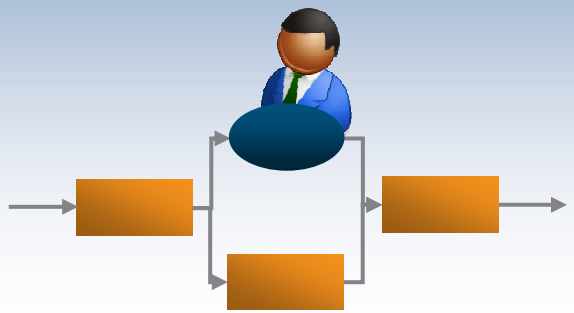


- Key Characteristics
  - No human interaction
  - Decisions are made automatically or retrieved from external systems
  - Unpredictable situations are handled and resolved automatically

Business Process Style: Automation and Human Tasks

Category: High Degree of Automation

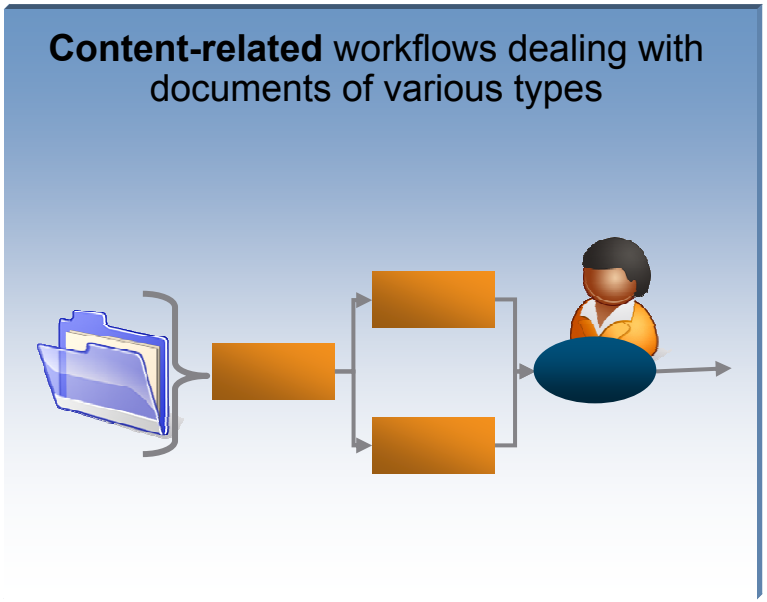
Combination of automation with human tasks for decision making or exception handling



- Key Characteristics
  - There are fully automated paths through the process
  - Human interaction is required for decision making, information collection, or exception handling

Business Process Style: Content-Related Flows

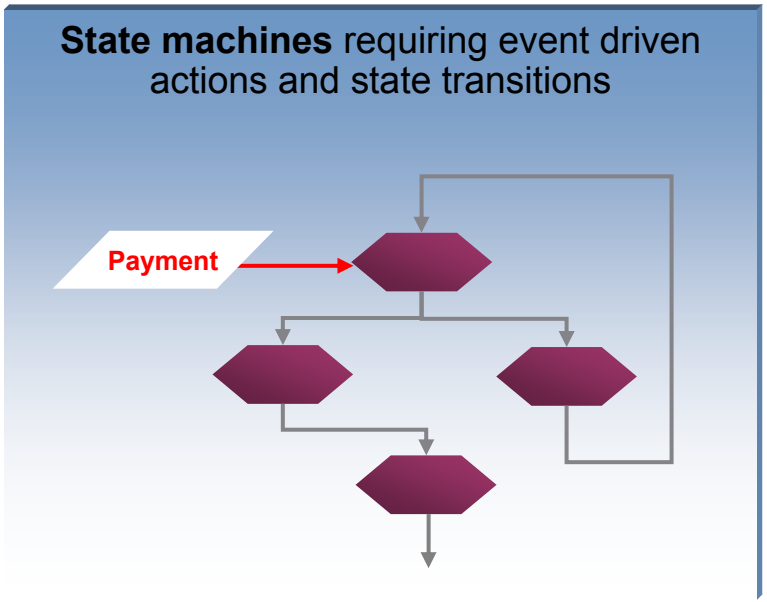
Category: High Degree of Automation



- Key Characteristics
  - Content or document centric
  - Often people are involved to handle processing of documents
  - Often „work baskets“ are being used to allow work force to review, collaborate, or complete documents

Business Process Style: State Machine Flows

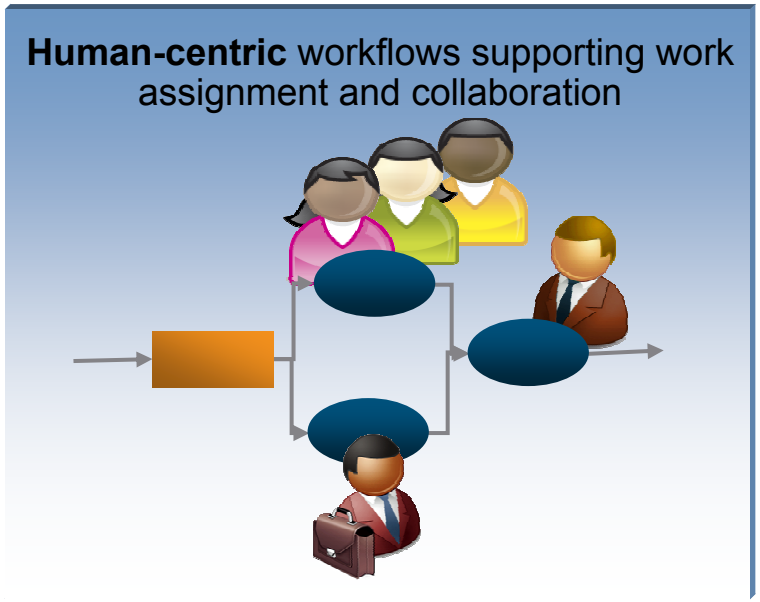
Category: High Degree of Automation



- Key Characteristics
  - State driven control of the workflow
  - Unpredictable order of incoming events that trigger state transitions
  - Actions coupled with state transitions
  - Flexible order of transitions with forward and backward steps

Business Process Style: Collaboration Flows

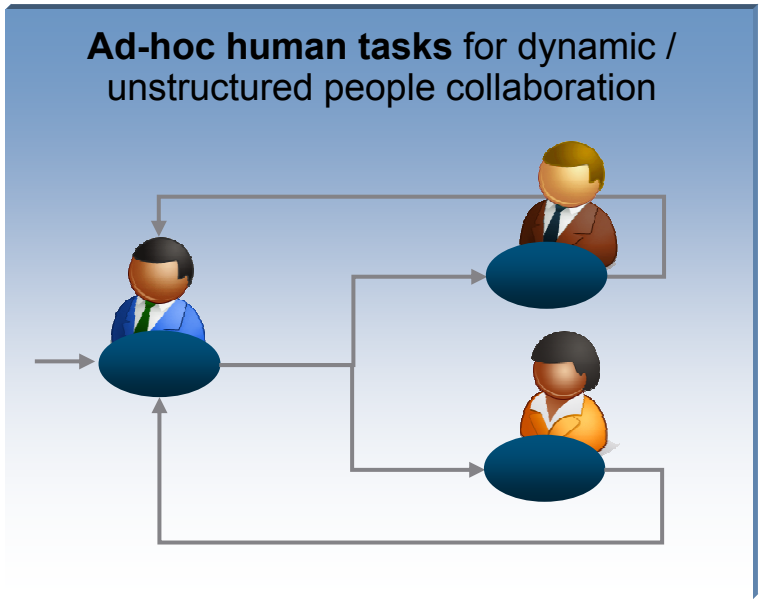
Category: High Degree of Human Interaction



- Key Characteristics
  - Multiple user working together, sometimes in parallel
  - User interaction through „work lists“
  - User interaction through business user client

Business Process Style: Ad-hoc Flows

Category: High Degree of Human Interaction

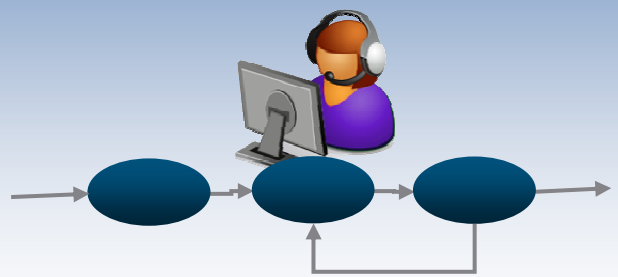


- Key Characteristics
  - Process not fully specified in advance
  - Activities are not predictable
  - Always based on human interaction
  - Highly dynamic
  - User interaction through business user client

Business Process Style: Page Flows

Category: High Degree of Human Interaction

Page flows controlling user interaction

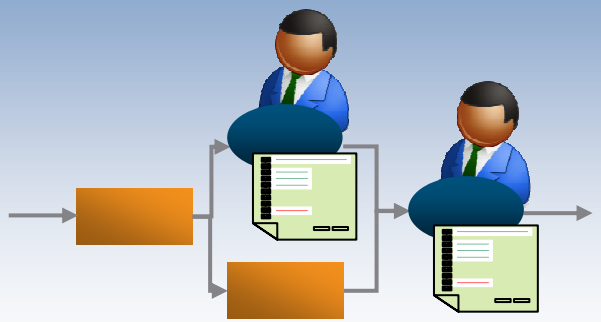


- Key Characteristics
  - User is guided through a sequence of activities in a dialog
  - Single user flow
  - Always based on human interaction
  - Next steps are controlled by the flow

Business Process Style: Forms Flows

Category: High Degree of Human Interaction

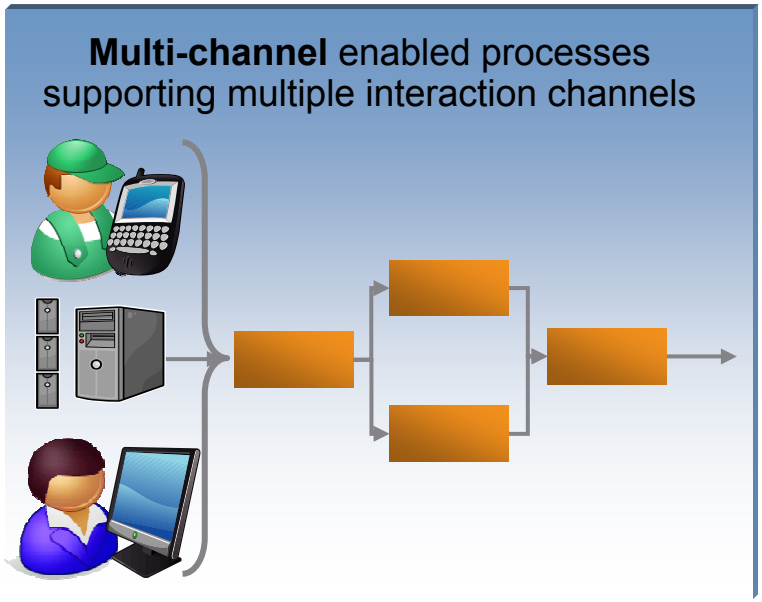
Forms flows present information to the user through electronic forms



- Key Characteristics
  - People interacting with processes are using forms
  - Forms presented to the user look familiar to forms used on paper
  - Combines manual as well as automated steps for process completion

Business Process Style: Multi-Channel Flows

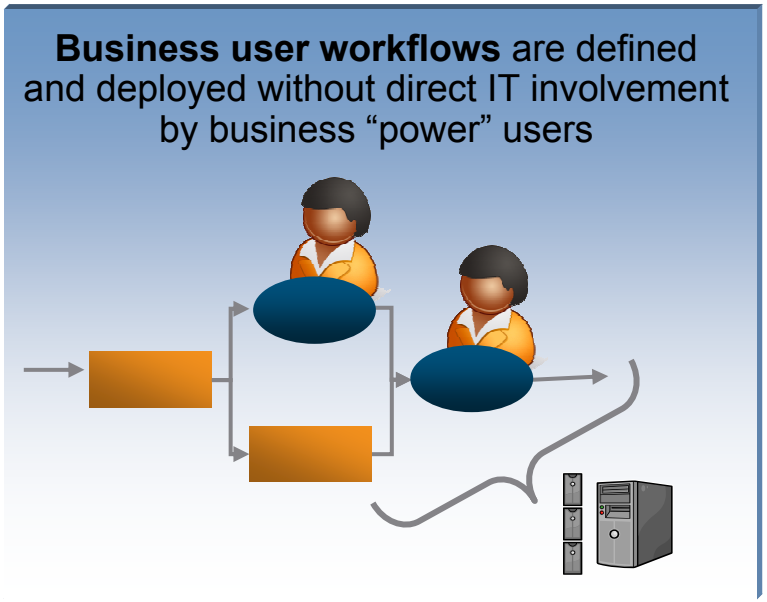
Category: Independent of the Degree of Automation



- Key Characteristics
  - Variety of different input or interaction channels (e.g. Web, phone, e-mail, mail, etc. …)
  - Similar end user experience and same processing for all channels
  - All other process types apply for further processing

Business Process Style: Business User Workflows

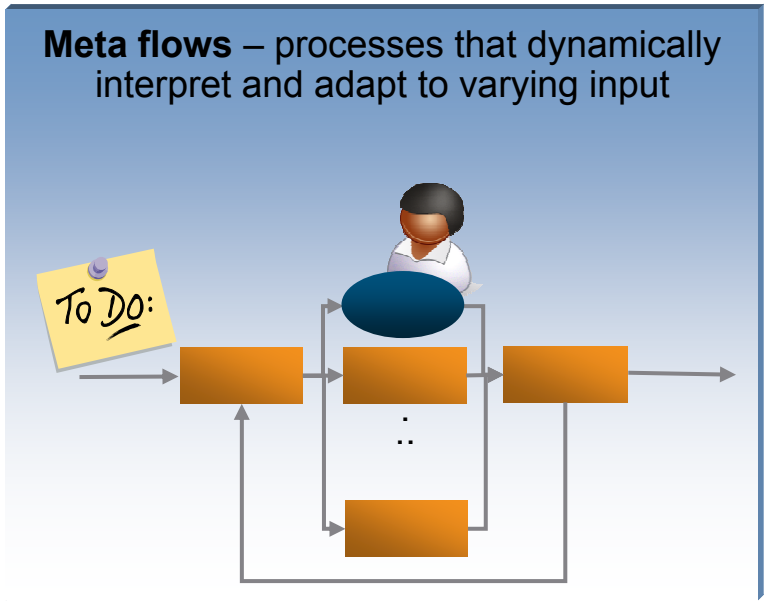
Category: Independent of the Degree of Automation



- Key Characteristics
  - Processes are typically simple in their nature, may include manual and automated steps
  - Process definition as well as deployment by “power” business users without direct IT involvement
  - Activity implementations provided by IT in ‚repository-like‘ tool boxes for use by business users

Business Process Style: "Meta" Flows

Category: Independent of the Degree of Automation

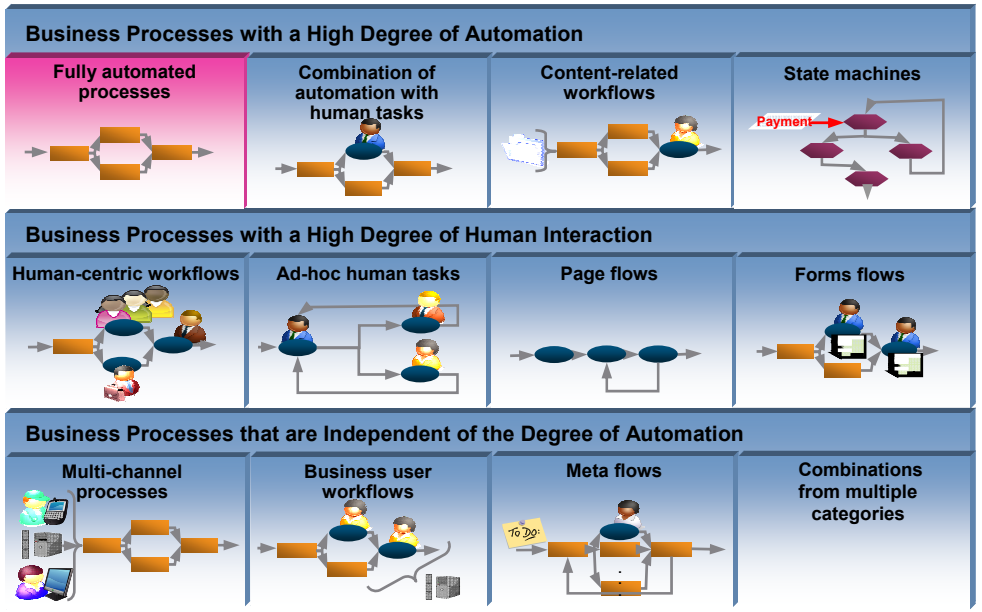


- Key Characteristics
  - Parameterized generic process model that is controlled at execution by information provided at process start or through external events
  - Reacts to external events dynamically, based event payload
  - Allows backward and forward jumps in the process logic

Real-Life Example: Frequent Flyer Program

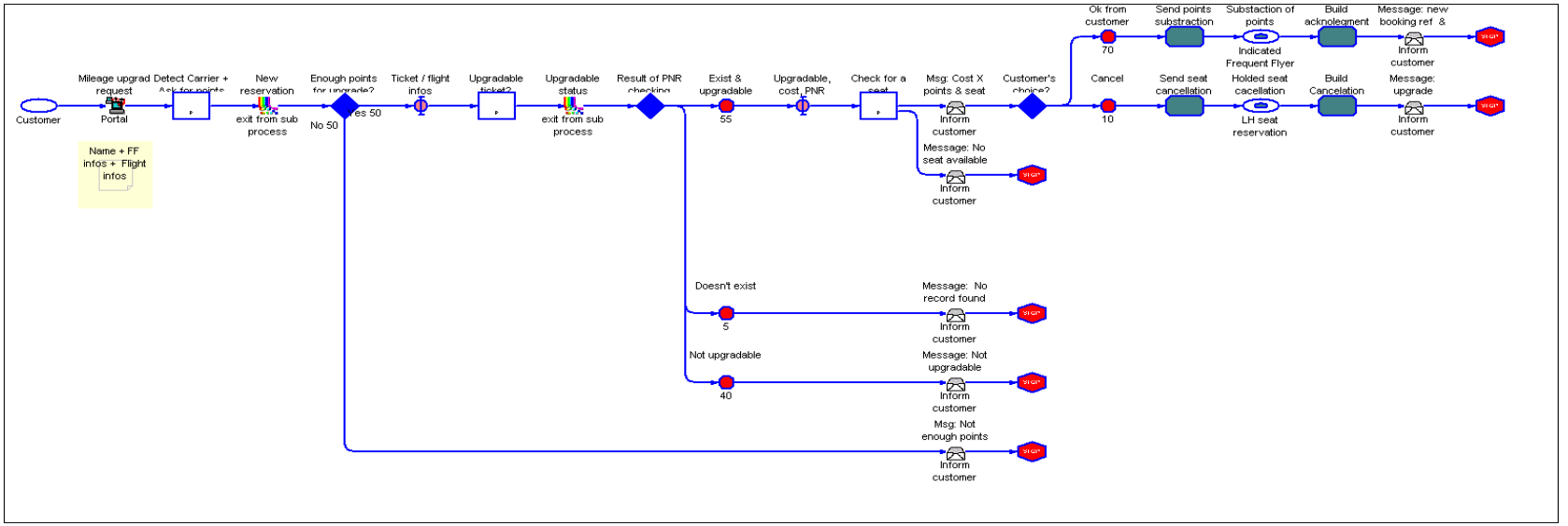
Overview

- Customer Scenario
  - This “Mileage Upgrade” project handles request from customers using an airline alliance’s mileage points upgrade awards offering
  - Offers customers the possibility to upgrade a booked flight using frequent flier miles for any of the airline alliance carriers’ frequent flier programs
  - The process utilizes different services in order to check for free seats, amount of miles, etc.



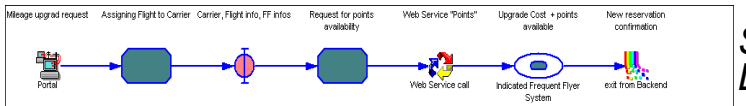
Real-Life Example: Frequent Flyer Program

Sample Business Process

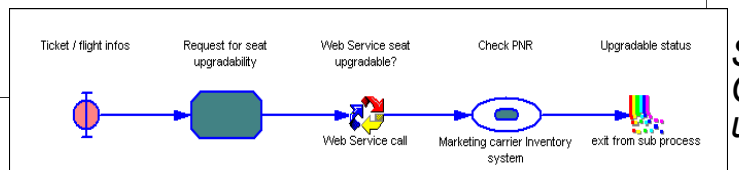


Main process: Handle seat upgrade request

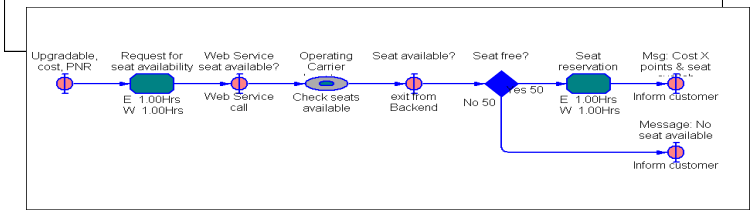
- Fully automated flow
  - Information is received by and passed to other systems
- WebSphere Portal integration
  - Portal as one entry point for all airlines of the airline alliance
  - In the first phase, one out of multiple frequent flyer programs has been integrated



Sub-process: Detect carrier and points



Sub-process Check ticket upgradability

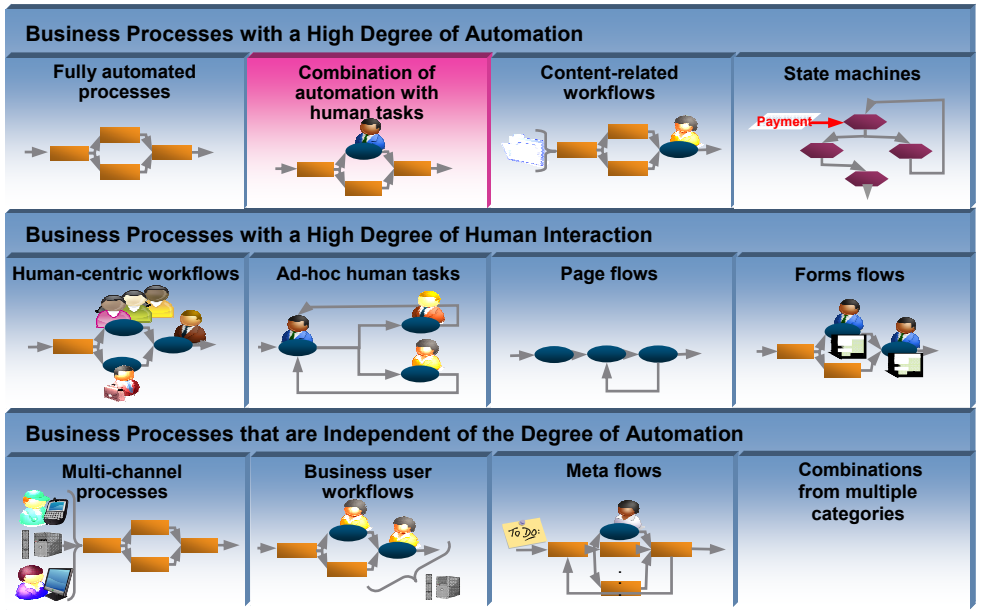


Sub-process Check seat availability

Real-Life Example: Banking Customer Service

Overview

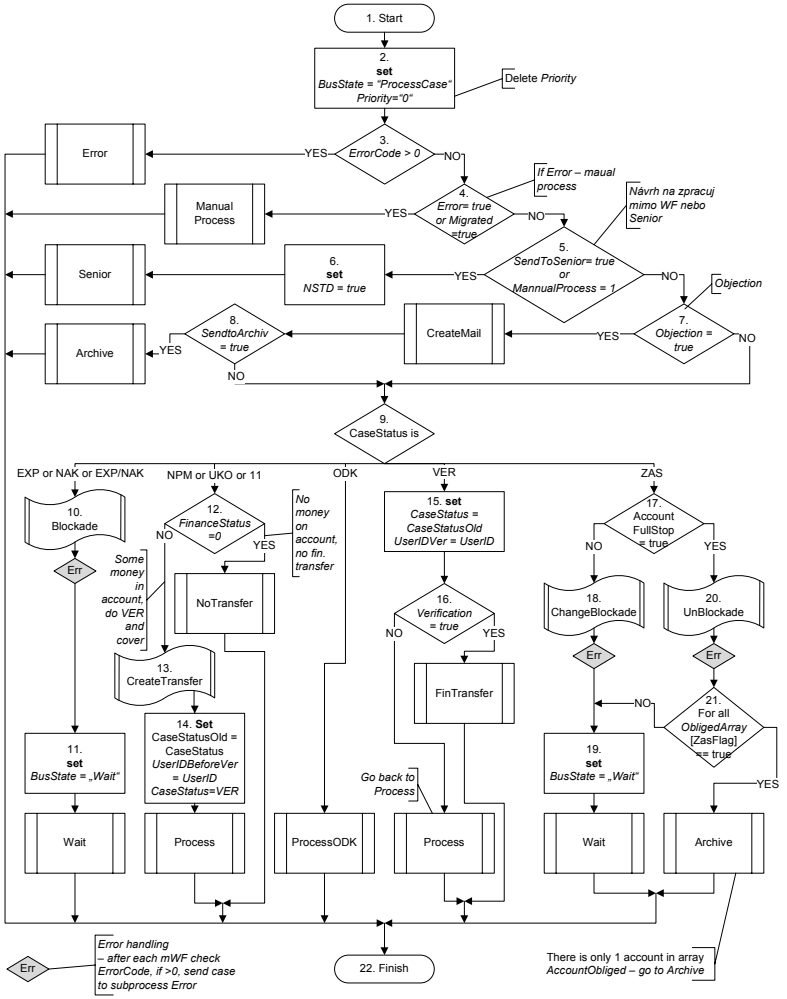
- Customer Scenario
  - Enterprise wide workflow implementation in a bank
  - Consists primarily of long-running BPEL processes with human tasks



Real-Life Example: Banking Customer Service

Sample Business Process

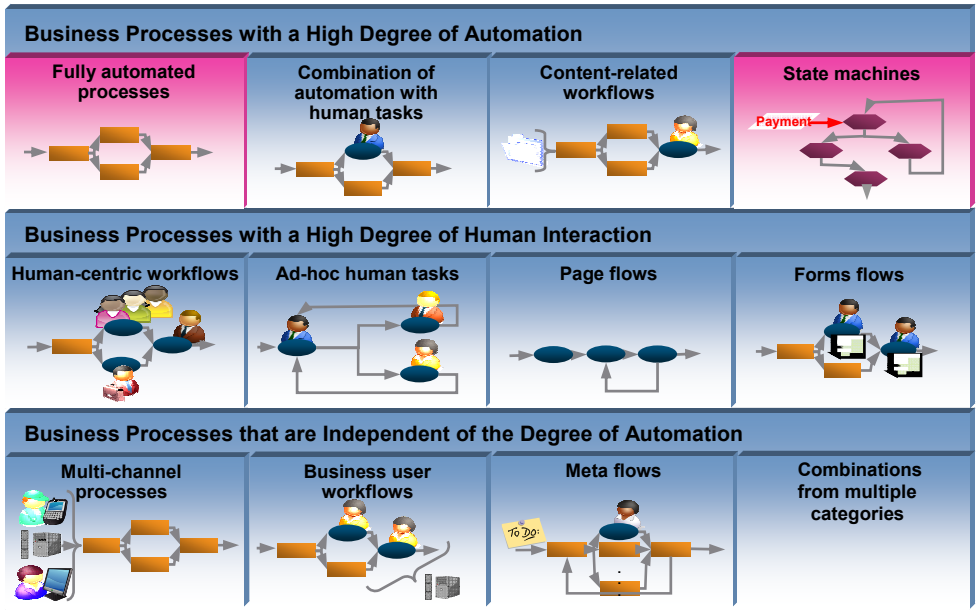
- Combination of automation with human tasks
  - Tasks assignment done with LDAP containing organizational hierarchy
  - Tasks are used for normal processing and error handling
  - Task selection criteria: Up to 10 custom properties



Real-Life Example: Telco Provisioning

Overview

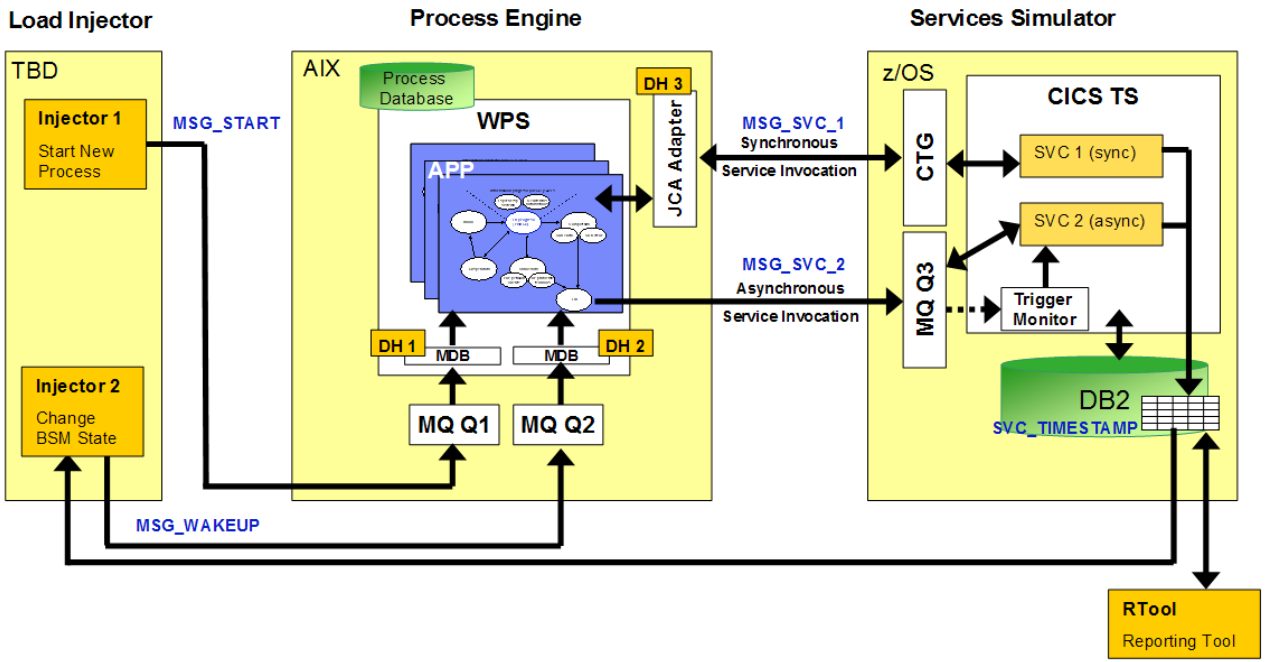
- Customer Scenario
  - Customer order processing
  - BPM project replacing an existing provisioning system
  - Fully automated process using business state machines as well as long-running and short-running processes



Real-Life Example: Telco Provisioning

System Design

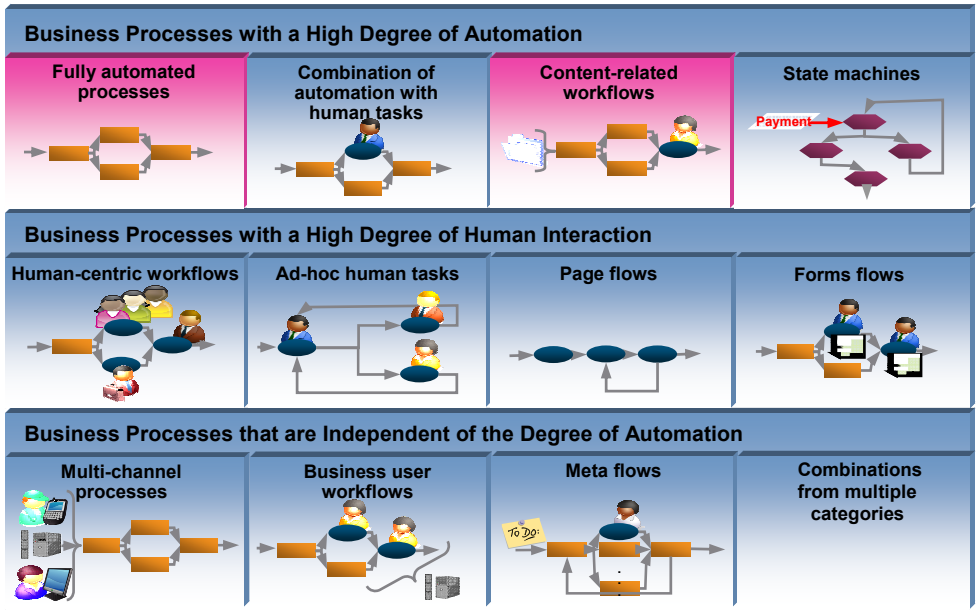
- Fully automated customer order process
  - Is a business state machine
  - Starts other business state machines
  - Starts other long-running business processes
  - Starts other short-running business processes
  - Customer order processing is fully event-driven



Real-Life Example: Health Insurance Claims

Overview

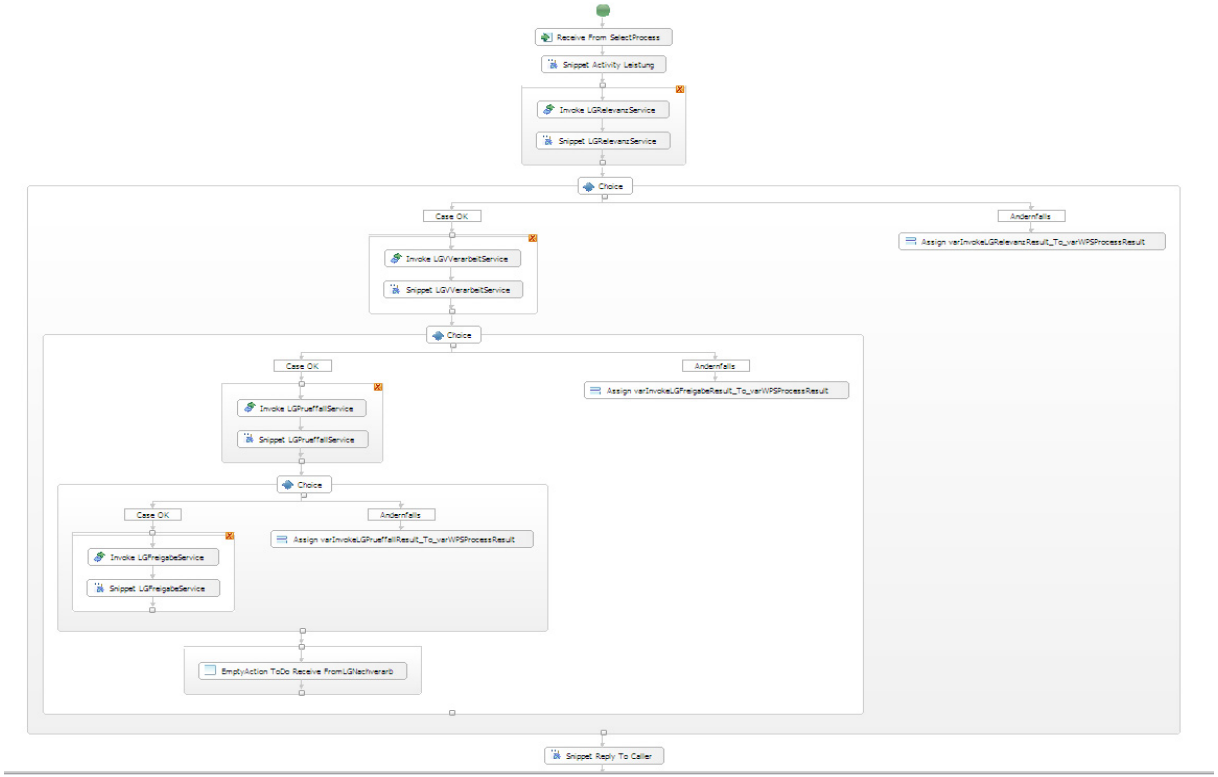
- Customer Scenario
  - The project handles all invoices which arrive at this health insurance company
  - The invoices are automatically scanned and processed
  - If the input documents can be properly read, the processing is fully automated
  - Otherwise the request has to be processed manually
  - Human interaction is currently not implemented
  - The currently implemented scenario is “fully automated”, while the overall scenario can be classified as “content related”



Real-Life Example: Health Insurance Claims

Sample Business Process

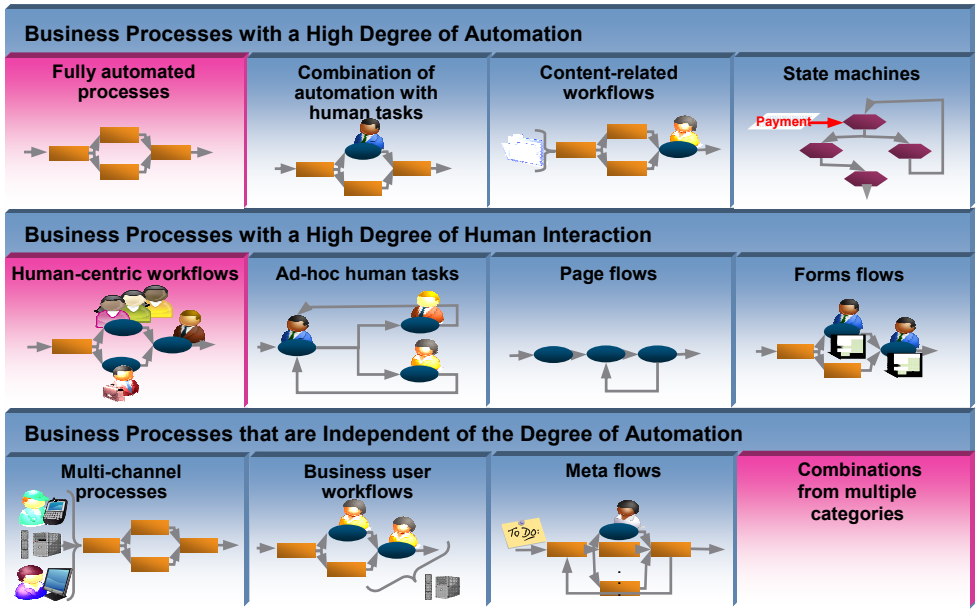
- Content related flow
  - Based on documents
  - Fully automated
  - Not automatable part is routed to an employee
- Other specifics
  - Integration to existing back-end (COBOL on z/OS) via SOAP/MQ
  - Configurable retry and error handling solution was required and implemented as BPEL
  - Better alternative: Out-of-the-box solution on e.g. SCA references



Real-Life Example: Banking Mortgage Handling

Overview

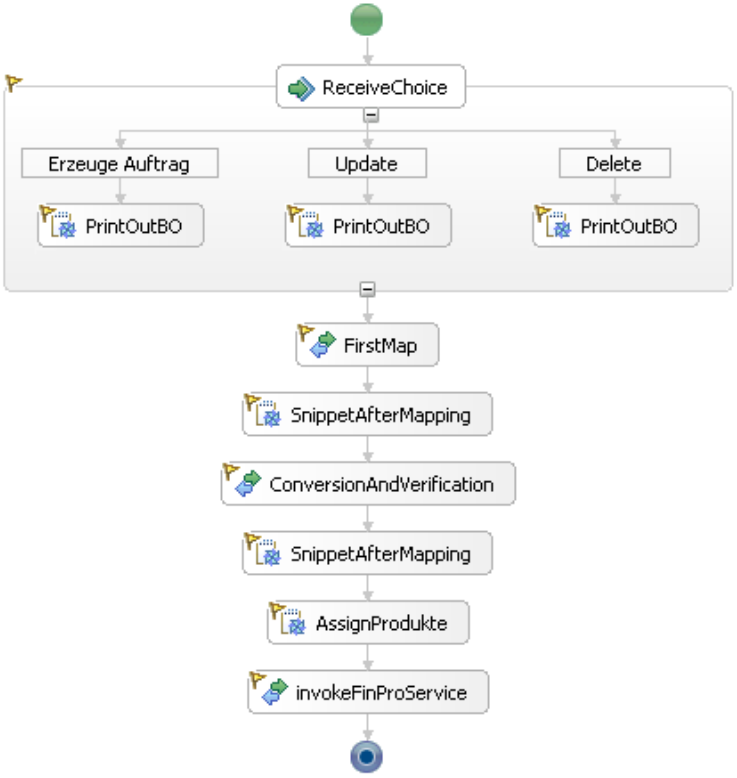
- Customer Scenario
  - Application to handle mortgage requests in the back office of a bank
  - The front office uses a Siebel system to capture the required information from the customer
  - Once this is done, the application is triggered and back office employees can start to work on the mortgage request.
  - The request is enriched with additional information, reviewed and once approved, passed back to the front office
  - In the case of missing information or denial, the request is also passed back

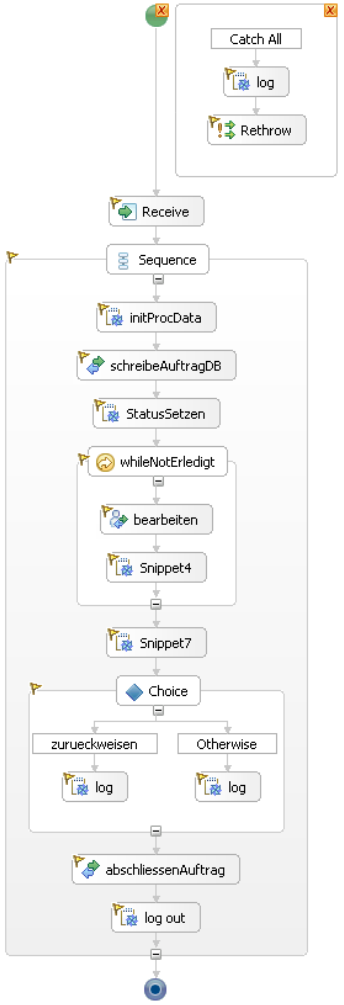


Real-Life Example: Banking Mortgage Handling

Sample Business Process

- Automated flow
  - No human interaction
  - Additional data is retrieved from external systems
- Collaboration flow
  - Human interaction is used to e.g. add additional information about the client
  - Human interaction is used for approving the mortgage request
  - Multiple users work in parallel (employees, reviewer, approver)
  - Users work on worklists

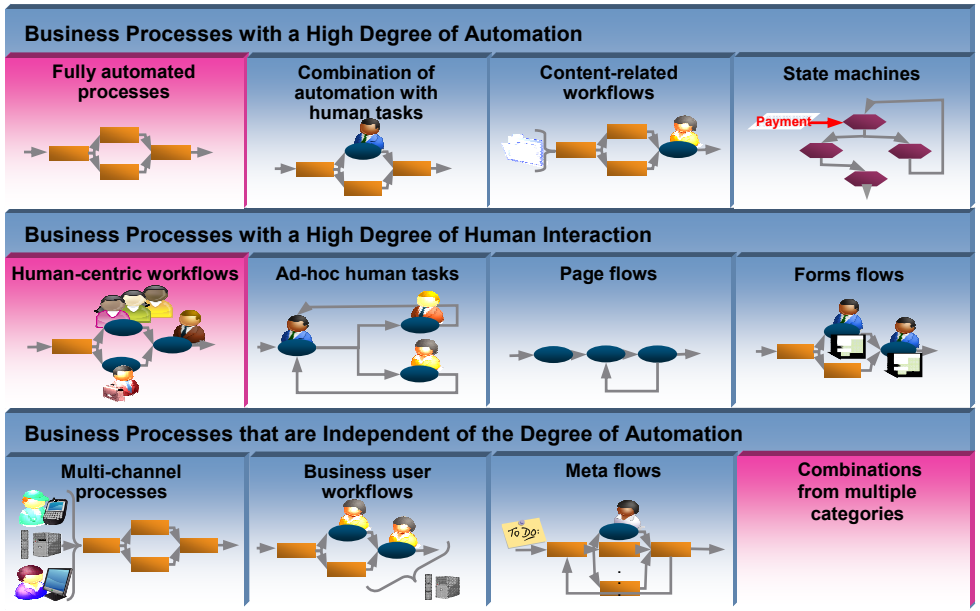




Real-Life Example: Customs Duty Management

Overview

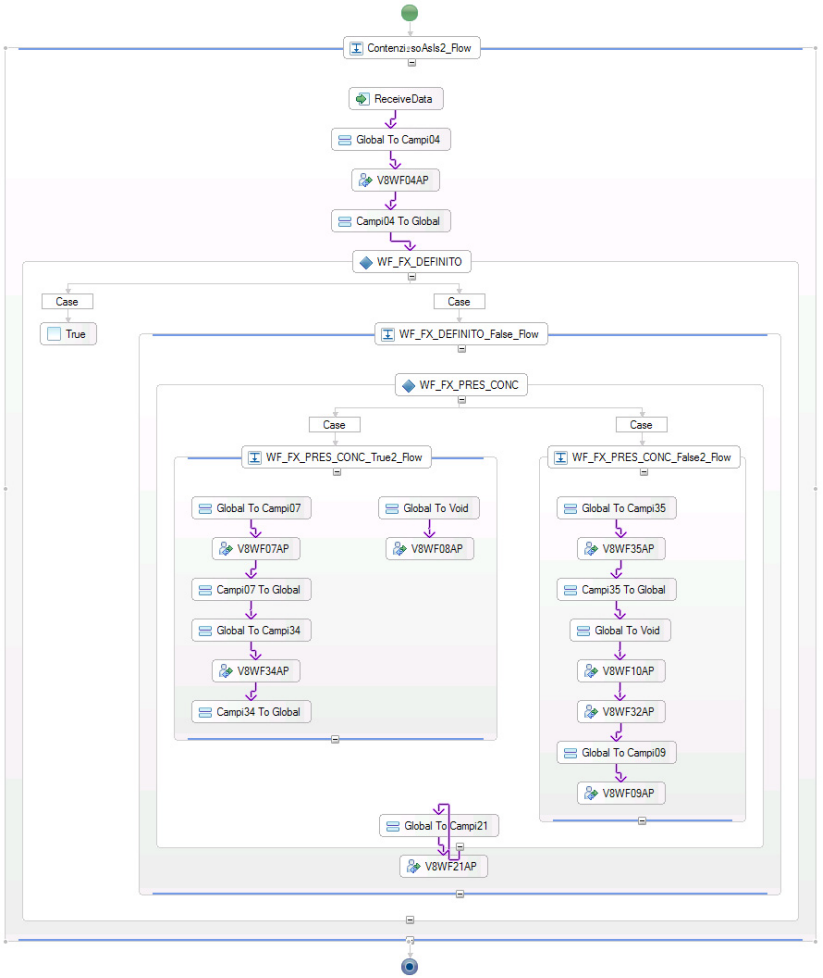
- Customer Scenario
  - Users of this application are clerks of a government agency processing cases that are related to goods that cross the border



Real-Life Example: Customs Duty Management

Sample Business Process

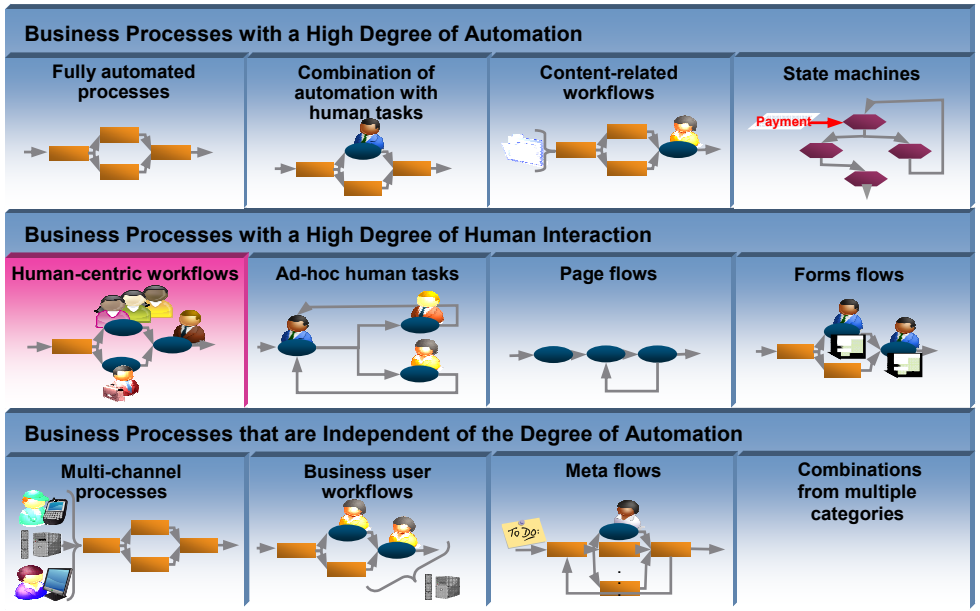
- Human-centric flows
  - Users are divided into 10 roles
  - 15 flows, 5 to 6 are main flows
  - Each flow has an average of 4 to 5 steps
  - 2 flows are completely automatic (without human intervention)
  - The duration of a flow varies from a few minutes to 5 days
- Other specifics
  - Integration of an Oracle DB and LDAP



Real-Life Example: Direct Insurance

Overview

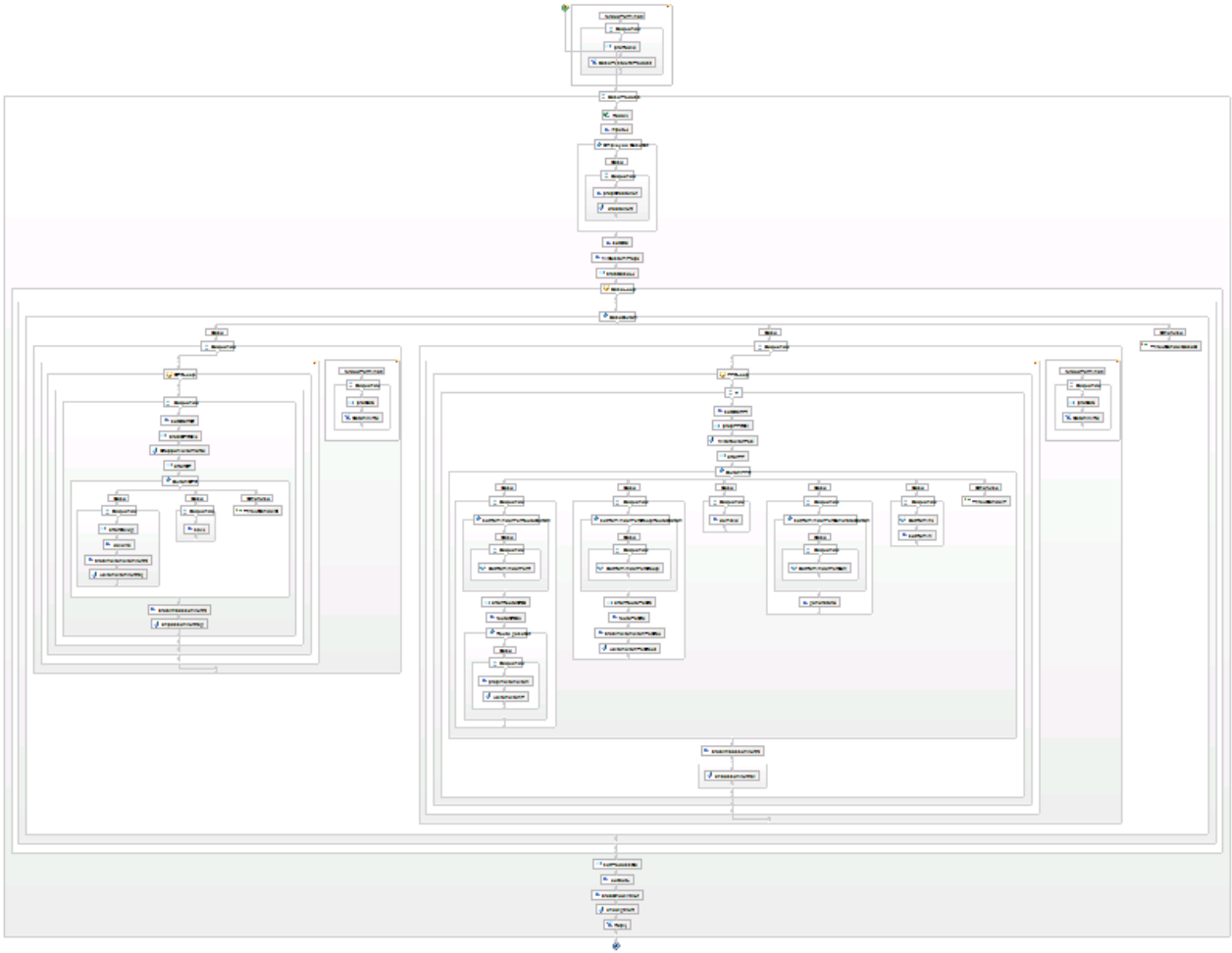
- Customer Scenario
  - Requests for insurance coverage from customers are routed to the system and show up as "tasks" in the inbox of managers.
  - Custom business user client application for managers and employees
  - Filtering of processes and tasks based on custom properties
  - Managers distribute work (tasks) to their employees



Real-Life Example: Direct Insurance

Sample Business Process

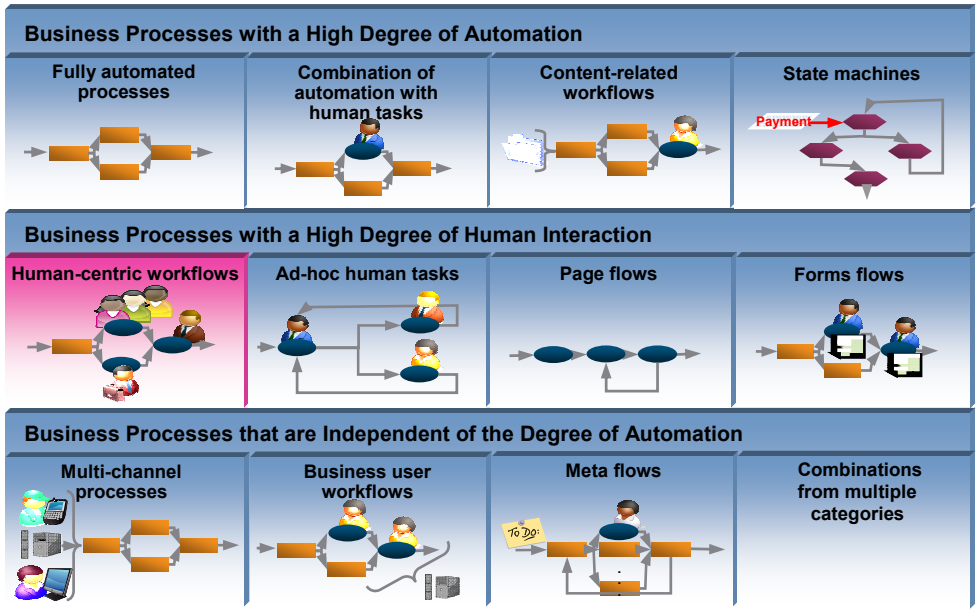
- Human-centric flow
  - Tasks are distributed to the employees
  - Tasks can be routed among the employees (collaboration)
  - No automation to process a task



Real-Life Example: Human Resources Management

Overview

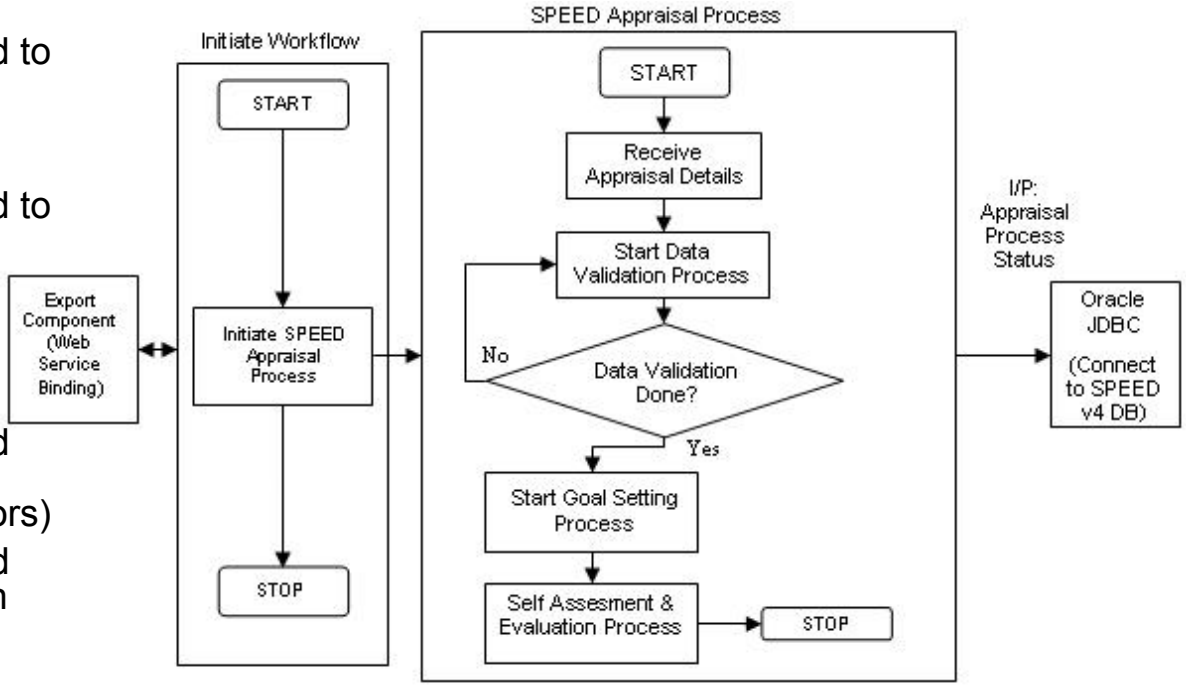
- Customer Scenario
  - This application implements the appraisal process for employees of a large systems integrator (System for Performance Evaluation and Employee Development)
  - It provides opportunities for personal and career development and brings all HR associates under a single strategic umbrella
  - Most importantly, it gives supervisors and associates an equal opportunity to express themselves under structured conditions



Real-Life Example: Human Resources Management

Sample Business Process

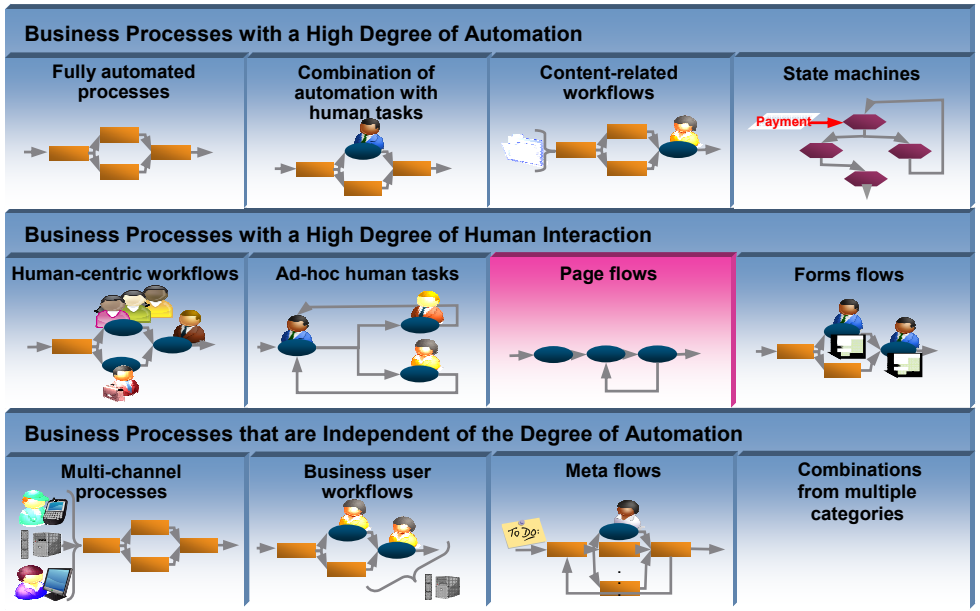
- Collaboration flow
  - Human interaction is used to provide additional information like employee data, goals, etc.
  - Human interaction is used to decide if data is complete and valid
  - Human interaction is used for approving the goals
  - Human interaction is used for handling infrastructure problems (by administrators)
  - Human interaction is used for distributing information (FYI tasks)
  - Multiple users work in parallel (employees, supervisors, etc.)
  - Users work on worklists (supervisors are responsible for multiple employees)



Real-Life Example: Telco Call Center

Overview

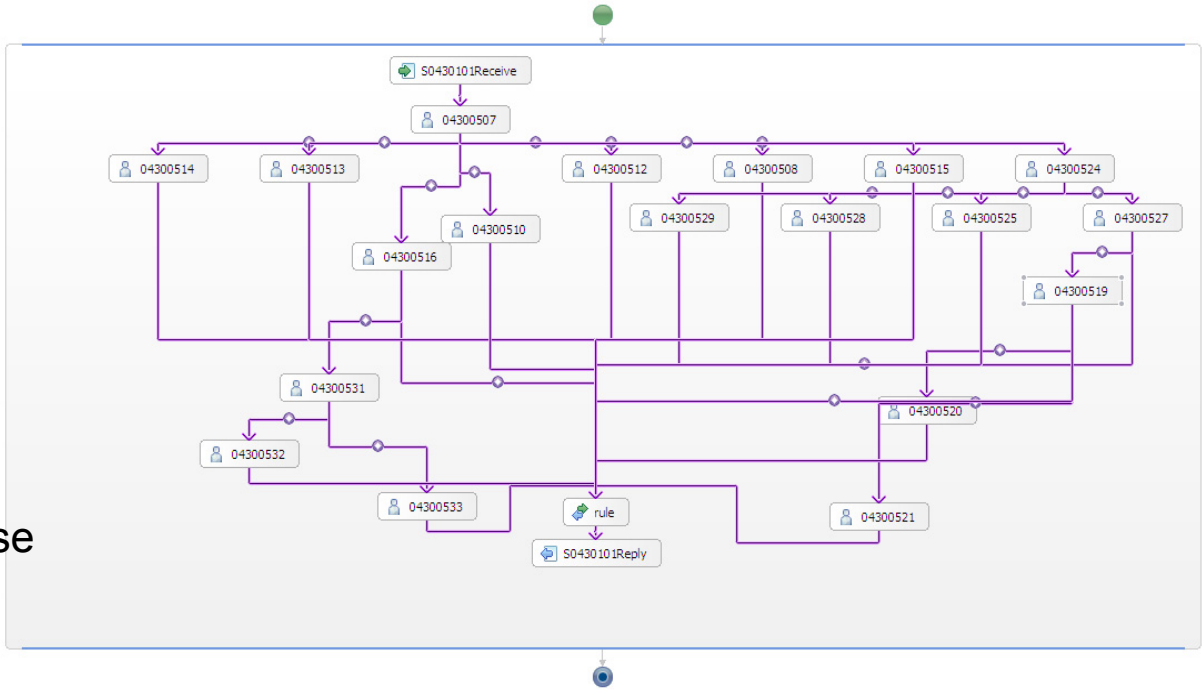
- Customer Scenario
  - This project handles incoming calls for a Call Center
  - The implemented page flow displays a screen with predefined questions that a call center agent has to ask the caller
  - Each page is implemented as a human task. The large number of pages and processes requires the use of sub-processes.
  - The processes consist only of human tasks and some assign and snippet activities



Real-Life Example: Telco Call Center

Sample Business Process

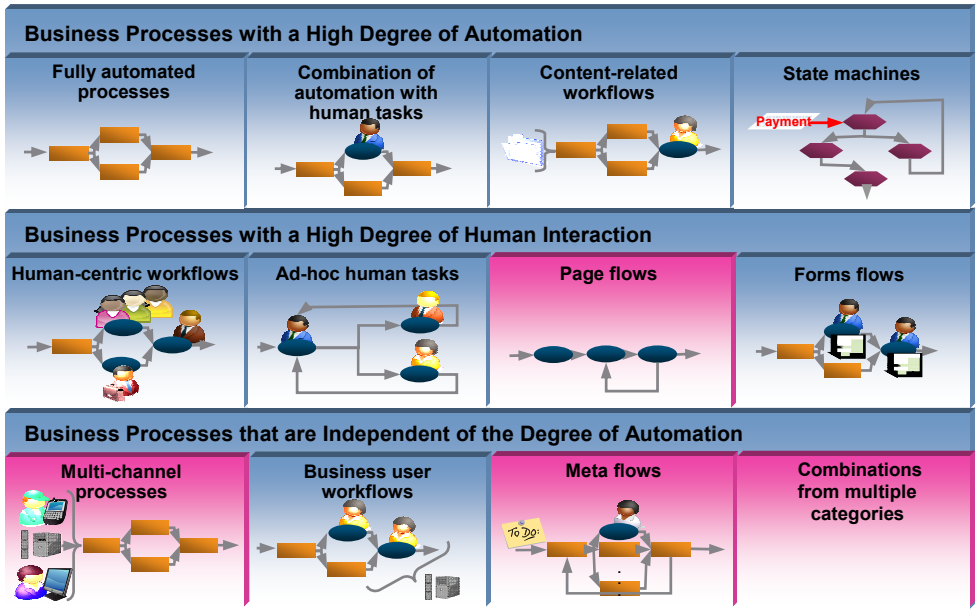
- Page flow
  - Each page is a human task
  - Versioning with late binding is required
  - Large application with hundreds of tasks in about 50 modules
  - Integration to a customer database done within the servlet
  - Specific performance requirements, e.g., sub-second response times



Real-Life Example: Telco Order Management

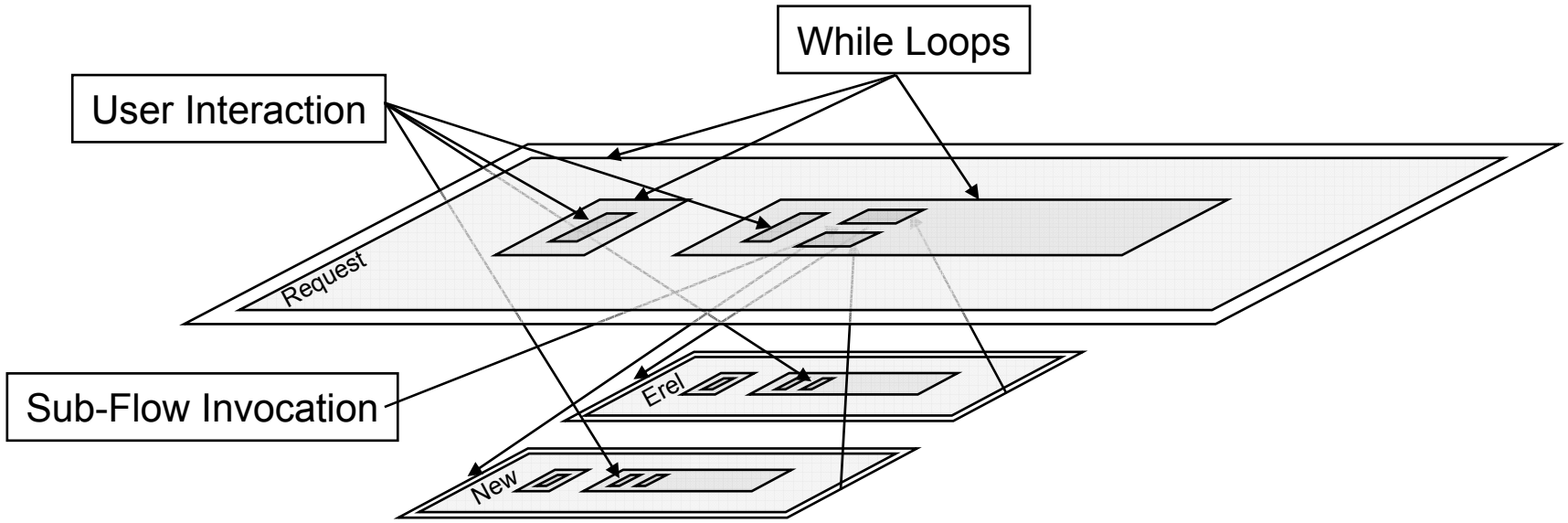
Overview

- Customer Scenario
  - Customers order new phone services or apply for relocation of existing connections by using a web browser or calling the service center
  - If the web browser is used, the user passes several different pages guiding him through the business process
  - There is a BPEL based main process which calls several sub-processes and offers the possibility to dynamically jump back and execute previously done steps again



Real-Life Example: Telco Order Management

Business Process Design



- Parent process keeps control and calls sub-processes
- Sub-processes perform the action on the backend systems
- Each process can be cancelled at any time

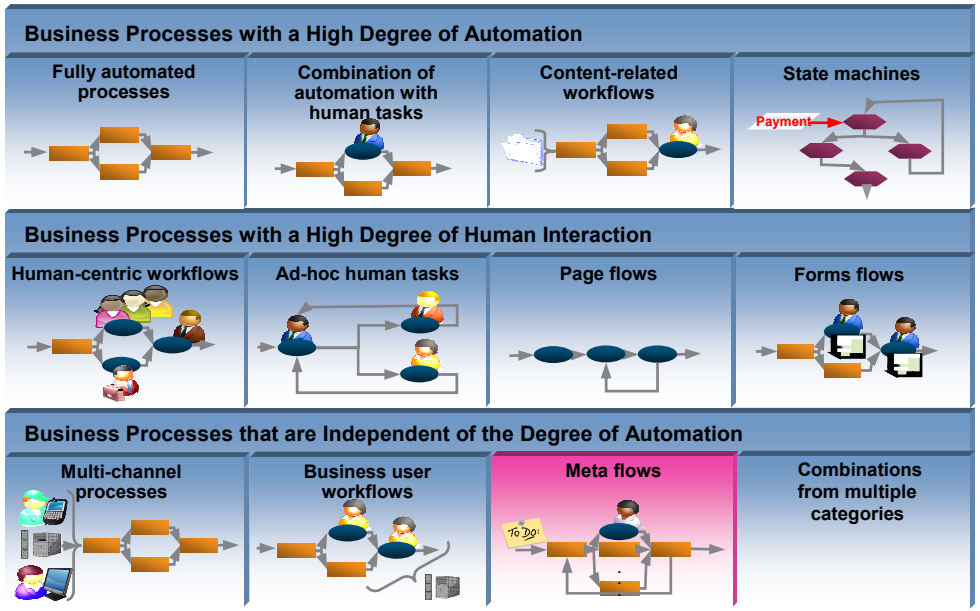
Real-Life Example: Telco Order Management
Details

- Page flow
  - Each single interaction steps in the main parent process are represented by different pages which are presented to the user
  - BPEL processes determine the flow of the different pages
- Meta flow
  - The parent process triggers different sub-processes to perform physical work on the backend systems
  - Depending on the result of those sub-processes the parent process either continues the work or jumps back and takes another execution path
- Multi Channel
  - The BPEL process is started either through Web services calls or through Web client calls

Real-Life Example: Payments Clearing House

Overview

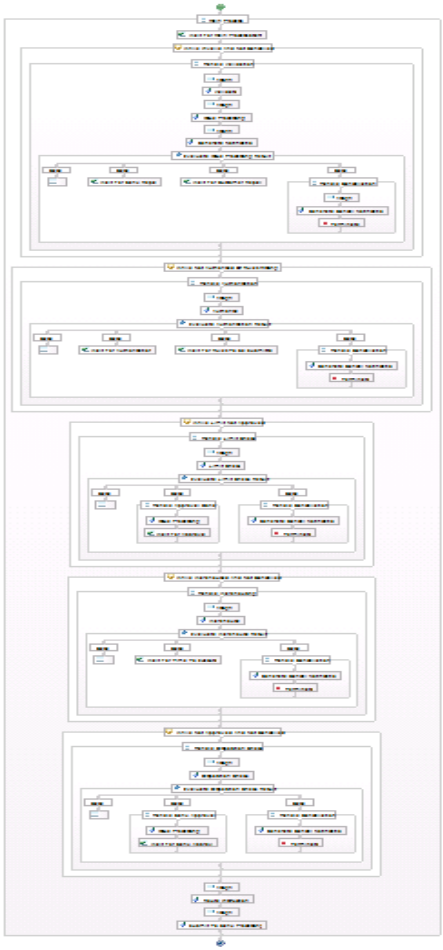
- Customer Scenario
  - The objective of this project was to develop a reference SOA architecture for payment processing in the banking sector (clearing house for inter-bank payments)
  - The activities involved in processing a payment are choreographed using a BPEL based business process
  - The customer is offering these processes as services to other banks



Real-Life Example: Payments Clearing House

Sample Business Process

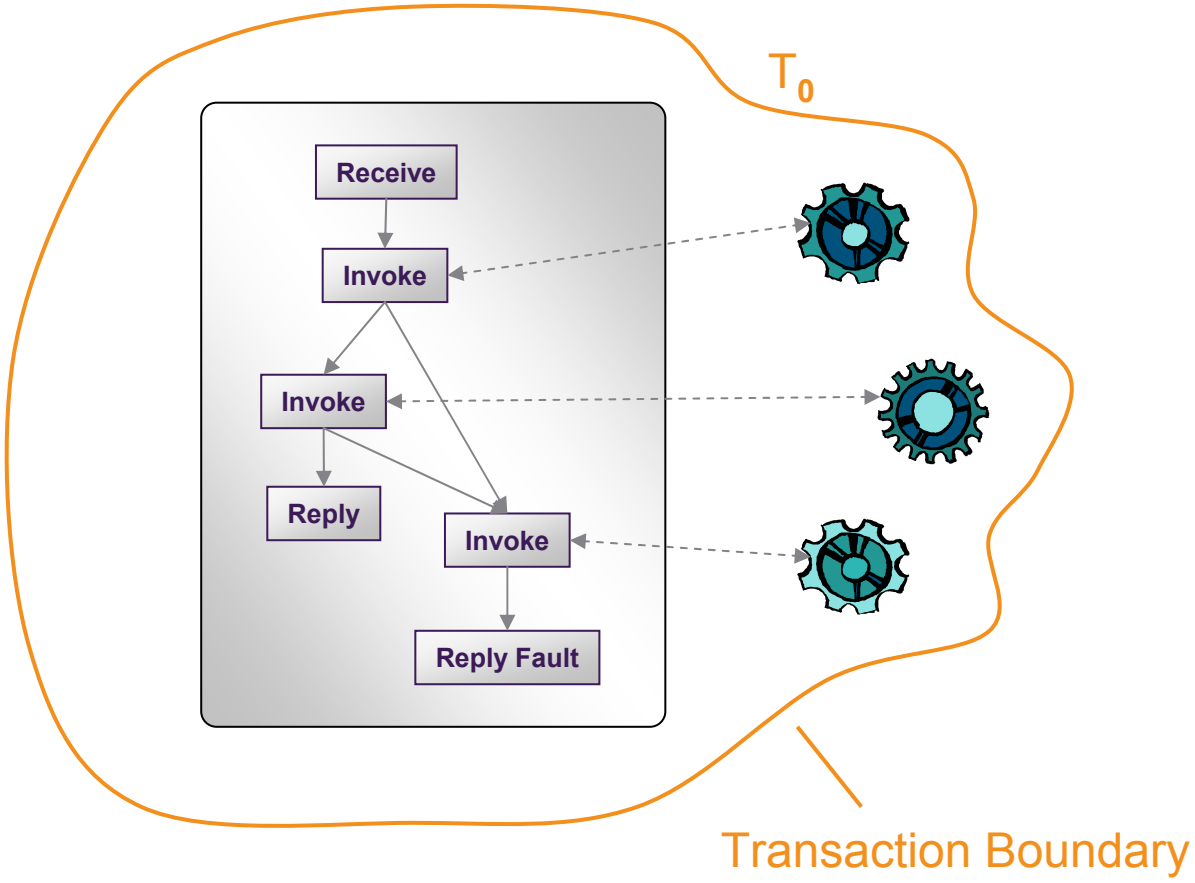
- Meta flow
  - BPEL process consists of several while loops
  - Several receive activities in the course of the process are used to make the process listen for outstanding external requests
  - Depending on the result of those external requests, the process loops back and eventually another execution path is processed
  - The data is passed to the system as XML files/transactions through WebSphere MQ
  - The process reacts on external events/requests
  - Depending on the content of these requests, the process jumps back and takes another execution path



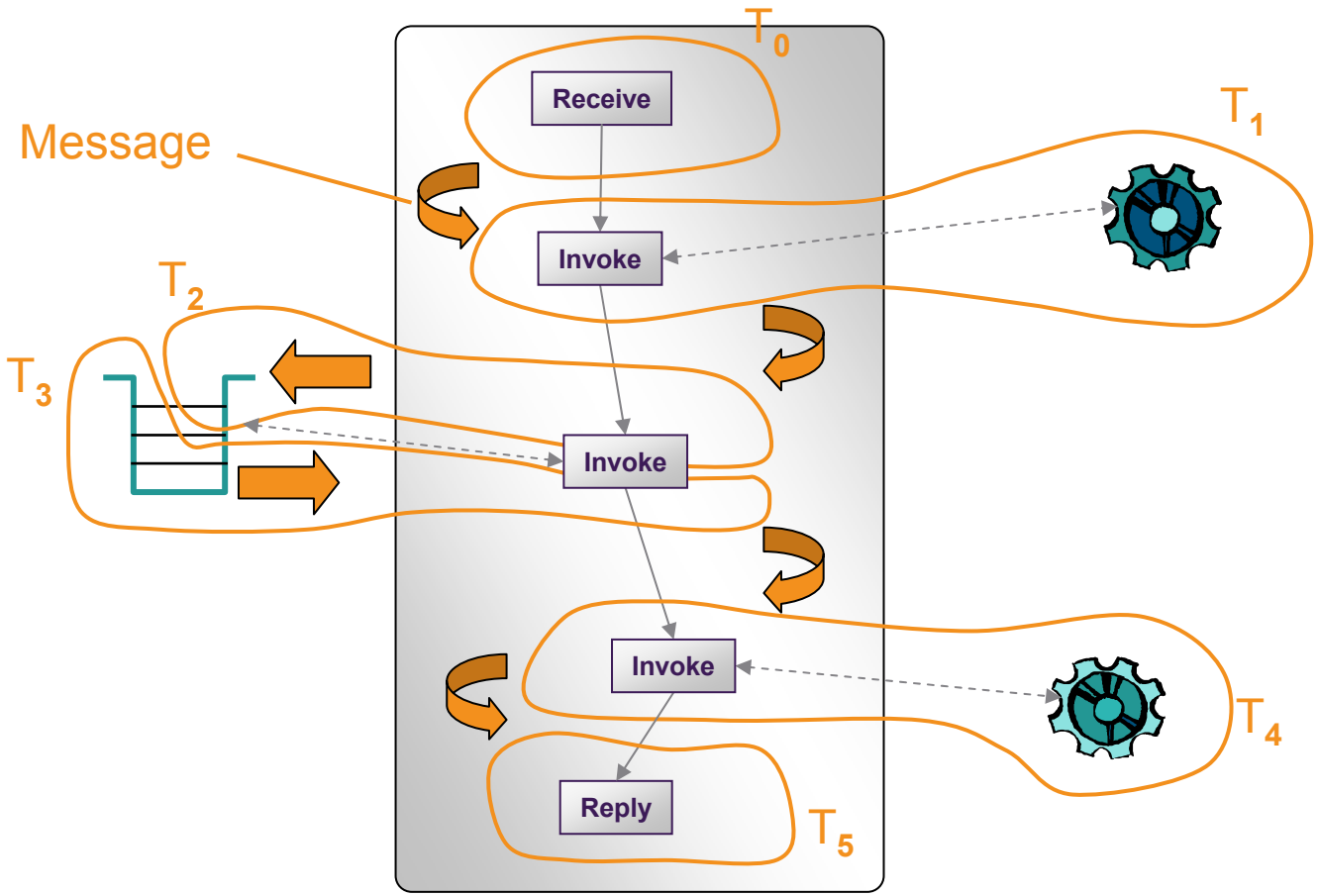
Transaction Considerations

- Microflow
  - Non-interruptible
  - Single unit of work
  - Not persistent → No database / message queueing system involved
  - High throughput
- Long-running process
  - Interruptible
  - Persistent
    - State stored persistently in database
    - Usage of persistent messaging to reliably hold the navigation information of the flow
  - Transacted execution
  - Supports all types of activities (including those not supported in microflows, e.g. human interaction and asynchronous invocations)

Microflow Transaction



Long-Running Process Transactions



Web Service Orchestration and WS-BPEL

- Part I – Web Service Orchestration
  - Motivation and Overview
  - Business Process Modeling Styles
- Part II – The WS-BPEL 2.0 Standard
  - WS-BPEL 2.0 Concepts and Language Elements
- Part III – WS-BPEL 2.0 Extensions
  - WS-BPEL Extension for People (BPEL4People)
  - WS-BPEL Extension for Subprocesses (BPEL-SPE)
  - WS-BPEL Extension for Java (BPELJ)
- Part IV – Additional Related Standards
  - Service Component Architecture (SCA)
  - Business Process Modeling Notation (BPMN)

Web Services – Business Process Execution Language (WS-BPEL) 2.0

OASIS Standard

April 11, 2007

http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

BPEL Objectives

- Standard language for expressing business processes
  - Leverage a common skill set and language
- Integrated into the Web services stack
  - Expressed entirely in XML
  - Uses and extends WSDL 1.1
  - Uses XML Schema 1.0 for the data model
- Portable across platform and vendor
  - Will run on any WS-BPEL-compliant engine
- Interoperable
  - Layering on top of Web services stack

WS-BPEL in a Nutshell

- WS-BPEL describes in an SOA how your company performs its business processes
- With WS-BPEL, it is straightforward to let your business partners and customers directly participate in your business processes
- With WS-BPEL, it is straightforward to tie in Web services as activities of your business processes

BPEL in SOA: Abstract View

WS-BPEL is a
Recursive Aggregation Model
for Web Services

- Aggregation: a set of Web services can be tied into one or more new Web service by means of a business process model
- Recursive: these new Web services can again be tied into other new Web services

BPEL Process Usage Patterns

- Executable processes
  - Contain the partner's business logic behind an external protocol
- Abstract processes
  - Define the publicly visible behavior of some or all of the services an executable process offers
  - Define a process template embodying domain-specific best practices

WS-BPEL Language Constructs

- WS-BPEL process definition
- Recursive composition and partner links
- Variables
- Variable properties
- Correlation sets
- Basic and structured activities
- Scopes
- Compensation handling

Process Definition

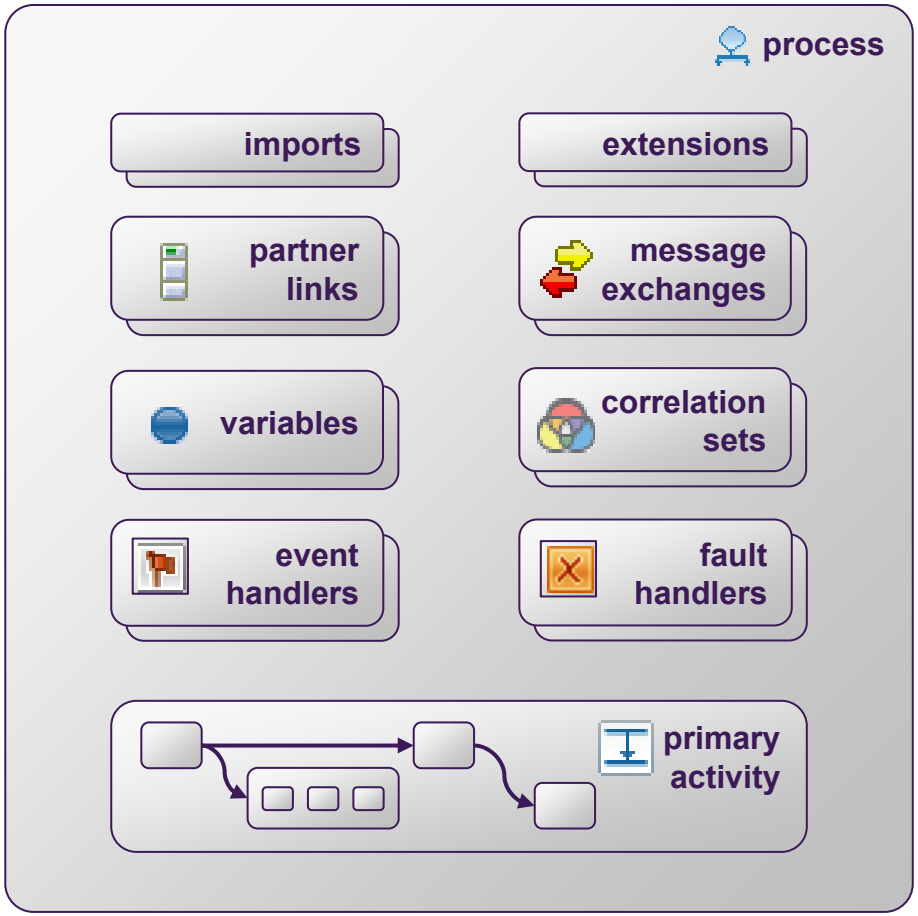
Declare dependencies on external XML Schema or WSDL definitions

Relationships that a WS-BPEL process will employ in its behavior

Data holding state of a business process or exchanged with partners

Concurrently process inbound messages or timer alarms

Perform the process logic – any number of activities may be recursively nested

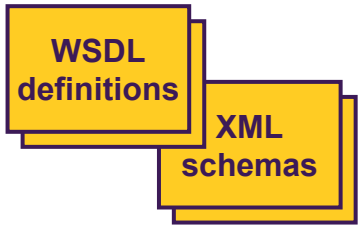


Declare namespaces of WS-BPEL extension attributes and elements

Relationship between inbound and outbound message activities

Application data fields that together identify a conversation

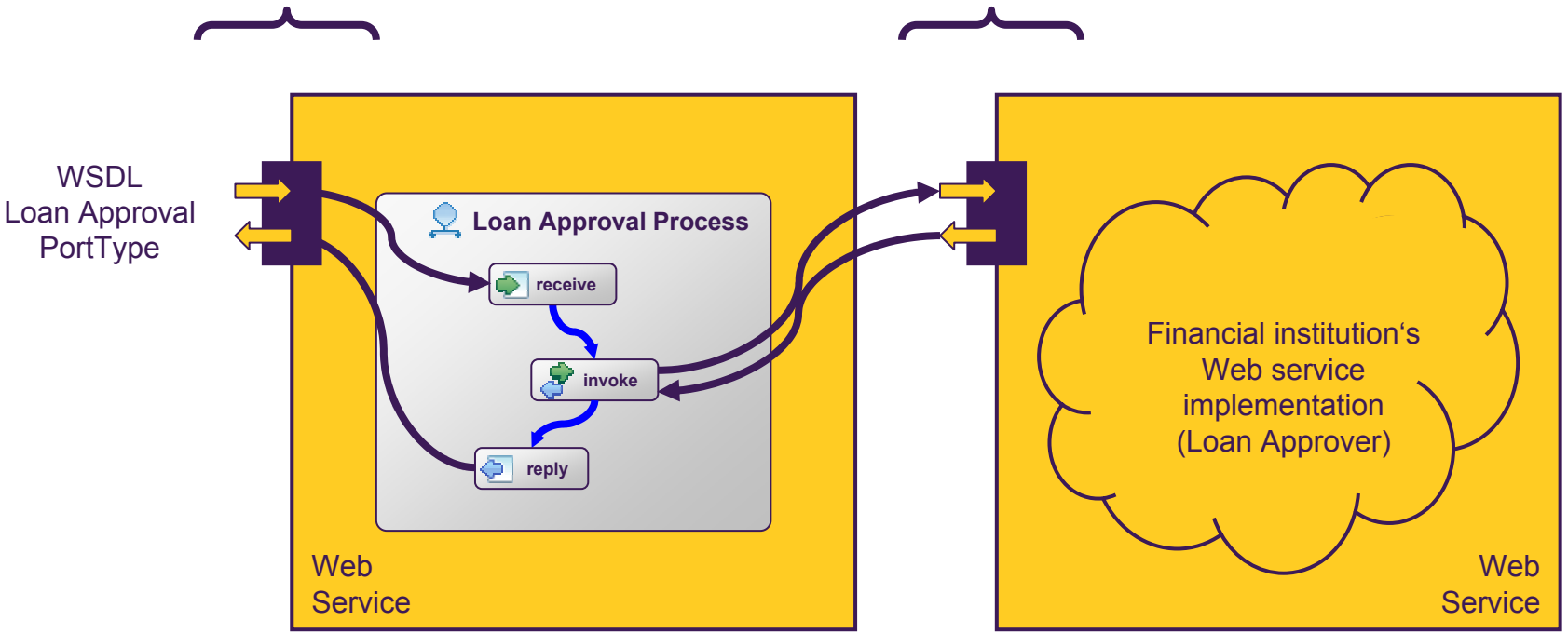
Deal with exceptional situations in a process



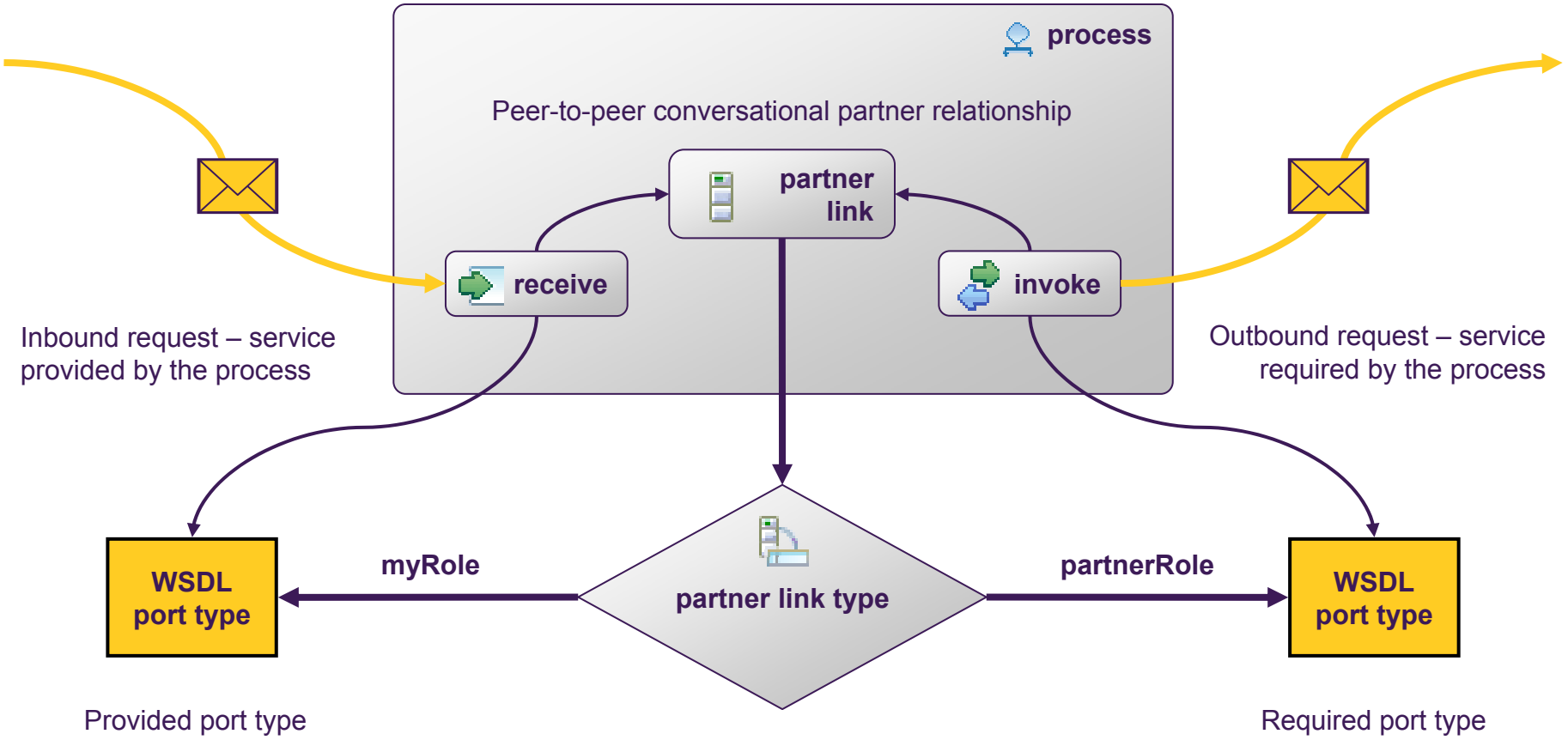
Recursive Composition Model

WS-BPEL processes are exposed as Web services to business partners

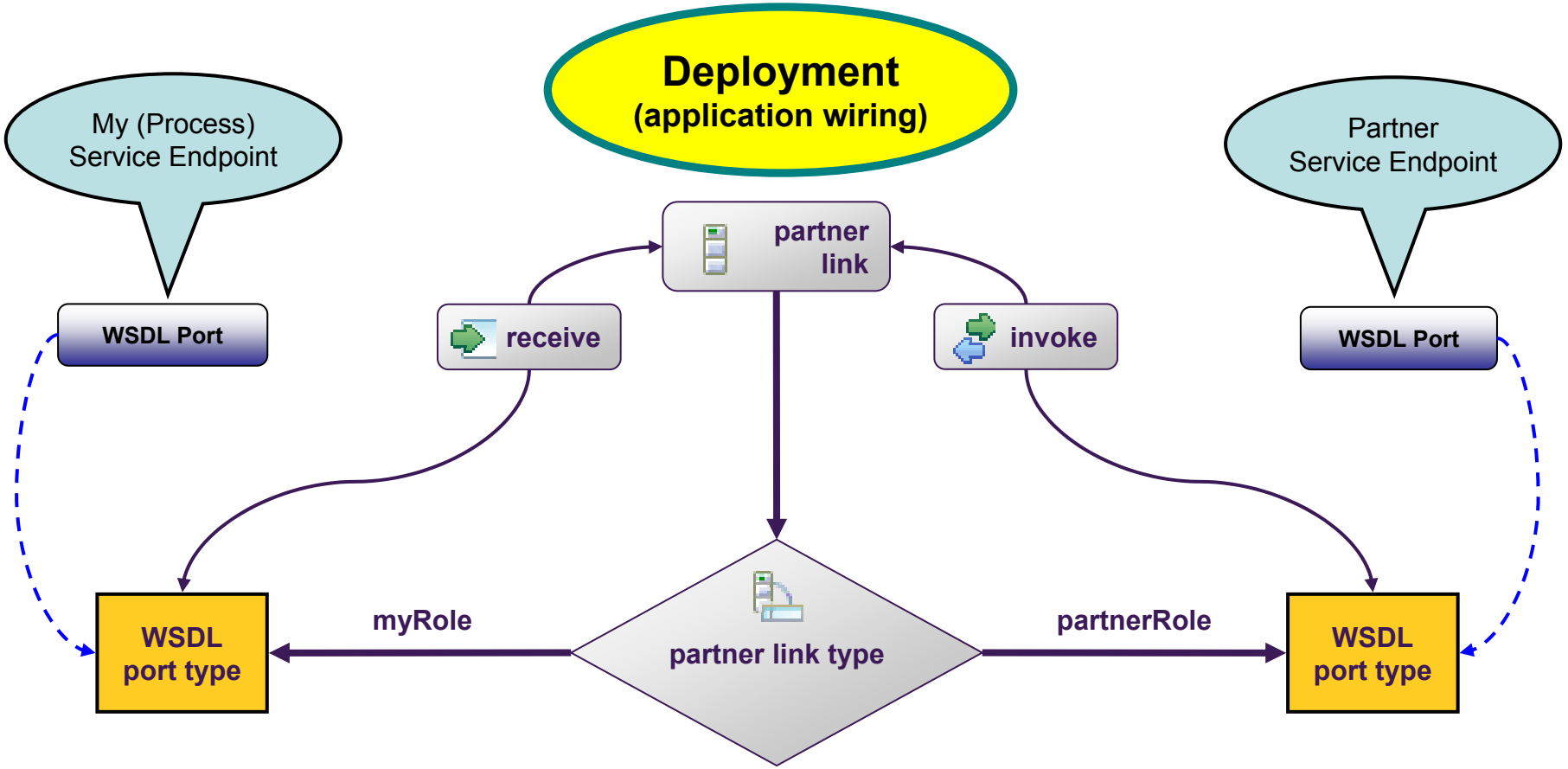
WS-BPEL processes interact with Web services exposed by business partners



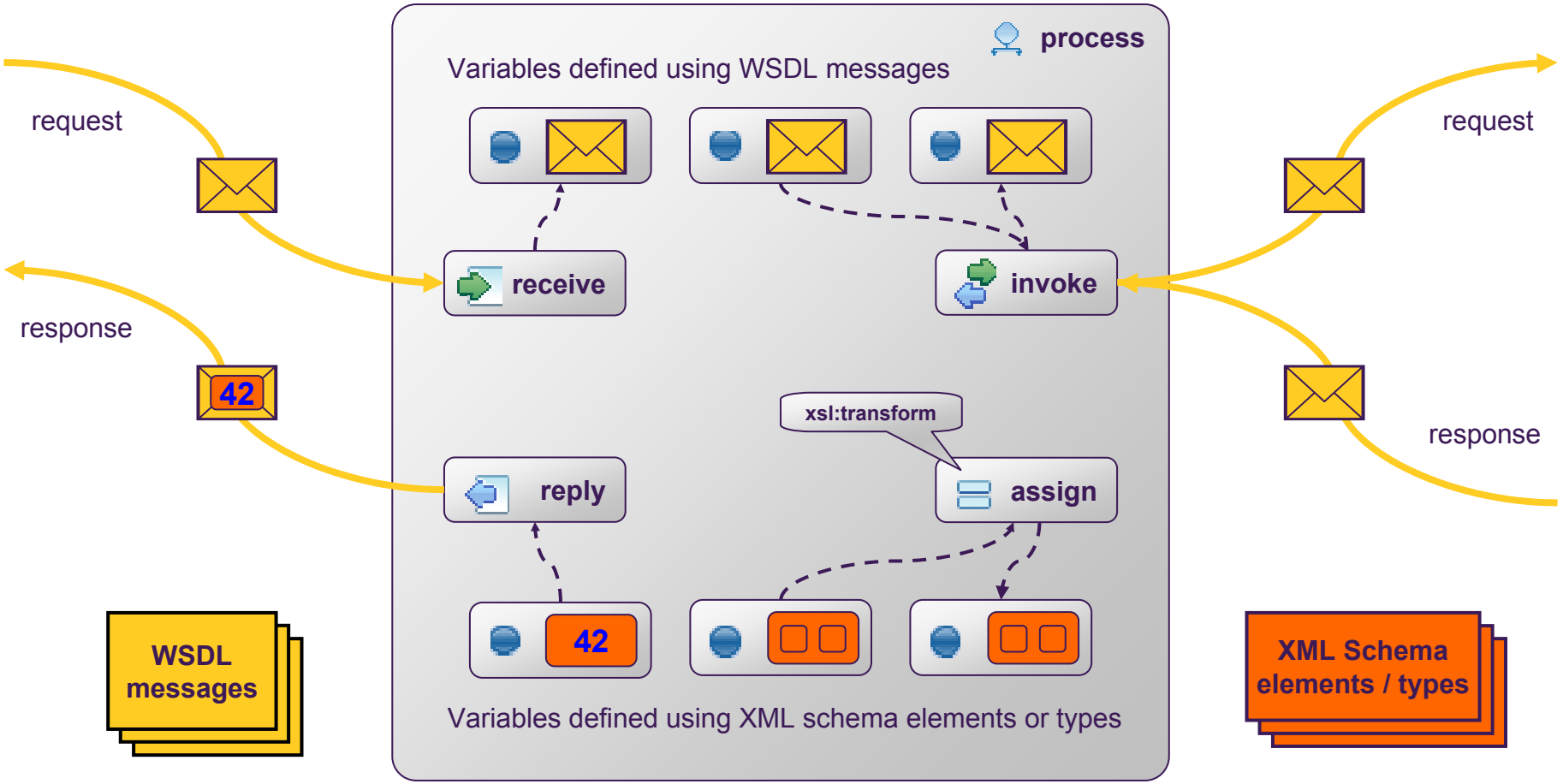
Partner Links



Binding BPEL Partner Links

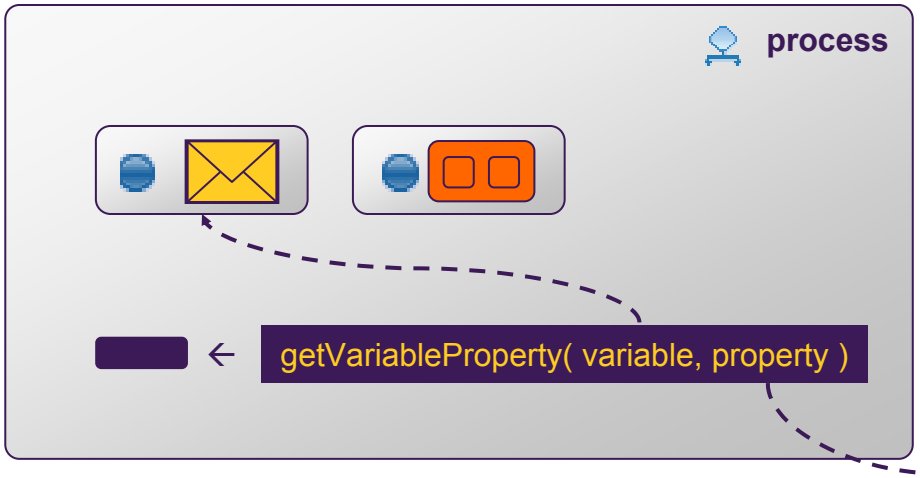


Variables

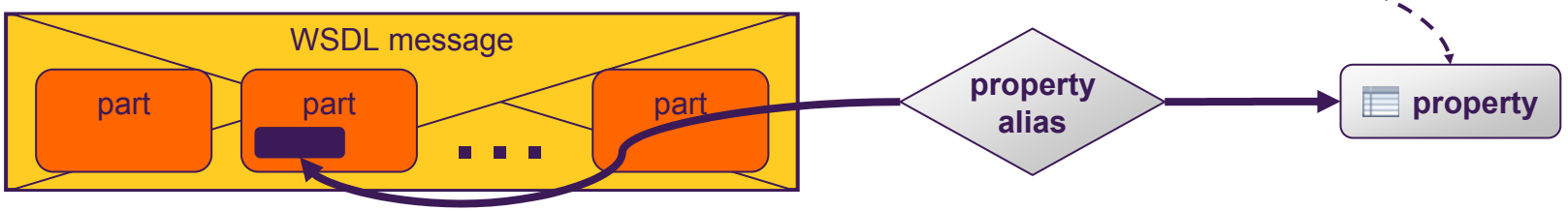

process
Variables defined using WSDL messages
request
response
42
receive
invoke
xsl:transform
reply
assign
42
request
response
WSDL messages
XML Schema elements / types
Variables defined using XML schema elements or types

Variable Properties

A property creates a name that has semantic significance beyond an XML schema type



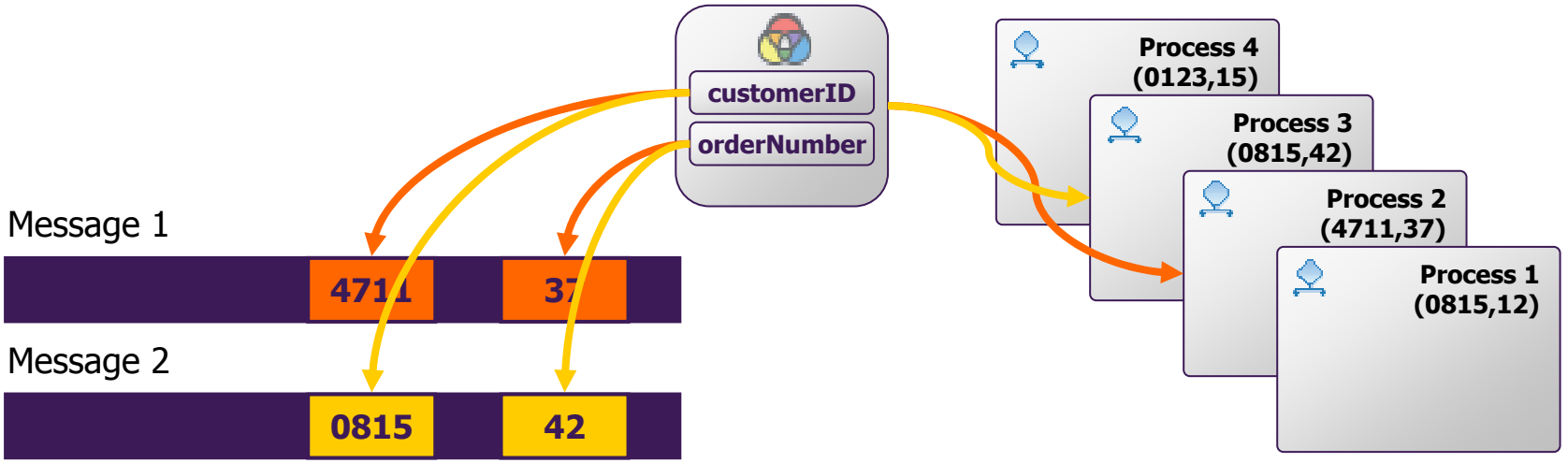
Properties isolate the process logic from the details of a variable definition



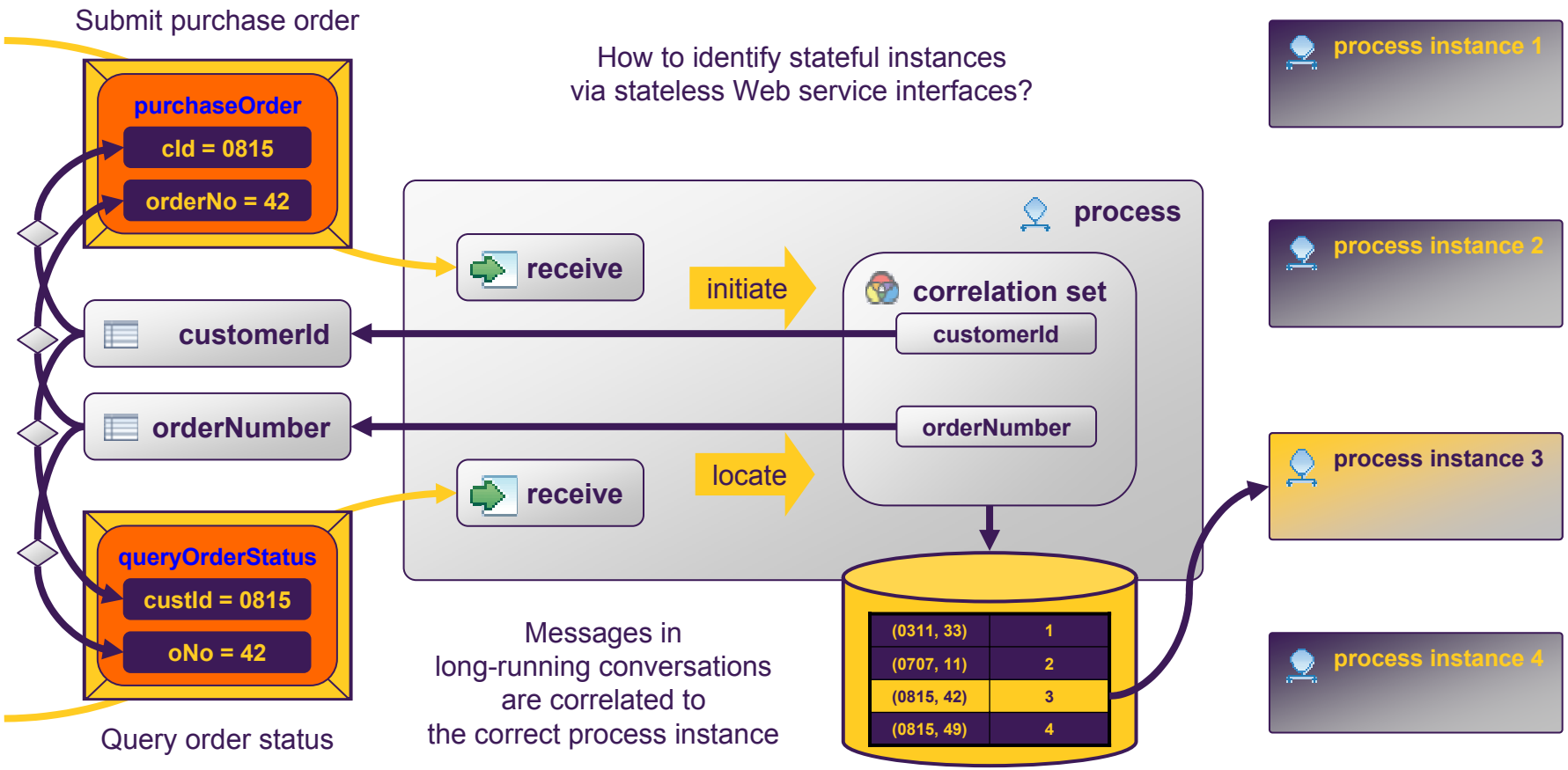
Typed properties are mapped (aliased) to parts of WSDL messages or XML schema elements

Properties and Correlation Sets

- How to identify stateful instances via stateless WS interfaces?
- A process instance is assigned one or more keys
  - Business data is used as key, e.g., customerID
  - A key can be compound, e.g., (customerID, orderNumber)
  - WS-BPEL calls a key a correlation set – it is used to correlate an incoming message with a process instance



Properties and Correlation Sets



BPEL Basic Activities

Do a blocking wait for a matching message to arrive / send a message in reply

Invoke a one-way or request-response operation

Update the values of variables or partner links with new data

Validate XML data stored in variables

Generate a fault from inside the business process

Forward a fault from inside a fault handler

Immediately terminate execution of a business process instance

Invoke compensation on all completed child scopes in default order

Invoke compensation on one completed child scope

Wait for a given time period or until a certain time has passed

No-op instruction for a business process

Wrapper for language extensions

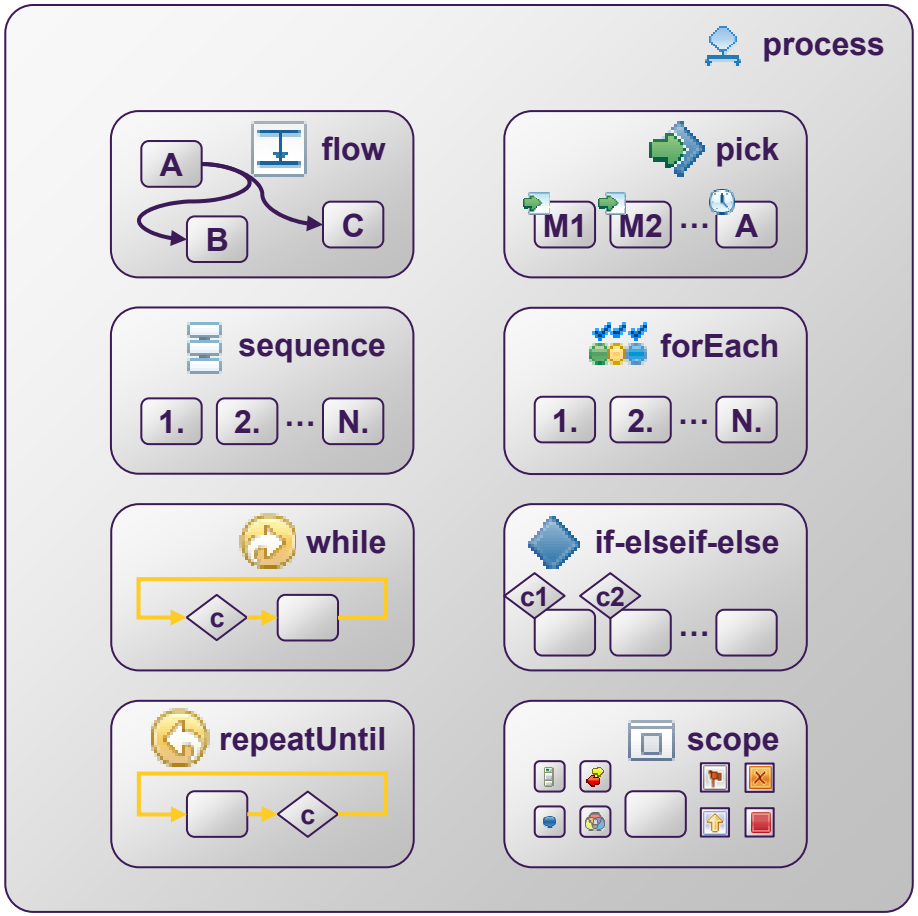
BPEL Structured Activities

Contained activities are executed in parallel, partially ordered through control links

Contained activities are performed sequentially in lexical order

Contained activity is repeated while a predicate holds

Contained activity is repeated until a predicate holds



Block and wait for a suitable message to arrive (or time out)

Contained activity is performed sequentially or in parallel, controlled by a specified counter variable

Select exactly one branch of activity from a set of choices

Associate contained activity with its own local variables, partner links, etc., and handlers

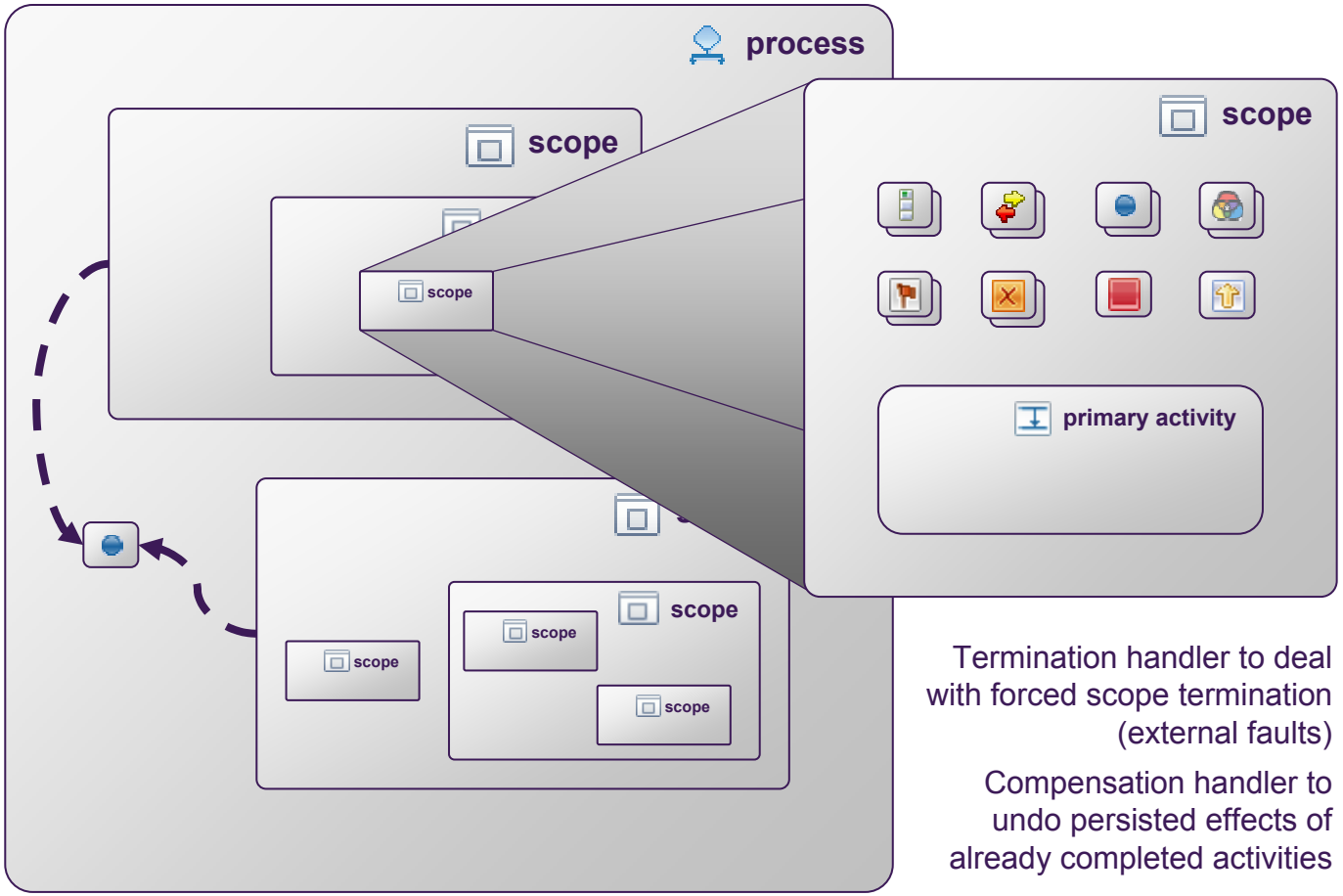
Scopes

Scopes provide a context which influences the execution behavior of its enclosed activities

Local declarations – partner links, message exchanges, variables, correlation sets

Local handlers – event handlers, fault handlers, a termination handler, and a compensation handler

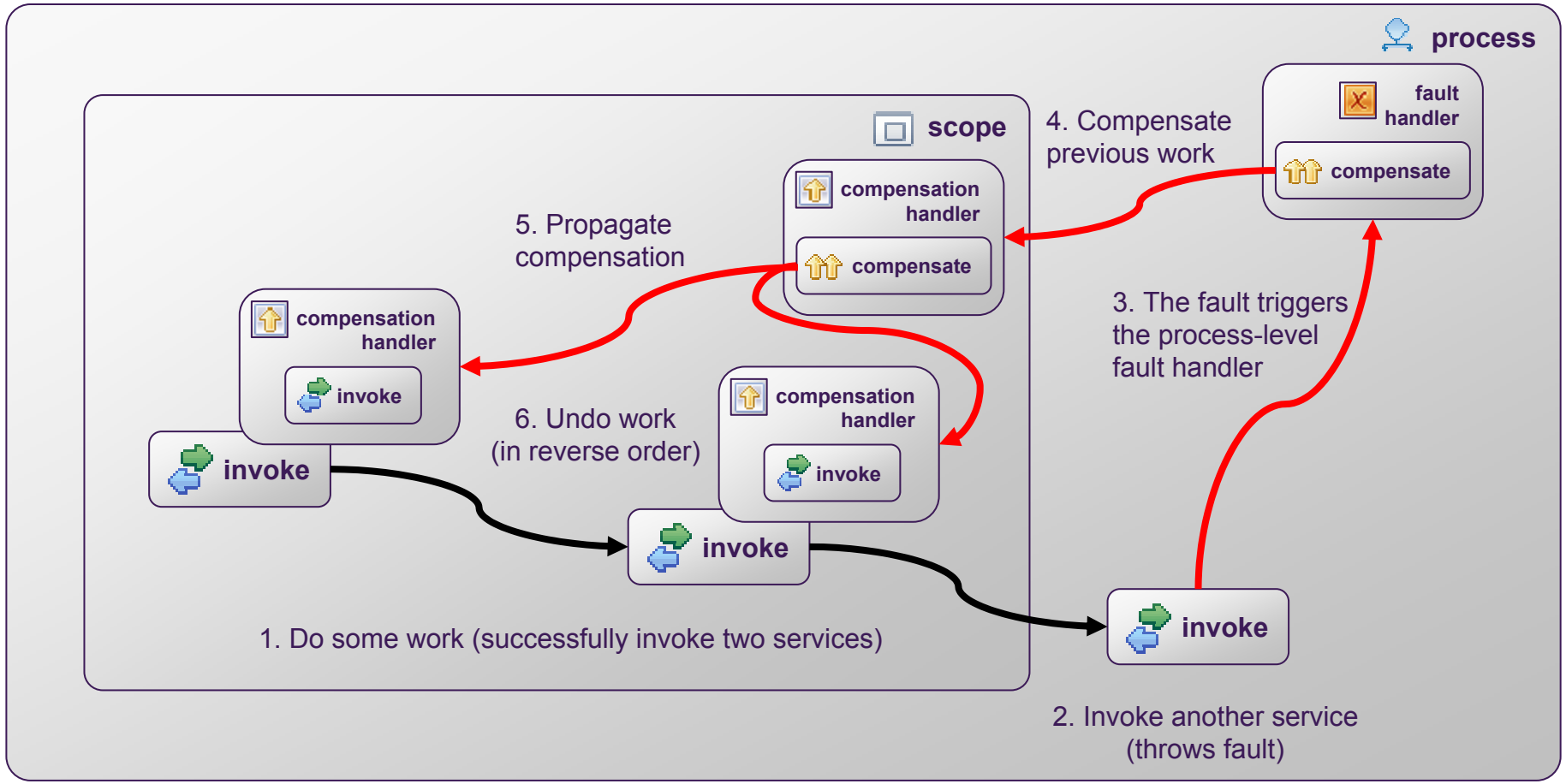
Isolated scopes provide control of concurrent access to shared resources



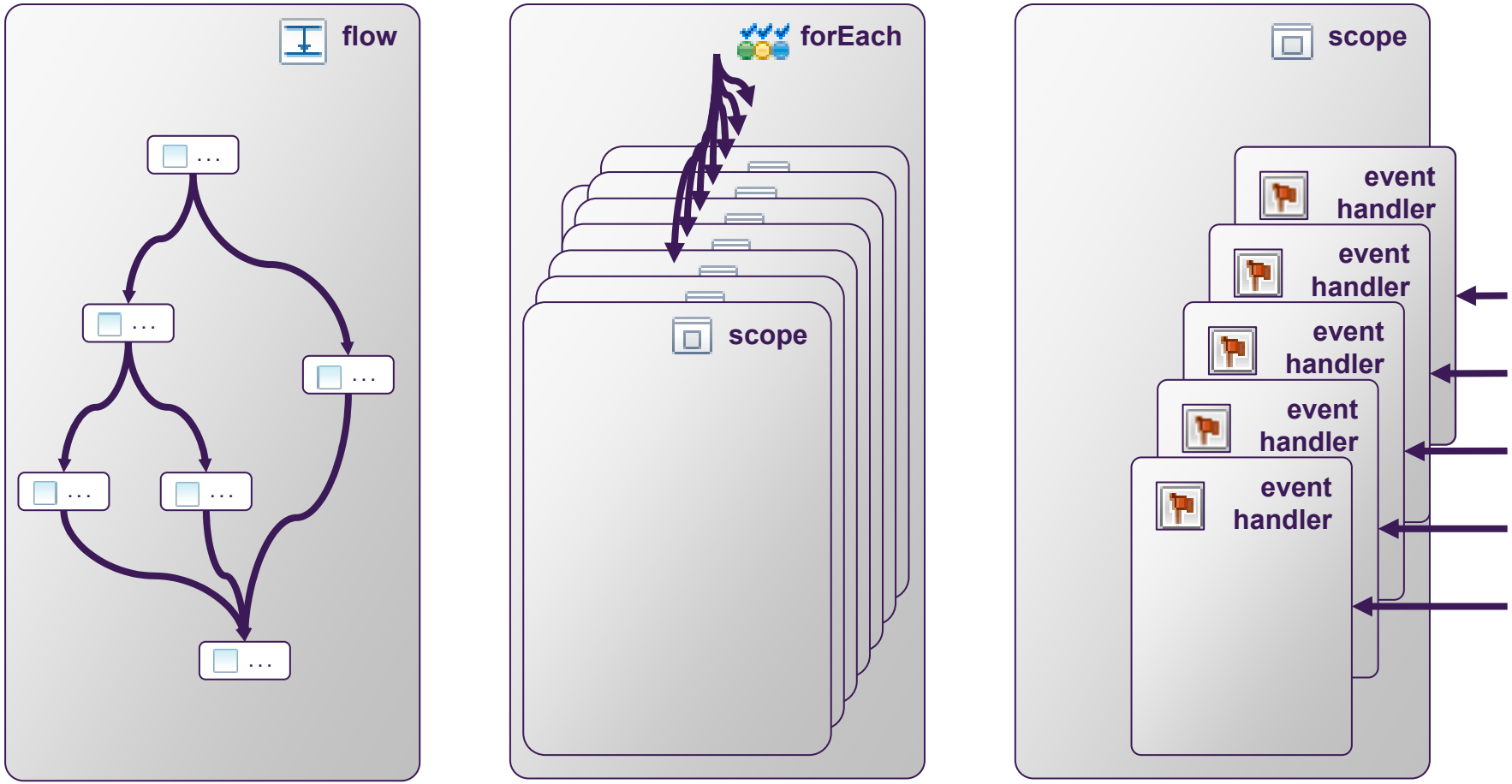
Termination handler to deal with forced scope termination (external faults)

Compensation handler to undo persisted effects of already completed activities

Compensation Handling



Parallel Processing in WS-BPEL



Parallel Processing – ForEach

1. Prepare input for parallel forEach activity

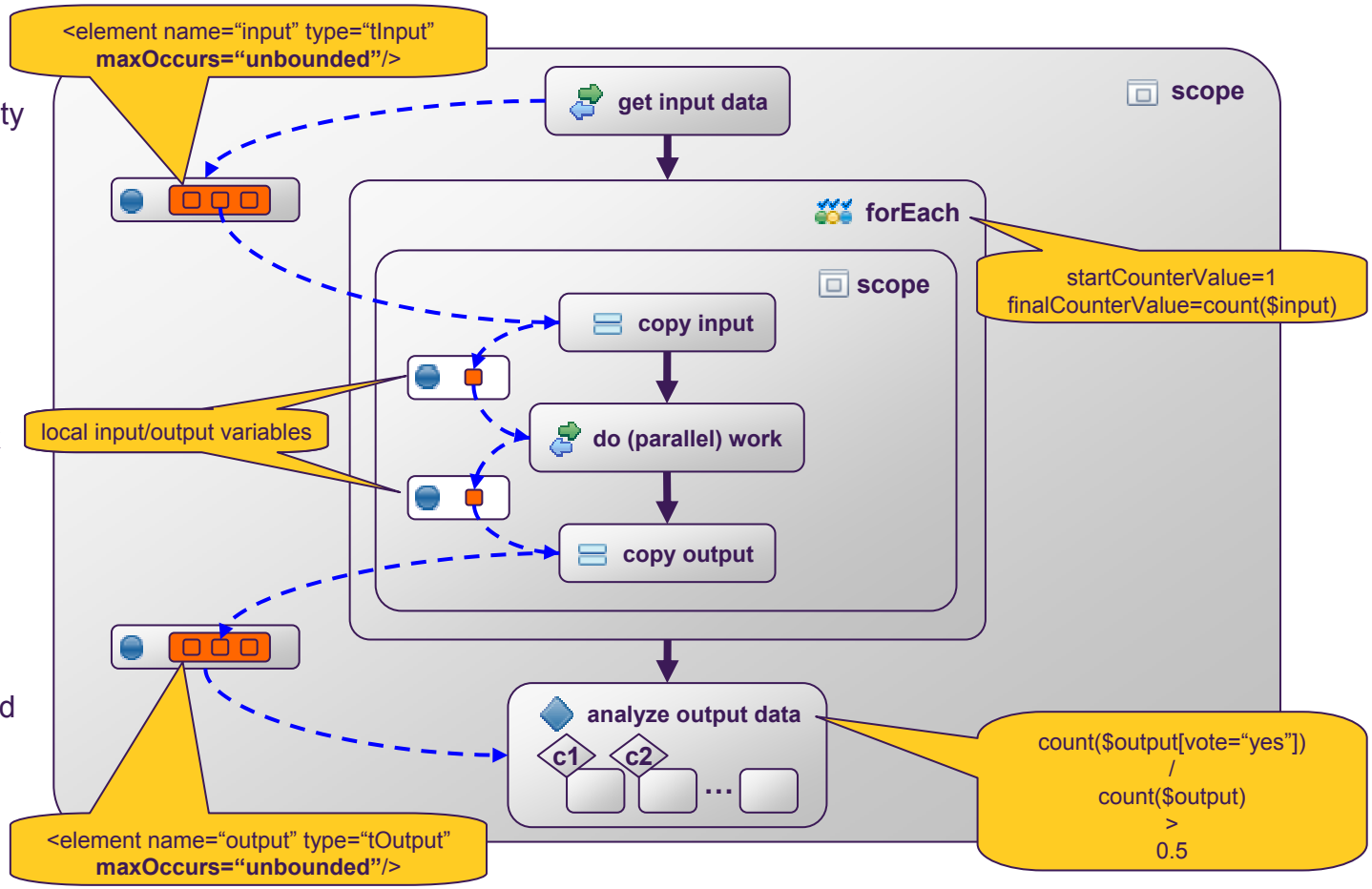
2. Parallel branch for each input element

2a. Copy individual input element

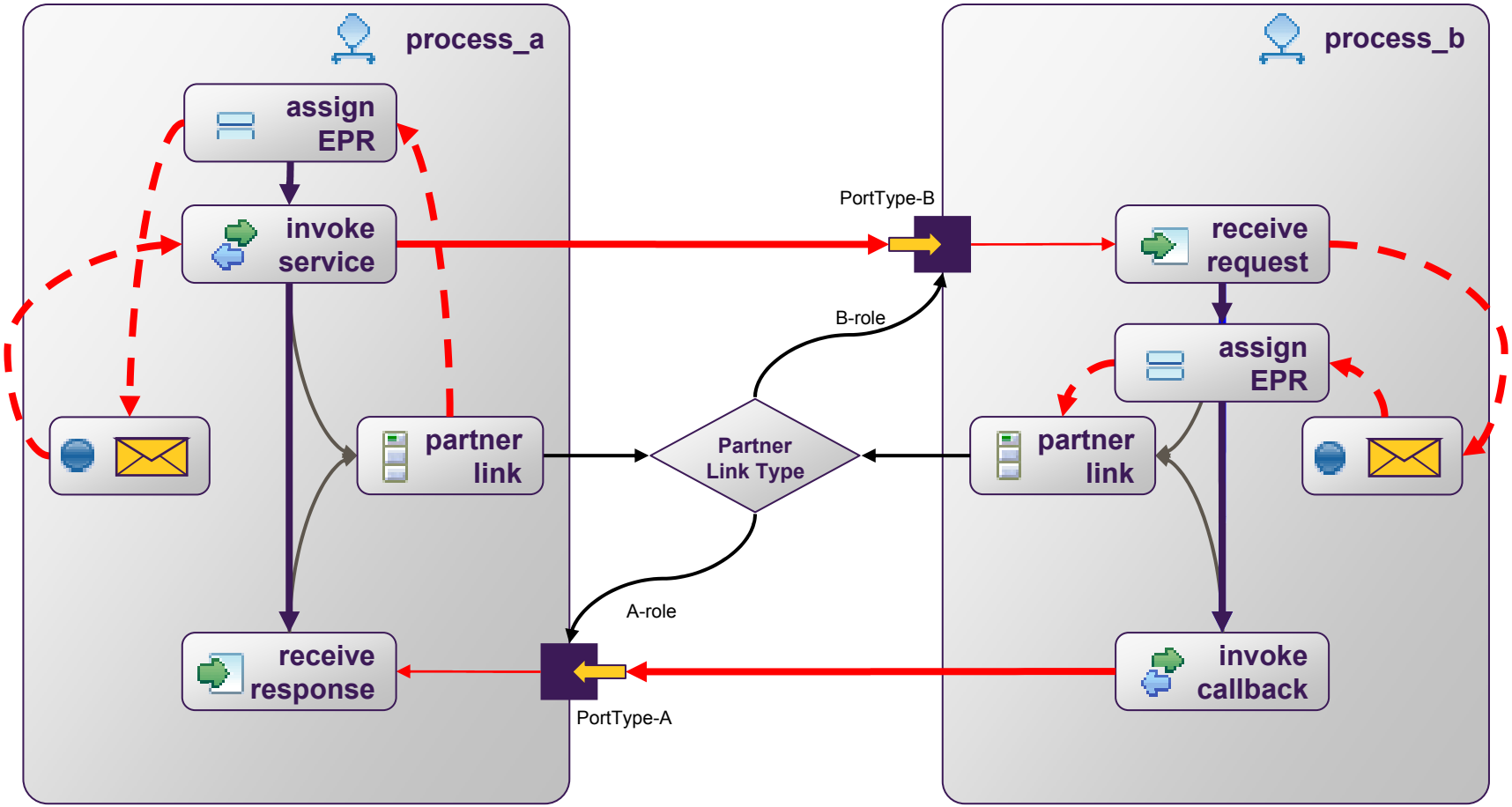
2b. Do (parallel) work

2c. Copy individual output element

3. Analyze aggregated output



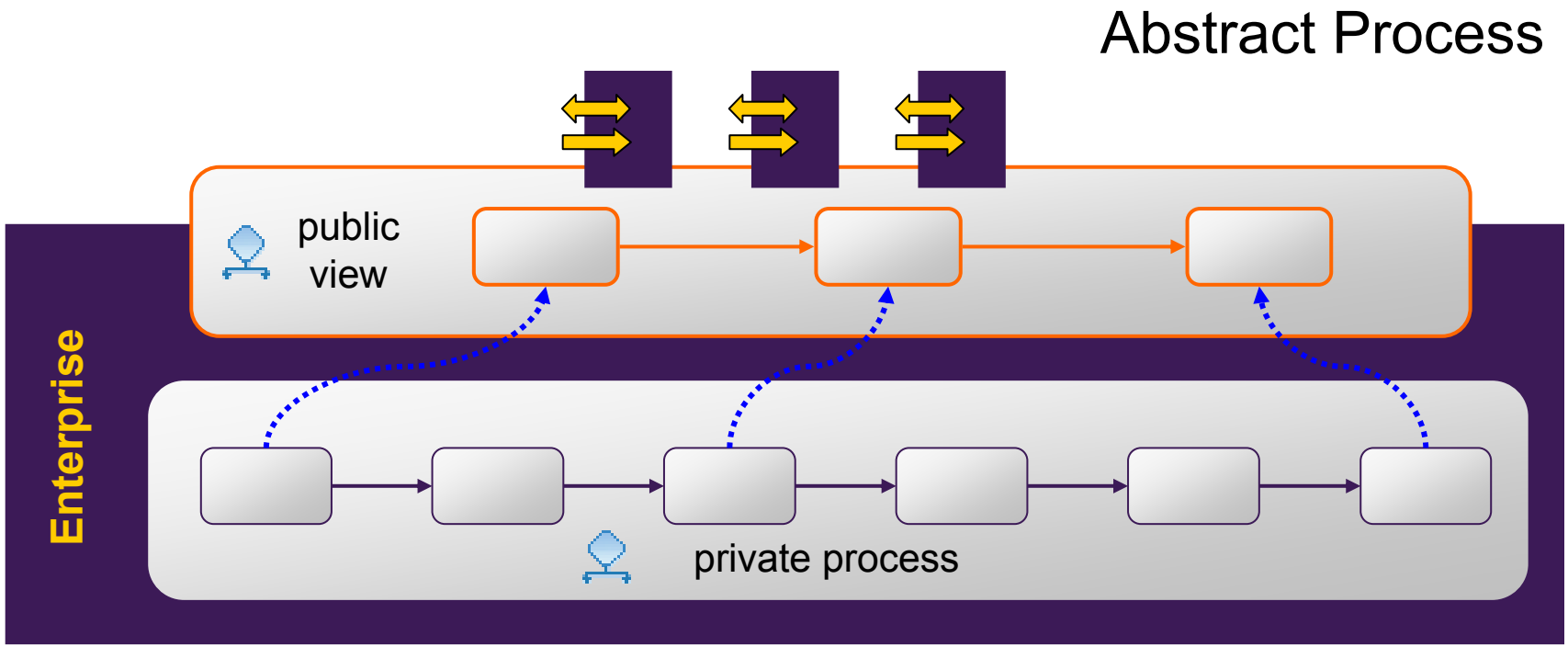
Partner Link Assignment



WS-BPEL Abstract Processes

- Hiding process details
- Abstract process use cases

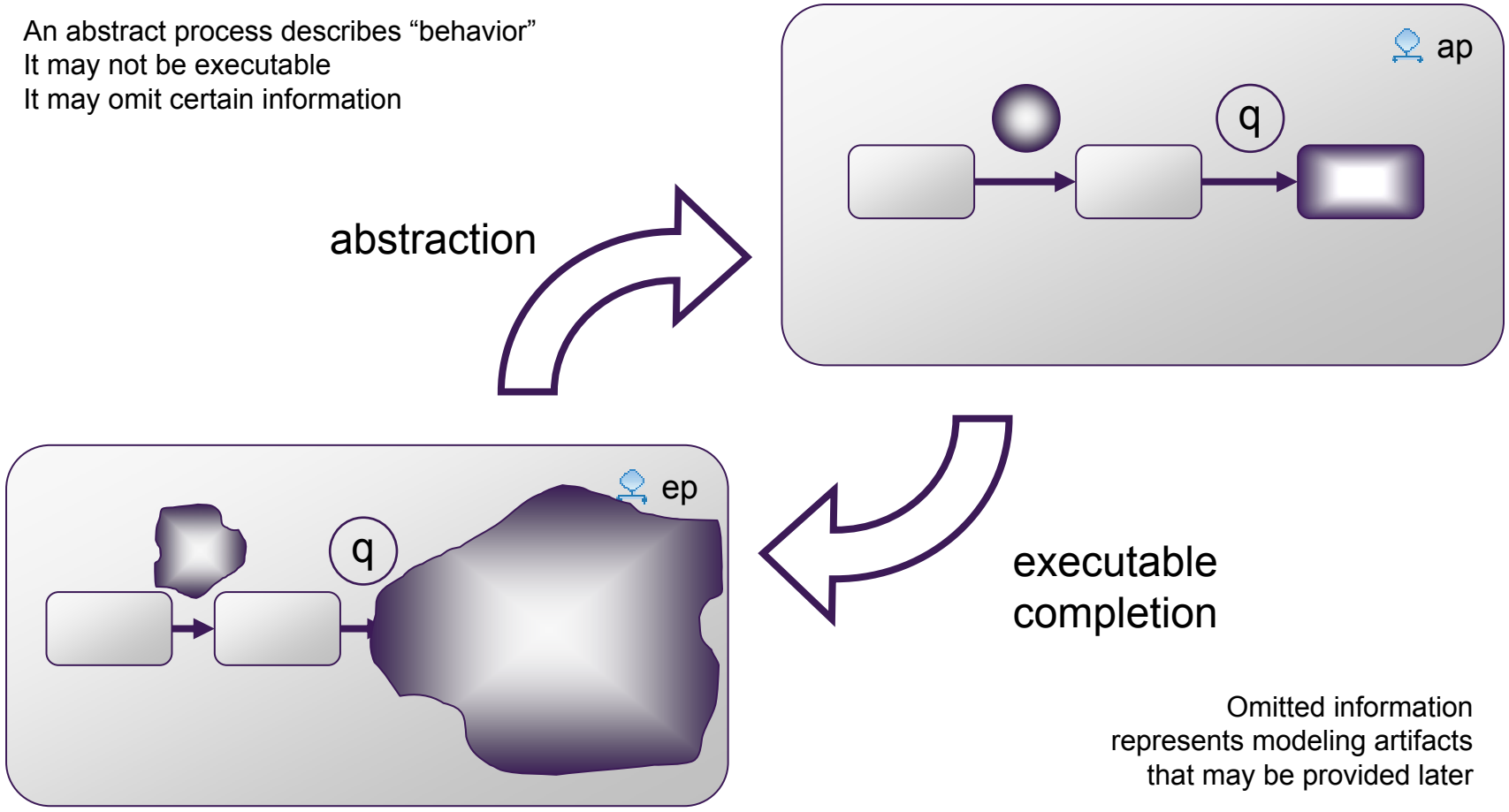
Hiding Process Details



Executable Process

BPEL Abstract Processes

An abstract process describes “behavior”
It may not be executable
It may omit certain information



Omitted information
represents modeling artifacts
that may be provided later

Abstract Process Use Cases

- View on internal process
  - Only a projection of an internal (executable) process is made visible to the outside
    - …to protect process model as corporate asset
    - …to hide non-optimal parts of a process model
- Template as “best practice”
  - Specification of common activities, major data structures, and main control flow
  - Must be refined into an executable processes on a case-by-case basis
- Constraints on message exchange
  - Specification about the order in which messages are consumed or produced
    - Business functionality is implemented as a (set of) port types, and operations must be used in a certain order to achieve intended business goal

View (Export)

- An abstract process is derived from an executable by abstracting away parts that are not part of the behavior one wishes to expose
- Example:
  - Show a particular business partner the interactions that the partner must follow
    - Interactions with all other partners are dropped
  - Use an abstract process to represent common behavior in a set of executables, and drop any non-repeated behavior
    - An executable process of a more general business model may need parts tagged as points of variability, and those are made explicitly opaque

Template (Import)

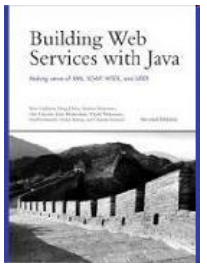
- An abstract process is basis to create one or more executables, or more detailed abstract processes
- Example:
  - One needs to create an implementation of an abstract process provided as a behavioral prescription for complying with a known, domain-specific business function
  - Multiple abstract processes can be created in a series of iterative refinements to a design
  - One wants to implement “best practices” while maintaining some company specifics
    - The abstract process may have been purchased from a consulting firm, as a model of an optimized approach to a problem

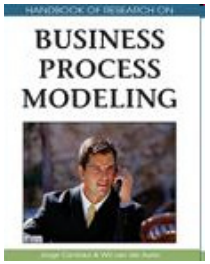
Service Usage Constraint

- Typically, the operations of a services may not be used in order
  - E.g. it doesn't make sense to use the Cancel operation of an Order service before using its Buy operation ☺
- To describe such ordering constraint an abstract process can be used that only refers to operations of a single port type
- The port type of a service may be associated with a process which describes the order in which the operations of the port type can be used

BPEL Resources

- WS-BPEL 2.0 Approved Standard – 04/2007
  http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
- WS-BPEL 2.0 Online Community – 02/2008
  http://bpel.xml.org/
- BPEL4WS 1.1
  http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
- Wikipedia
  http://en.wikipedia.org/wiki/BPEL
- Many books, papers, and analyst reports
  http://www.igi-global.com/reference/details.asp?ID=33287
  http://www.amazon.com/exec/obidos/tg/detail/-/0131488740/qid=1115045110/sr=1-12/ref=sr_1_12/002-7000836-7612862?v=glance&s=books
  http://www.amazon.com/exec/obidos/tg/detail/-/0672326418/qid=1123839655/sr=8-1/ref=pd_bbs_sbs_1/002-7000836-7612862?v=glance&s=books&n=507846





WS-BPEL Language Elements

- Process
- Partner link
- Partner link type
- Message exchange
- Variable
- Property
- Property alias
- Correlation set
- Activity elements and attributes
- Invoke
- Receive
- Reply
- Assign
- Throw
- Compensate
- Wait
- Empty
- Exit
- Validate
- Sequence
- If-else
- While
- RepeatUntil
- Pick
- Flow
- ForEach
- ExtensionActivity
- Scope
- EventHandler
- FaultHandler
- TerminationHandler
- CompensationHandler
- Opaque activity
- Other opaque elements and attributes

Structure of a Business Process


```
<process name="NCName" targetNamespace="anyURI"
   queryLanguage="anyURI"? expressionLanguage="anyURI"?
   suppressJoinFailure="yes|no"? exitOnStandardFault="yes|no"?
   xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable">
   <extensions>?
      <extension namespace="anyURI" mustUnderstand="yes|no" />+
   </extensions>
   <import namespace="anyURI"? location="anyURI"? importType="anyURI" />*

   <partnerLinks>? ... </partnerLinks>
   <messageExchanges>? ... </messageExchanges>
   <variables> ... </variables>
   <correlationSets>? ... </correlationSets>

   <faultHandlers>? ... </faultHandlers>
   <eventHandlers>? ... </eventHandlers>

   activity
</process>
```

PartnerLink & PartnerLinkType

partnerLink

```
<partnerLinks>
   <partnerLink name="NCName"
      partnerLinkType="QName"
      myRole="NCName"?
      partnerRole="NCName"?
      initializePartnerRole="yes|no"? />+
</partnerLinks>
```

partnerLinkType

```
<wsdl:definitions ...>
   ...
   <plnk:partnerLinkType name="NCName">
      <plnk:role name="NCName" portType="QName" />
      <plnk:role name="NCName" portType="QName" />?
   </plnk:partnerLinkType>
   ...
</wsdl:definitions>
```

MessageExchange


```
<messageExchanges>?
   <messageExchange name="NCName" />+
</messageExchanges>
```

Variable


```
<variables>
   <variable name="BPELVariableName"
      messageType="QName"?
      type="QName"?
      element="QName"?>+
      from-spec?
   </variable>
</variables>
```

Property & PropertyAlias

property

```
<wsdl:definitions ... >
   ...
   <vprop:property name="NCName" type="QName"? element="QName"? />
   ...
</wsdl:definitions>
```

propertyAlias

```
<wsdl:definitions ... >
   ...
   <vprop:propertyAlias propertyName="QName"
      messageType="QName"?
      part="NCName"?
      type="QName"?
      element="QName"?>
      <vprop:query queryLanguage="anyURI"?>?
         queryContent
      </vprop:query>
   </vprop:propertyAlias>
   ...
</wsdl:definitions>
```

CorrelationSet


```
<process ...>
   ...
   <correlationSets>?
      <correlationSet name="NCName" properties="QName-list" />+
   </correlationSets>
   ...
</process>
```

Activities – Attributes & Elements

activity

```
<anyActivity name="NCName"?
   suppressJoinFailure="yes|no"?
   ... >
   <targets>?
      <joinCondition expressionLanguage="anyURI"?>?
         bool-expr
      </joinCondition>
      <target linkName="NCName" />+
   </targets>
   <sources>?
      <source linkName="NCName">+
         <transitionCondition expressionLanguage="anyURI"?>?
            bool-expr
         </transitionCondition>
      </source>
   </sources>
   ...
<anyActivity/>
```

Basic Activities – Invoke


```
<invoke partnerLink="NCName"
   portType="QName"? operation="NCName"
   inputVariable="BPELVariableName"? outputVariable="BPELVariableName"?
   standard-attributes>
   standard-elements
   <correlations>?
      <correlation set="NCName" initiate="yes|join|no"?
         pattern="request|response|request-response"? />+
   </correlations>
   <catch ... >* ... </catch>
   <catchAll>? ... </catchAll>
   <compensationHandler>? ... </compensationHandler>
   <toParts>?
      <toPart part="NCName" fromVariable="BPELVariableName" />+
   </toParts>
   <fromParts>?
      <fromPart part="NCName" toVariable="BPELVariableName" />+
   </fromParts>
</invoke>
```

Basic Activities – Receive


```
<receive partnerLink="NCName"
   portType="QName"? operation="NCName"
   variable="BPELVariableName"?
   createInstance="yes|no"?
   messageExchange="NCName"?
   standard-attributes>
   standard-elements
   <correlations>?
      <correlation set="NCName" initiate="yes|join|no"? />+
   </correlations>
   <fromParts>?
      <fromPart part="NCName" toVariable="BPELVariableName" />+
   </fromParts>
</receive>
```

Basic Activities – Reply


```
<reply partnerLink="NCName" portType="QName"? operation="NCName"
   variable="BPELVariableName"?
   faultName="QName"?
   messageExchange="NCName"?
   standard-attributes>
   standard-elements
   <correlations>?
      <correlation set="NCName" initiate="yes|join|no"? />+
   </correlations>
   <toParts>?
      <toPart part="NCName" fromVariable="BPELVariableName" />+
   </toParts>
</reply>
```

Basic Activities – Assign (I)


```
<assign validate="yes|no"? standard-attributes>
   standard-elements
   (
   <copy keepSrcElementName="yes|no"? ignoreMissingFromData="yes|no"? >
      from-spec
      to-spec
   </copy>
   |
   <extensionAssignOperation>
      assign-element-of-other-namespace
   </extensionAssignOperation>
   )+
</assign>
```

Basic Activities – Assign (II)

assign

```
<from variable="BPELVariableName" part="NCName"?>
   <query queryLanguage="anyURI"?>?
      queryContent
   </query>
</from>
<from partnerLink="NCName" endpointReference="myRole|partnerRole" />
<from variable="BPELVariableName" property="QName" />
<from expressionLanguage="anyURI"?>expression</from>
<from><literal>literal value</literal></from>
<from/>
```

assign

```
<to variable="BPELVariableName" part="NCName"?>
   <query queryLanguage="anyURI"?>?
      queryContent
   </query>
</to>
<to partnerLink="NCName" />
<to variable="BPELVariableName" property="QName" />
<to expressionLanguage="anyURI"?>expression</to>
<to/>
```

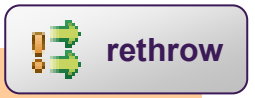
Queries and Expressions

```
<query queryLanguage="anyURI"?>query</query>
<expression expressionLanguage="anyURI"?>expression</expression>
```

- Queries and Expressions
  - XML infoset selection query
    - from and to query in <copy>
    - query in <propertyAlias>
  - Boolean expressions
    - transition, join, while, and if conditions
  - Deadline expressions
    - until expression of <onAlarm> and <wait>
  - Duration expressions
    - for expression of <onAlarm> and <wait>,
    - repeatEvery expression of <onAlarm>
  - Unsigned Integer expressions
    - startCounterValue, finalCounterValue, and branches in <forEach>
  - General expressions
    - from expressions in <copy>
- XPath variable binding
  - $variable-name [. part-name]
- XPath extension functions
  - object bpel:getVariableProperty(string, string)
  - object bpel:doXslTransform(string, node-set, (string, object)*)

Basic Activities – Throw


```
<throw faultName="QName" faultVariable="BPELVariableName"?
   standard-attributes>
   standard-elements
</throw>
```




```
<rethrow standard-attributes>
   standard-elements
</rethrow>
```

Basic Activities – Compensate


```
<compensate standard-attributes>
   standard-elements
</compensate>
```


```
<compensateScope target="NCName" standard-attributes>
   standard-elements
</compensateScope>
```

Basic Activities – Wait


```
<wait standard-attributes>
   standard-elements
   (
   <for expressionLanguage="anyURI"?>duration-expr</for>
   |
   <until expressionLanguage="anyURI"?>deadline-expr</until>
   )
</wait>
```

Basic Activities – Empty


```
<empty standard-attributes>
   standard-elements
</empty>
```

Basic Activities – Exit


```
<exit standard-attributes>
   standard-elements
</exit>
```

Basic Activities – Validate


```
<validate variables="BPELVariableNames" standard-attributes>
   standard-elements
</validate>
```

Structured Activities – Sequence


```
<sequence standard-attributes>
   standard-elements
   activity+
</sequence>
```

Structured Activities – If-Else


```
<if standard-attributes>
   standard-elements
   <condition expressionLanguage="anyURI"?>bool-expr</condition>
   activity
   <elseif>*
      <condition expressionLanguage="anyURI"?>bool-expr</condition>
      activity
   </elseif>
   <else>?
      activity
   </else>
</if>
```

Structured Activities – While


```
<while standard-attributes>
   standard-elements
   <condition expressionLanguage="anyURI"?>bool-expr</condition>
   activity
</while>
```

Structured Activities – RepeatUntil


```
<repeatUntil standard-attributes>
   standard-elements
   activity
   <condition expressionLanguage="anyURI"?>bool-expr</condition>
</repeatUntil>
```

Structured Activities – Pick


```
<pick createInstance="yes|no"? standard-attributes>
   standard-elements
   <onMessage partnerLink="NCName" portType="QName"? operation="NCName"
      variable="BPELVariableName"?
      messageExchange="NCName"?>+
      <correlations>?
         <correlation set="NCName" initiate="yes|join|no"? />+
      </correlations>
      <fromParts>?
         <fromPart part="NCName" toVariable="BPELVariableName" />+
      </fromParts>
      activity
   </onMessage>
   <onAlarm>*
      ( <for expressionLanguage="anyURI"?>duration-expr</for>
      | <until expressionLanguage="anyURI"?>deadline-expr</until> )
      activity
   </onAlarm>
</pick>
```

Structured Activities – Flow


```
<flow standard-attributes>
   standard-elements
   <links>?
      <link name="NCName">+
   </links>
   activity+
</flow>
```

Structured Activities – ForEach


```
<forEach counterName="BPELVariableName" parallel="yes|no"
   standard-attributes>
   standard-elements
   <startCounterValue expressionLanguage="anyURI"?>
      unsigned-integer-expression
   </startCounterValue>
   <finalCounterValue expressionLanguage="anyURI"?>
      unsigned-integer-expression
   </finalCounterValue>
   <completionCondition>?
      <branches expressionLanguage="anyURI"?
         successfulBranchesOnly="yes|no"?>
         unsigned-integer-expression
      </branches>
   </completionCondition>
   <scope ...>...</scope>
</forEach>
```

ExtensionActivity


```
<extensionActivity>
   <anyElementQName standard-attributes>
      standard-elements
   </anyElementQName>
</extensionActivity>
```

Scopes


```
<scope isolated="yes|no"? exitOnStandardFault="yes|no"?
   standard-attributes>
   standard-elements

   <variables>? ... </variables>
   <partnerLinks>? ... </partnerLinks>
   <messageExchanges>? ... </messageExchanges>
   <correlationSets>? ... </correlationSets>

   <eventHandlers>? ... </eventHandlers>
   <faultHandlers>? ... </faultHandlers>
   <compensationHandler>? ... </compensationHandler>
   <terminationHandler>? ... </terminationHandler>

   activity
</scope>
```

EventHandlers

eventHandlers

```
<eventHandlers>?
   <onEvent partnerLink="NCName"
      portType="QName"? operation="NCName"
      ( messageType="QName" | element="QName" )?
      variable="BPELVariableName"?
      messageExchange="NCName"?>*
      <correlations>? ... </correlations>
      <fromParts>? ... </fromParts>
      <scope ...> ... </scope>
   </onEvent>
   <onAlarm>*
      ( <for expressionLanguage="anyURI"?>duration-expr</for>
      | <until expressionLanguage="anyURI"?>deadline-expr</until> )?
      <repeatEvery expressionLanguage="anyURI"?>?
         duration-expr
      </repeatEvery>
      <scope ...> ... </scope>
   </onAlarm>
</eventHandlers>
```

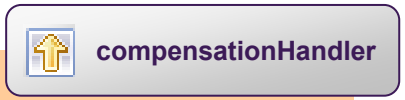
FaultHandlers


```
<faultHandlers>
   <catch faultName="QName"?
      faultVariable="BPELVariableName"?
      ( faultMessageType="QName" | faultElement="QName" )? >*
      activity
   </catch>
   <catchAll>?
      activity
   </catchAll>
</faultHandlers>
```

TerminationHandler


```
<terminationHandler>
   activity
</terminationHandler>
```

CompensationHandler




```
<compensationHandler>
   activity
</compensationHandler>
```

Service References

```
<sref:service-ref reference-scheme="anyURI"? >
   ...
</sref:service-ref>
```

```
<sref:service-ref
   reference-scheme="http://www.w3.org/2005/08/addressing">
   <wsa:EndpointReference>
      <wsa:Address>...</wsa:Address>
      ...
   </wsa:EndpointReference>
</sref:service-ref>
```

Abstract Processes


```
<process name="NCName" targetNamespace="anyURI"
   queryLanguage="anyURI"? expressionLanguage="anyURI"?
   suppressJoinFailure="yes|no"? exitOnStandardFault="yes|no"?
   xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/abstract">
   <extensions>?
      <extension namespace="anyURI" mustUnderstand="yes|no" />+
   </extensions>
   <import namespace="anyURI"? location="anyURI"? importType="anyURI" />*

   <partnerLinks>? ... </partnerLinks>
   <messageExchanges>? ... </messageExchanges>
   <variables> ... </variables>
   <correlationSets>? ... </correlationSets>

   <faultHandlers>? ... </faultHandlers>
   <eventHandlers>? ... </eventHandlers>

   activity
</process>
```

Opaque Activity


```
<opaqueActivity standard-attributes>
   standard-elements
</opaqueActivity>
```

Opaque Queries & Expressions

```
<transitionCondition expressionLanguage="anyURI"? opaque="yes"/>
<joinCondition expressionLanguage="anyURI"? opaque="yes"/>
<condition expressionLanguage="anyURI"? opaque="yes"/>
<until expressionLanguage="anyURI"? opaque="yes"/>
<for expressionLanguage="anyURI"? opaque="yes"/>
<repeatEvery expressionLanguage="anyURI"? opaque="yes"/>
<startCounterValue expressionLanguage="anyURI"? opaque="yes"/>
<finalCounterValue expressionLanguage="anyURI"? opaque="yes"/>
<branches ... expressionLanguage="anyURI"? opaque="yes"/>
<from expressionLanguage="anyURI"? opaque="yes"/>
<from variable="BPELVariableName" part="NCName"?>
   <query queryLanguage="anyURI"? opaque="yes"/>?
</from>
<to expressionLanguage="anyURI"? opaque="yes"/>
<to variable="BPELVariableName" part="NCName"?>
   <query queryLanguage="anyURI"? opaque="yes"/>?
</to>
```

Opaque From

```
<opaqueFrom/>
```

Opaque Attribute Values

```
##BPELopaque
```

Web Service Orchestration and WS-BPEL

- Part I – Web Service Orchestration
  - Motivation and Overview
  - Business Process Modeling Styles
- Part II – The WS-BPEL 2.0 Standard
  - WS-BPEL 2.0 Concepts and Language Elements
- Part III – WS-BPEL 2.0 Extensions
  - WS-BPEL Extension for People (BPEL4People)
  - WS-BPEL Extension for Subprocesses (BPEL-SPE)
  - WS-BPEL Extension for Java (BPELJ)
- Part IV – Additional Related Standards
  - Service Component Architecture (SCA)
  - Business Process Modeling Notation (BPMN)

BPEL4People Objectives

- BPEL4People & WS-HumanTask
  - Integration of human-executed activities in Web services-based business processes
  - Integration of human-executed activities in SOA-based applications
  - Standard-based solution to support interoperability and portability scenarios

BPEL4People History

- June 2007
  - B4P and WS-HT specifications published

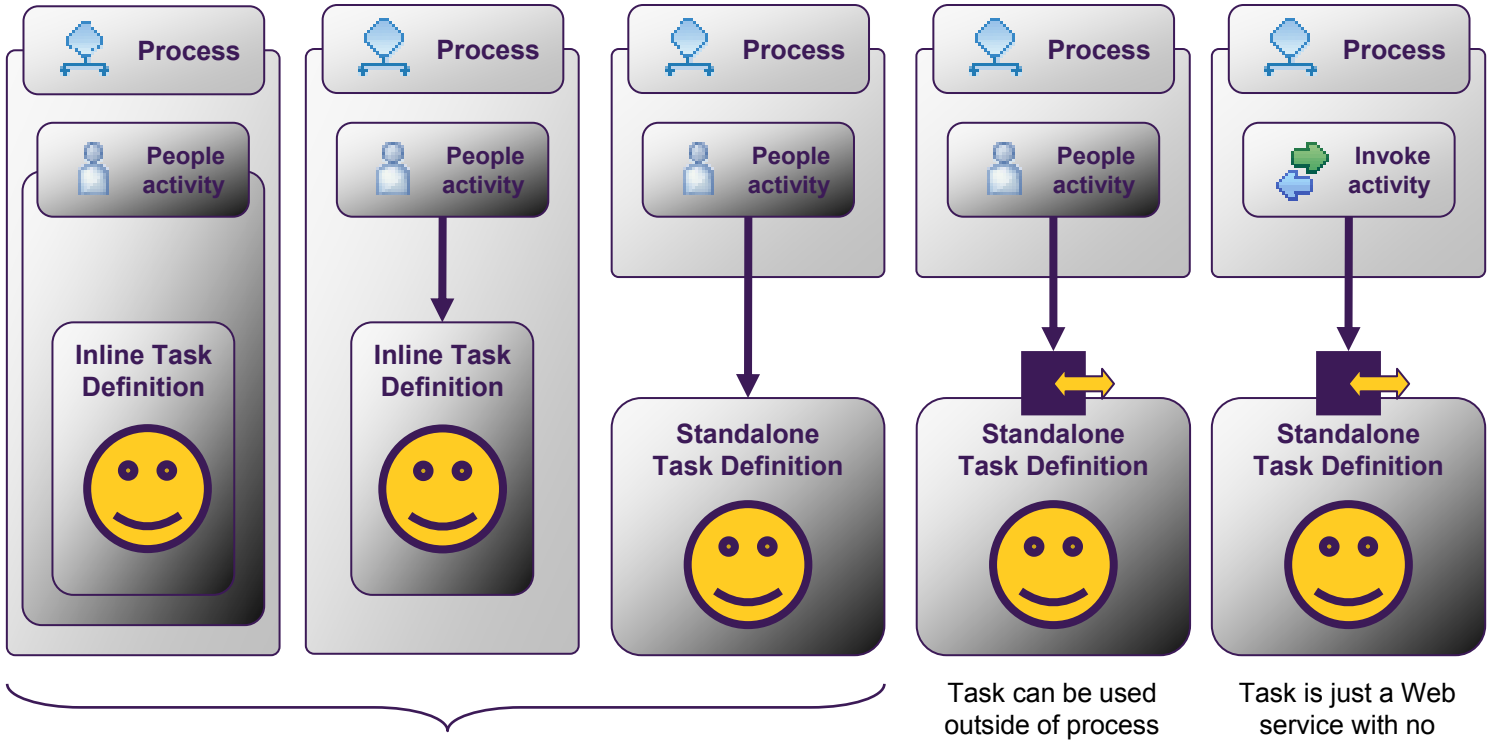
    http://www-128.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/

- February 2008
  - OASIS Technical Committee created
- March 2008
  - OASIS Technical Committee Inaugural Meeting

BPEL4People Approach

- BPEL4People
  - Definition of human interactions within WS-BPEL processes
  - Specification built on top of WS-BPEL 2.0
- WS-HumanTask
  - Definition of service-enabled human tasks and notifications
  - Coordination protocol used to control autonomy and life cycle of service-enabled human tasks in an interoperable manner
  - Interoperable programming interface enabling task client applications to work with human tasks

BPEL4People Constellations



Task can only be used inside of process context

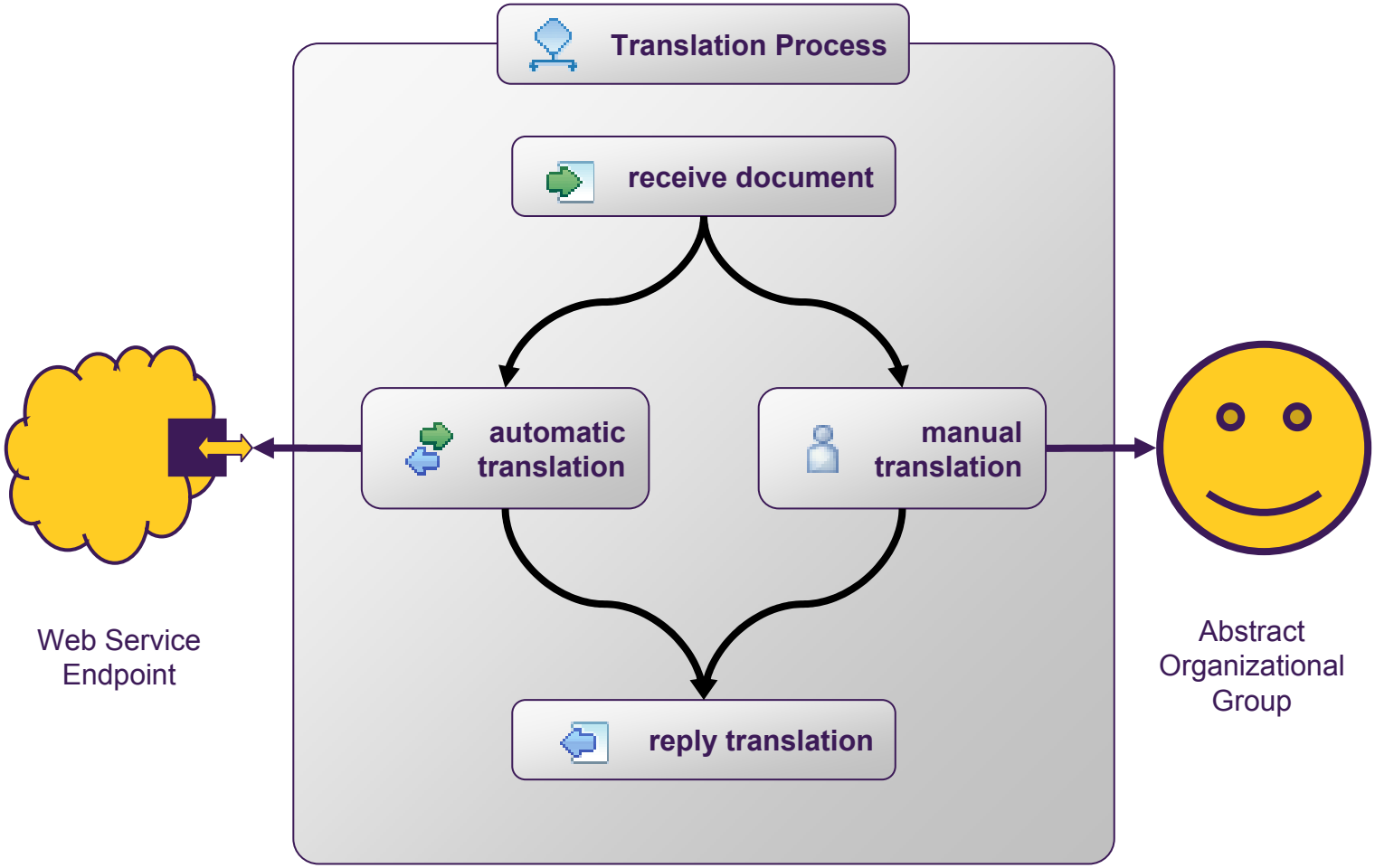
Task can be used outside of process context

Communication with task is based on coordination protocol

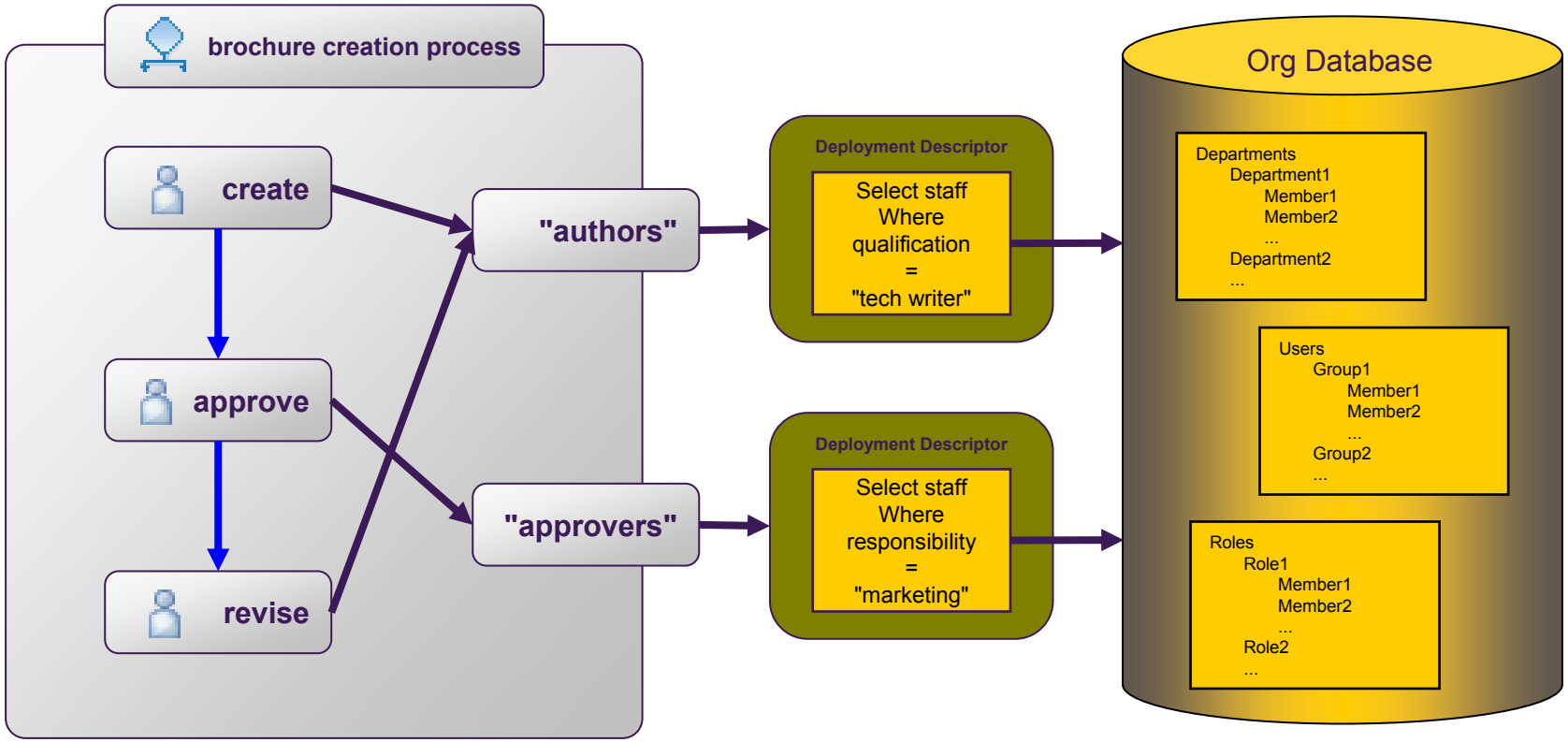
Task is just a Web service with no additional constraints

Easy to switch between performing the task by humans or programs

BPEL4People Human Task



WS-HT People Resolution



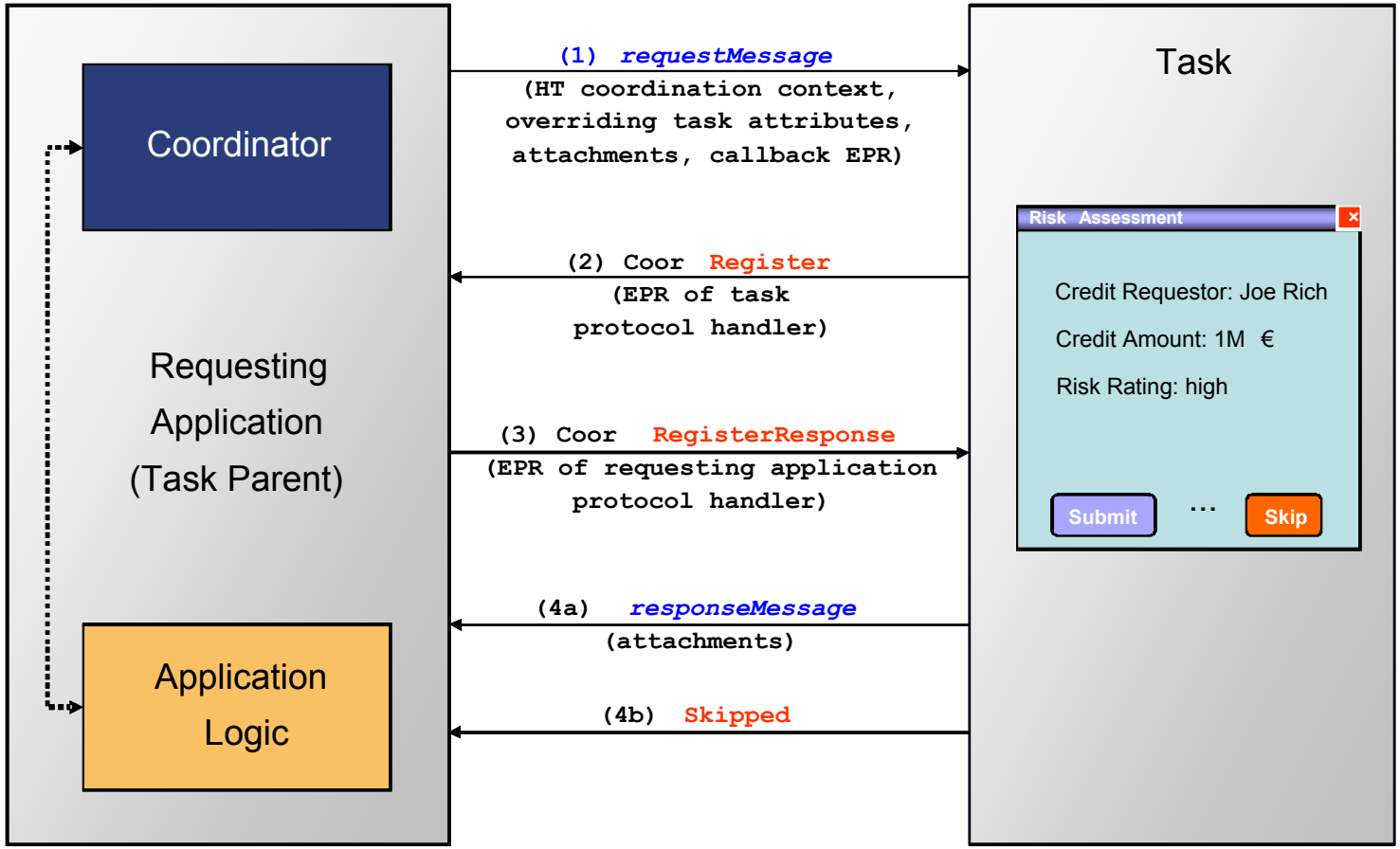
People Activity Definition

```
<bpel:extensionActivity>
  <b4p:peopleActivity name="NCName" inputVariable="NCName"?
    outputVariable="NCName"? isSkipable="xsd:boolean"?
    standard-attributes>
    standard-elements
    ( <htd:task>...</htd:task>
    | <b4p:localTask>...</b4p:localTask>
    | <b4p:remoteTask>...</b4p:remoteTask>
    | <htd:notification>...</htd:notification>
    | <b4p:localNotification>...</b4p:localNotification>
    | <b4p:remoteNotification>...</b4p:remoteNotification>
    )
    <b4p:scheduledActions>? ... </b4p:scheduledActions>
    <bpel:toParts>?
      <bpel:toPart part="NCName" fromVariable="BPELVariableName" />+
    </bpel:toParts>
    <bpel:fromParts>?
      <bpel:fromPart part="NCName" toVariable="BPELVariableName" />+
    </bpel:fromParts>
    <b4p:attachmentPropagation fromProcess="all|none"
      toProcess="all|newOnly|none" />?
  </b4p:peopleActivity>
</bpel:extensionActivity>
```

Human Task Definition

```
<htd:task name="NCName">
  <htd:interface portType="QName" operation="NCName"
    responsePortType="QName"? responseOperation="NCName"? />
  <htd:priority expressionLanguage="anyURI"? >?
    integer-expression
  </htd:priority>
  <htd:peopleAssignments> ... </htd:peopleAssignments>
  <htd:delegation potentialDelegatees="anybody|nobody|potentialOwners|other" />?
    <htd:from>? ... </htd:from>
  </htd:delegation>
  <htd:presentationElements> ... </htd:presentationElements>
  <htd:outcome part="NCName" queryLanguage="anyURI">? queryContent </htd:outcome>
  <htd:searchBy expressionLanguage="anyURI"? >? expression </htd:searchBy>
  <htd:renderings>?
    <htd:rendering type="QName">+ ... </htd:rendering>
  </htd:renderings>
  <htd:deadlines>?
    <htd:startDeadline>* ... </htd:startDeadline>
    <htd:completionDeadline>* ... </htd:completionDeadline>
  </htd:deadlines>
</htd:task>
```

WS-HT Coordination Protocol



B4P and WS-HT – Benefits

- BPEL4People
  - Enable unified design tools for automated Web services-based business processes and human-centric processes
  - Enable unified monitoring environments for automated Web services-based business processes and human-centric processes
  - Interoperate and integrate with BPEL4People-aware environments
- WS-HumanTask
  - Leverage UI technology that can consume and execute tasks from different task engines
  - Build and leverage vendor neutral task-aware applications
  - Interoperate and exchange work with other WS-HumanTask-aware environments

Web Service Orchestration and WS-BPEL

- Part I – Web Service Orchestration
  - Motivation and Overview
  - Business Process Modeling Styles
- Part II – The WS-BPEL 2.0 Standard
  - WS-BPEL 2.0 Concepts and Language Elements
- Part III – WS-BPEL 2.0 Extensions
  - WS-BPEL Extension for People (BPEL4People)
  - WS-BPEL Extension for Subprocesses (BPEL-SPE)
  - WS-BPEL Extension for Java (BPELJ)
- Part IV – Additional Related Standards
  - Service Component Architecture (SCA)
  - Business Process Modeling Notation (BPMN)

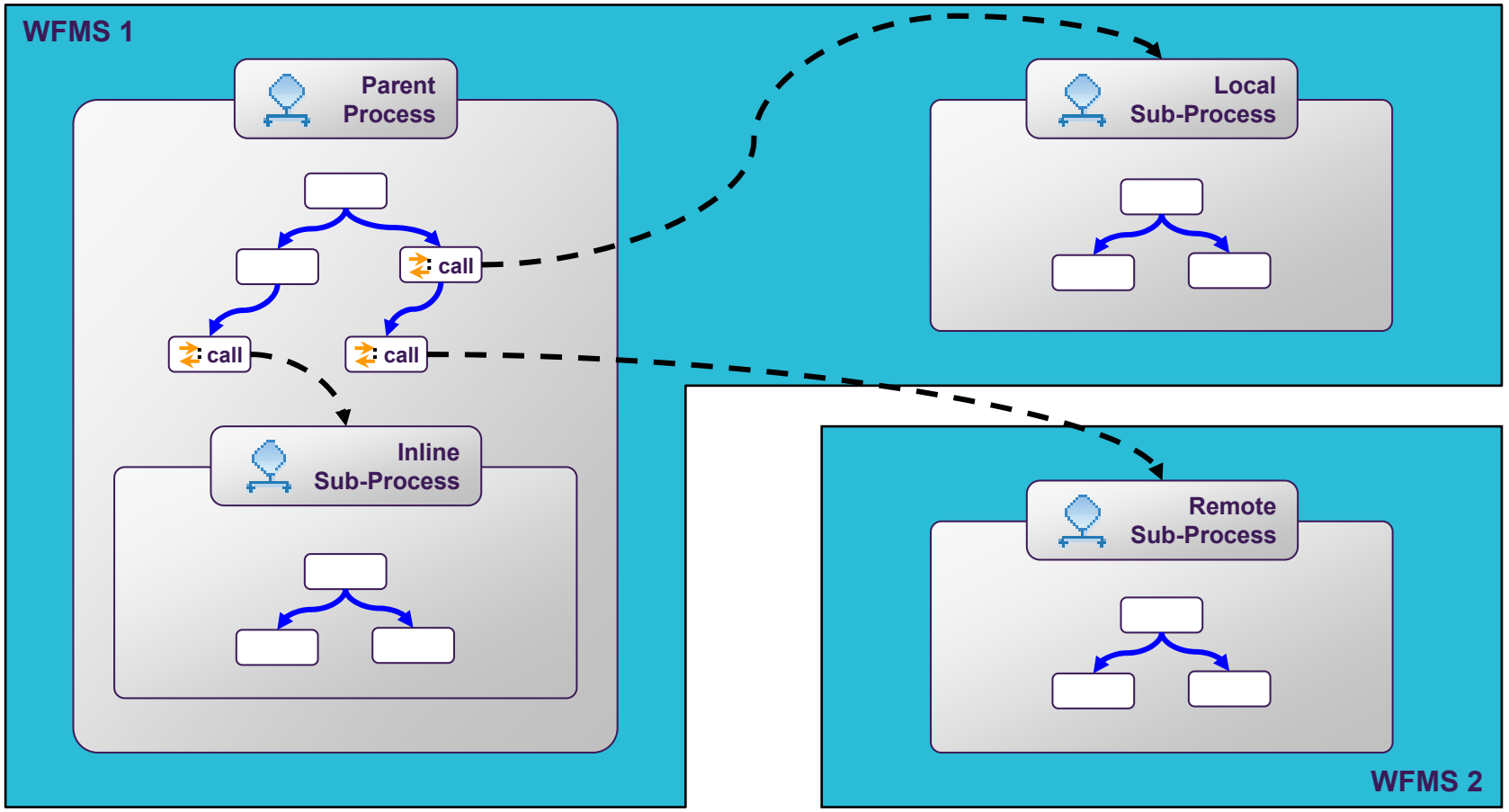
WS-BPEL 2.0 Extension for Sub-Processes

- Designing complex and large business processes requires a language that supports modularization and re-use, in a portable, interoperable way
- Business processes need to invoke other business processes
  - ... within the same process
  - ... within the same infrastructure
  - ... across infrastructures
- Overall managed business process lifecycle
  - Across all involved sub-processes
- BPEL today cannot do that

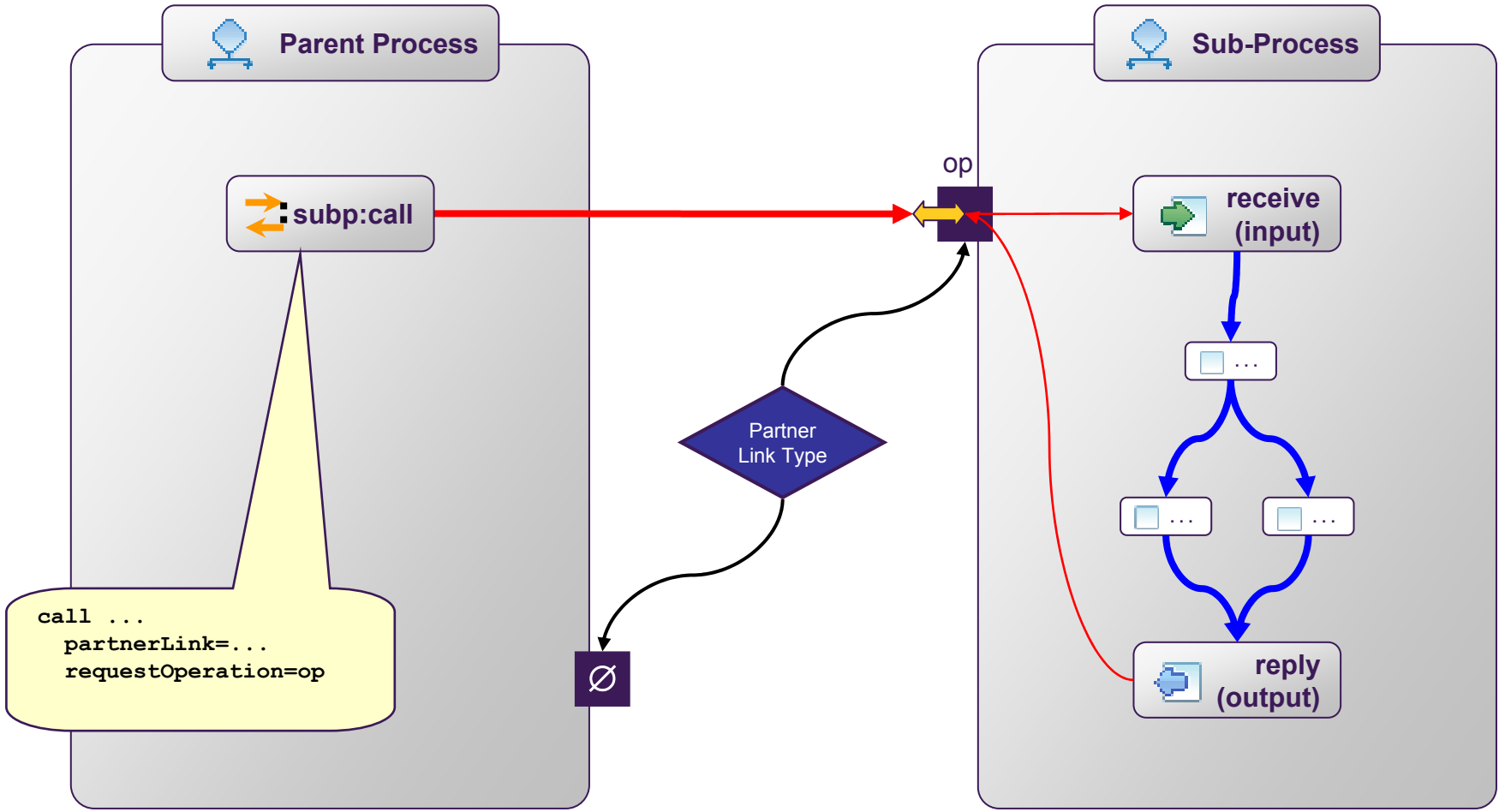
Kinds of Sub-processes

- Inline
  - Sub-process defined within same document as parent process
    - New `<subprocesses>` element as container for inline sub-processes
  - Sub-process may access content of parent process
- Local
  - Sub-process defined as standalone process but managed by same engine that manages the parent process
    - Sub-processes can be called by name because it is known in the environment
    - Lifecycle of sub-process can be managed proprietarily
- Remote
  - Sub-process and parent process are managed by separate engines (of possibly different vendors)
    - Interoperability requires protocol to couple lifecycle of parent process and sub-process

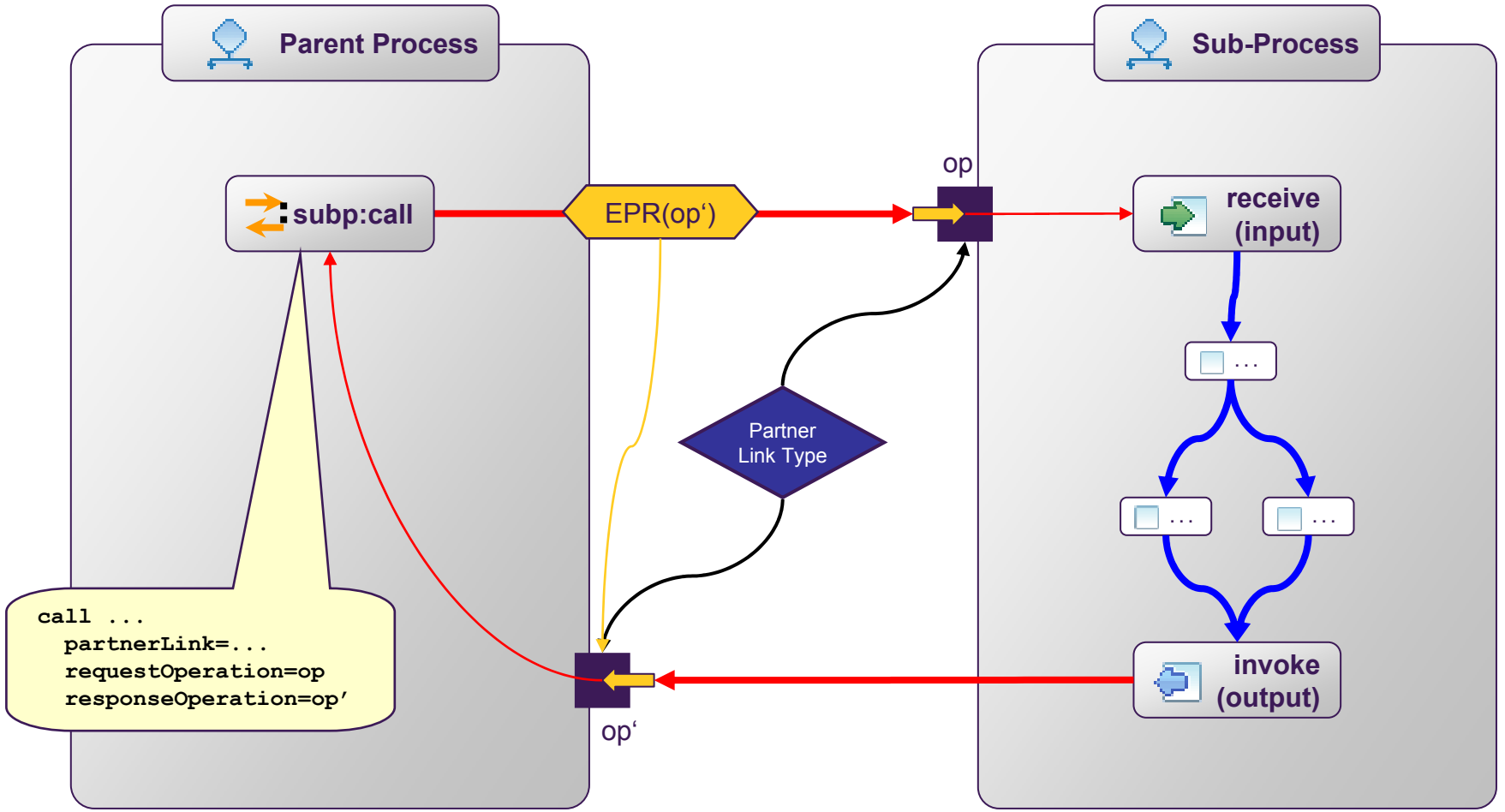
Sub-Processes – Definition and Invocation



Sub-Process Invocation – Request-Response



Sub-Process Invocation – One-Way + Callback



BPEL-SPE Status

- IBM-SAP Whitepaper published in October 2005

  http://www.ibm.com/developerworks/webservices/library/specification/ws-bpelsubproc/

Web Service Orchestration and WS-BPEL

- Part I – Web Service Orchestration
  - Motivation and Overview
  - Business Process Modeling Styles
- Part II – The WS-BPEL 2.0 Standard
  - WS-BPEL 2.0 Concepts and Language Elements
- Part III – WS-BPEL 2.0 Extensions
  - WS-BPEL Extension for People (BPEL4People)
  - WS-BPEL Extension for Subprocesses (BPEL-SPE)
  - WS-BPEL Extension for Java (BPELJ)
- Part IV – Additional Related Standards
  - Service Component Architecture (SCA)
  - Business Process Modeling Notation (BPMN)

WS-BPEL 2.0 Extension for Java

- Add Java support to WS-BPEL 2.0 using the well-defined extension points of WS-BPEL 2.0
- Standardize the proprietary Java capabilities most Java-based BPEL implementations have today
- Provide Portability of BPEL processes that use Java across vendors

BPELJ Technical Goals

- Portability between Java-based BPEL processes from different vendors
- Enable Java and BPEL to cooperate by allowing sections of inline Java code to be included in BPEL process definitions, such as
  - Loop conditions
  - Branching conditions
  - Variable initialization
  - Web service message preparation
  - Logic of business functions
- Make it possible to use BPEL for orchestrating long-running interactions with Java components
- Combine BPEL with Java transaction capabilities (JTA) (Atomic Scopes and Atomic Processes)

BPELJ Examples – Inline Java Expressions

- Condition (Boolean Expression)

```
<if>
   <condition
      expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:java1.4">
      widget.equals(getVariableProperty("PO", "productName"))
   </condition>
      ...
   <else>...</else>
</if>
```

- Variable initialization (General Expression)

```
<variable name="salesTax" type="xsd:float">
   <from
      expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:java1.4">
      subtotal * taxRate
   </from>
</variable>
```

BPELJ Examples – Inline Java Activity

- Extension Activity

```
<process name="purchaseOrderProcess" bpelj:xmlBinding="bpelj:DOM3" ...>
   <variables>
      <variable name="justificationDoc" type="lns:justificationDocument"/>
      <variable name="po" type="lns:POMsg" bpelj:xmlBinding="bpelj:SDO1.0"/>
   </variables>
   <sequence>
      ...
      <extensionActivity>

         <bpelj:snippet>
            // Get the approver using SDO accessor
            Approver approver = po.getApprover();
            // Get the approver's comments
            NodeList commentNodeList =
               justificationDoc.getElementsByTagName("approverComment");
            ...
         </bpelj:snippet>

      </extensionActivity>
   </sequence>
</process>
```

BPELJ Status

- IBM-BEA Whitepaper published in March 2004
  http://www.ibm.com/developerworks/library/specification/ws-bpelj/
- Specification completed (October 2005)

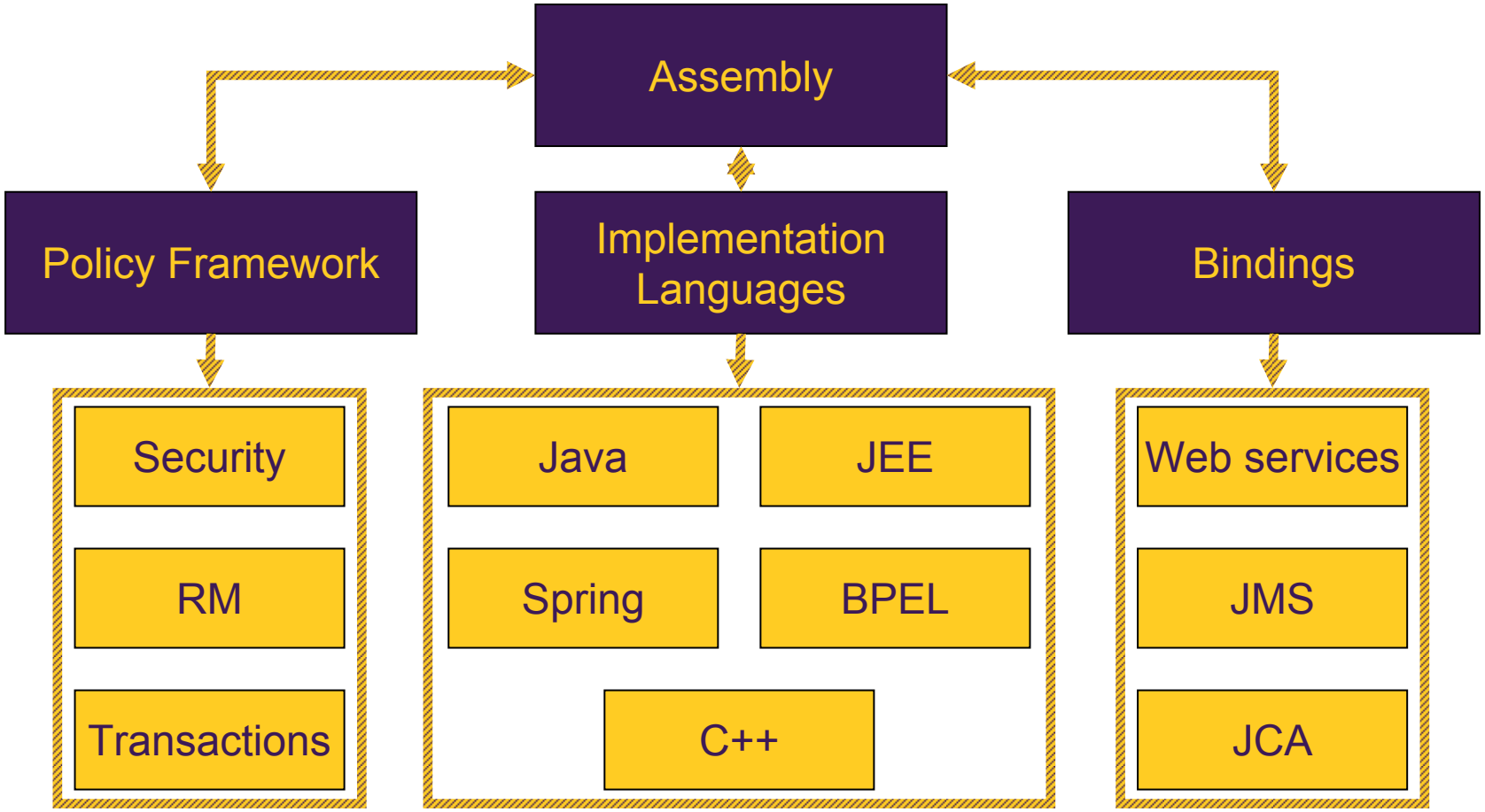
Web Service Orchestration and WS-BPEL

- Part I – Web Service Orchestration
  - Motivation and Overview
  - Business Process Modeling Styles
- Part II – The WS-BPEL 2.0 Standard
  - WS-BPEL 2.0 Concepts and Language Elements
- Part III – WS-BPEL 2.0 Extensions
  - WS-BPEL Extension for People (BPEL4People)
  - WS-BPEL Extension for Subprocesses (BPEL-SPE)
  - WS-BPEL Extension for Java (BPELJ)
- Part IV – Additional Related Standards
  - Service Component Architecture (SCA)
  - Business Process Modeling Notation (BPMN)

SCA Objectives

- Simplified Programming Model for SOA
  - Building service components
  - Assembling components into applications
  - Deploying to (distributed) runtime environments
- Service components built from new or existing code using SOA principles
  - Vendor-neutral – supported across the industry
  - Language-neutral – components written using any language
  - Technology-neutral – use any communication protocols and infrastructure to link components

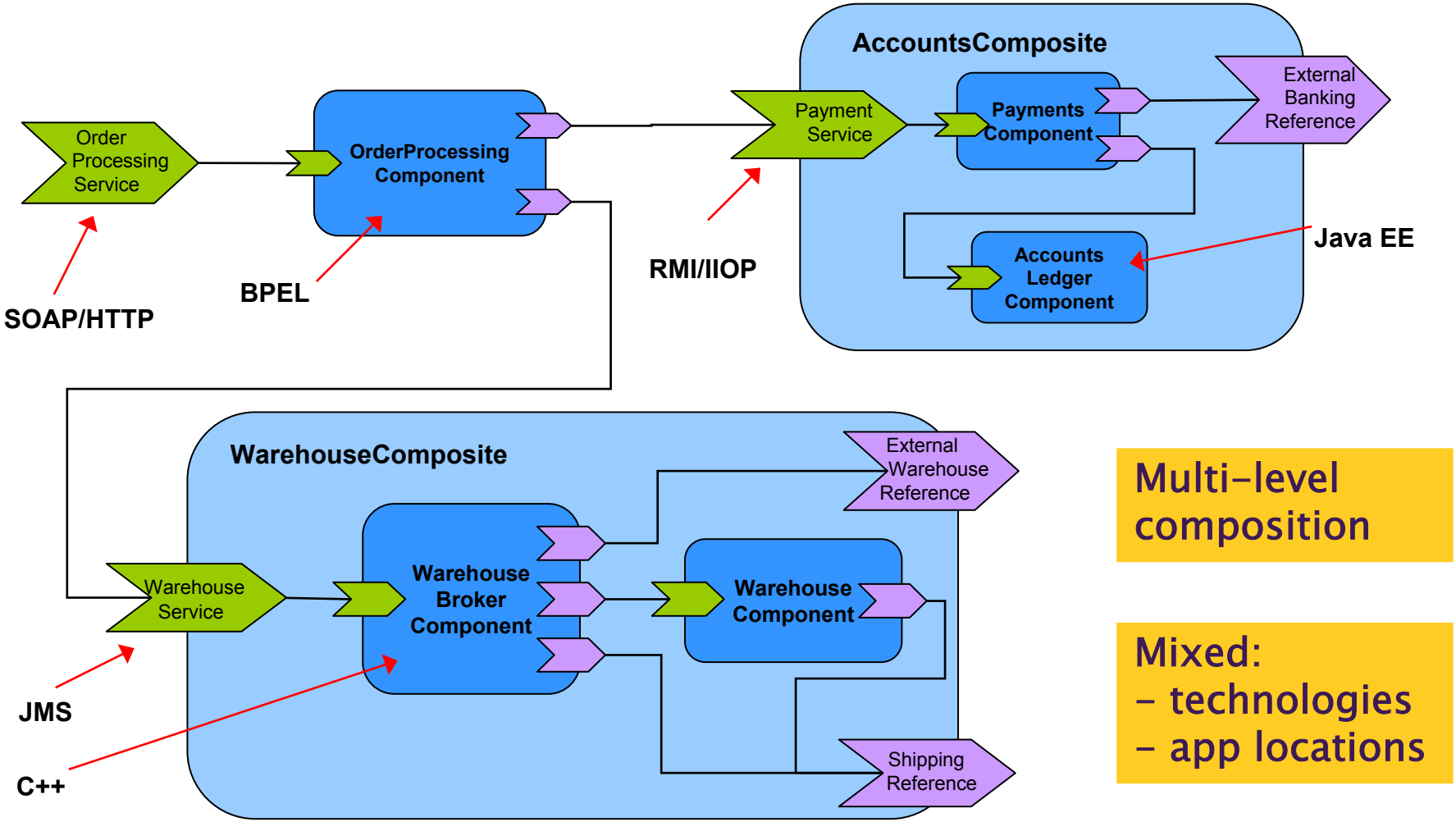
SCA Specifications



SCA Technical Committees

- Open CSA Member Section
  - Coordination between TCs
- Technical Committees
  - SCA Assembly TC
  - SCA Bindings TC
  - SCA Policy TC
  - SCA J TC
  - SCA C-C++ TC
  - SCA BPEL TC

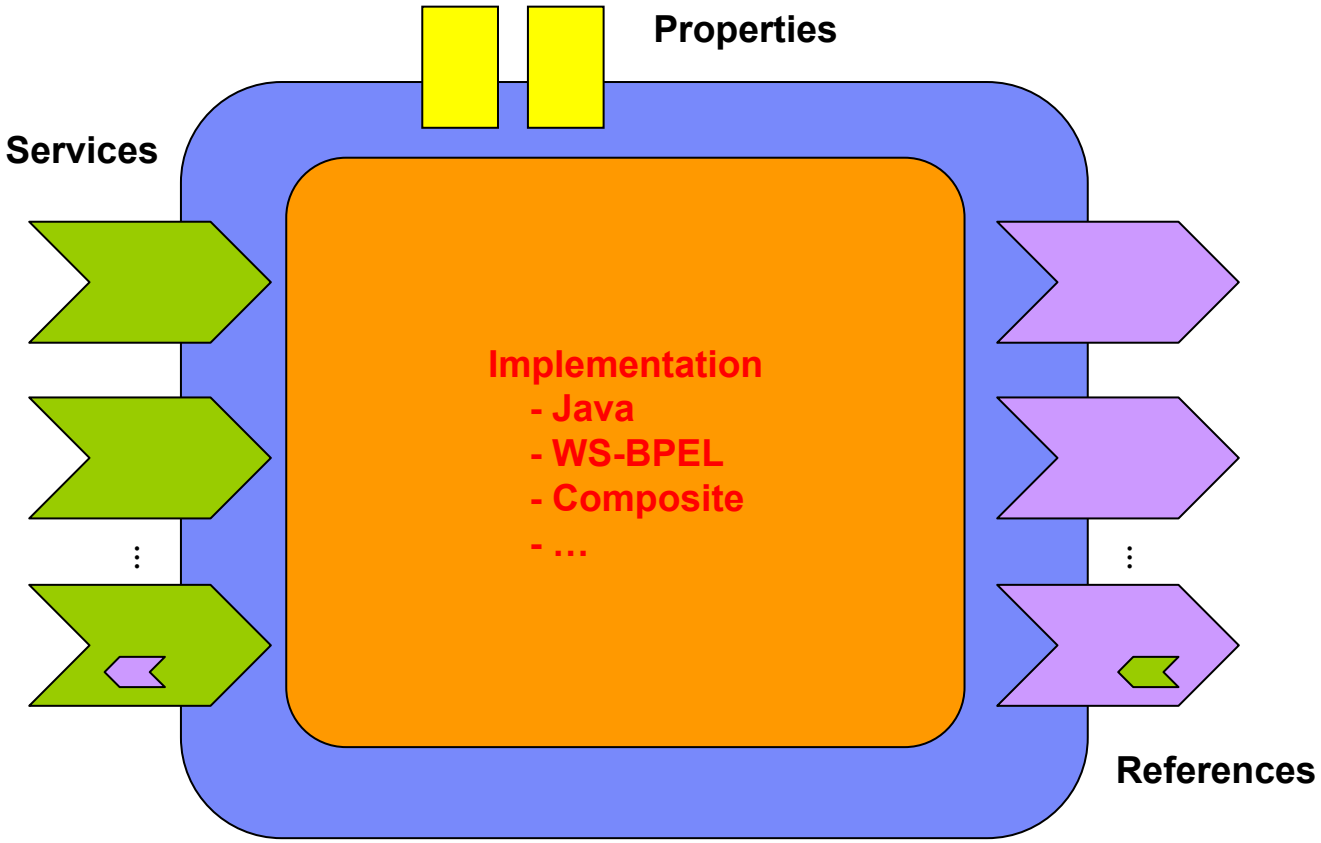
SCA Assembly Model



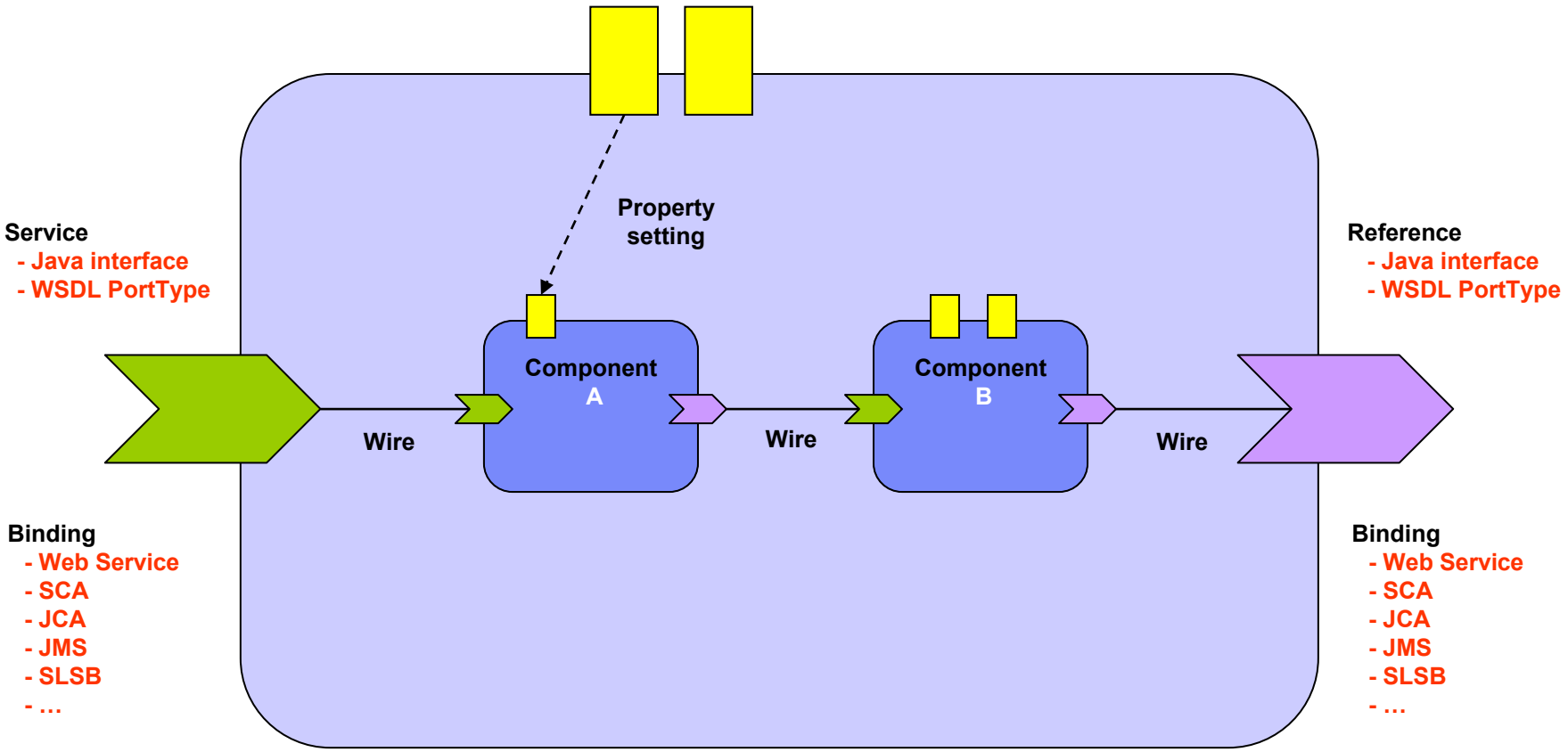
Multi-level composition

Mixed:
- technologies
- app locations

SCA Assembly – Component



SCA Assembly – Composite



SCA Local & Remotable Services

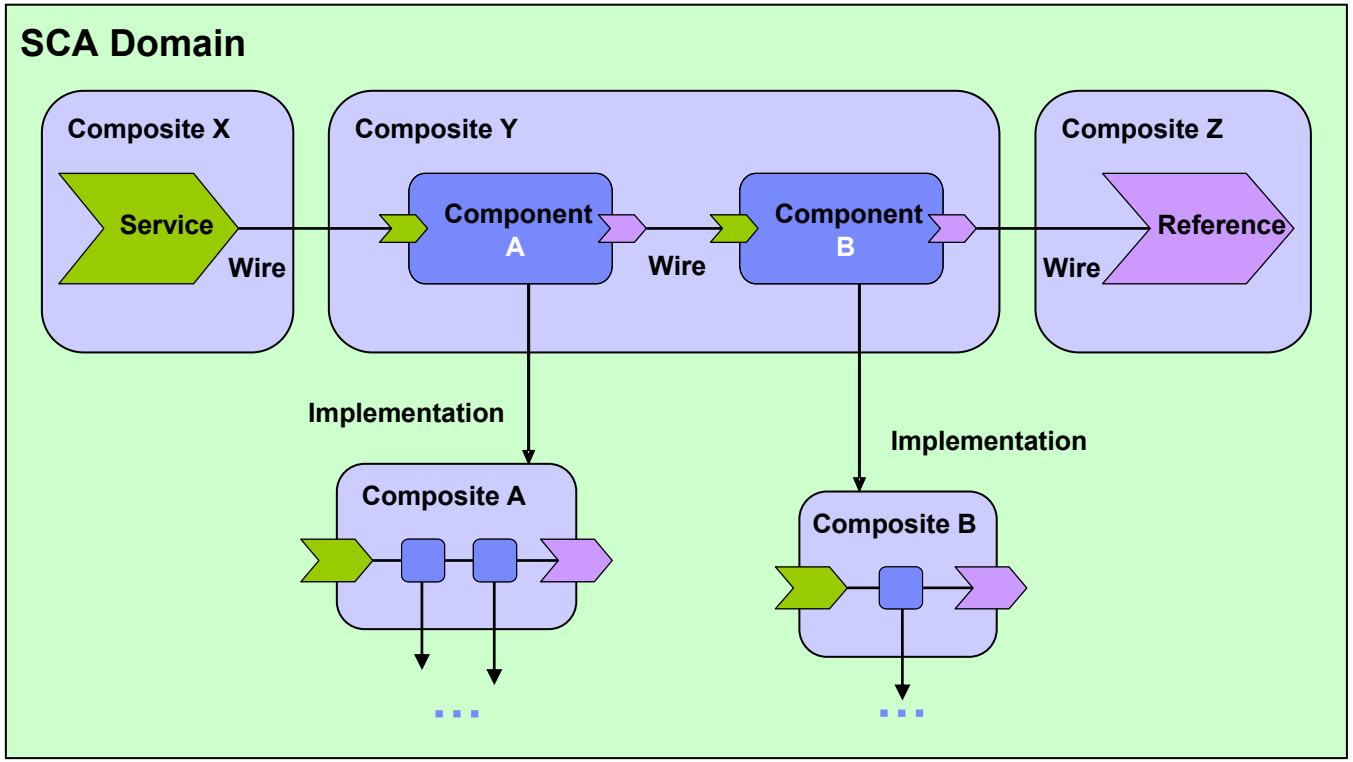
- Local-only services
  - Fine-grained, tightly coupled interfaces
  - Interface may be language specific (e.g. Java or C++)
  - Parameters & return values by-reference
  - Client must be local (same address space)
- Remotable services
  - Coarse-grained, loosely coupled interfaces
  - Interface must be representable using WSDL (possibly generated)
  - Parameters & return values by-value
  - Client may be remote

SCA Bidirectional Services

- Bidirectional interfaces
  - Service interface for outbound calls
  - Callback interface implemented by client

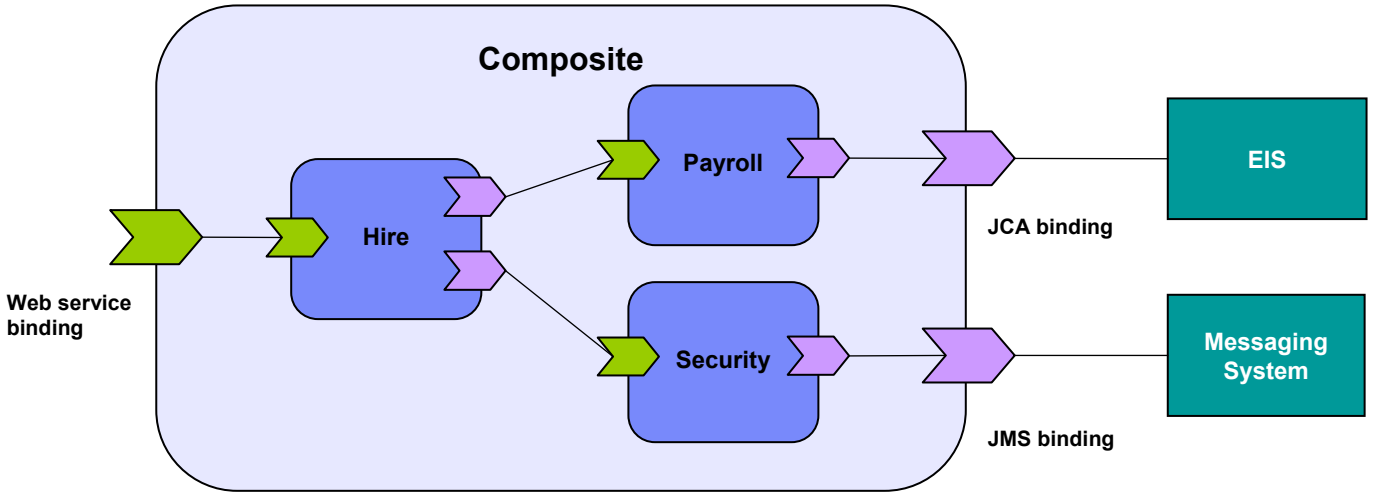
```
@Callback(ApprovalCallback.class)
public interface LoanApproval {
  public void approve( LoanApplication application );
}

public interface ApprovalCallback {
  public void answer( String approval );
}
```

SCA Assembly – Deployment



SCA Bindings

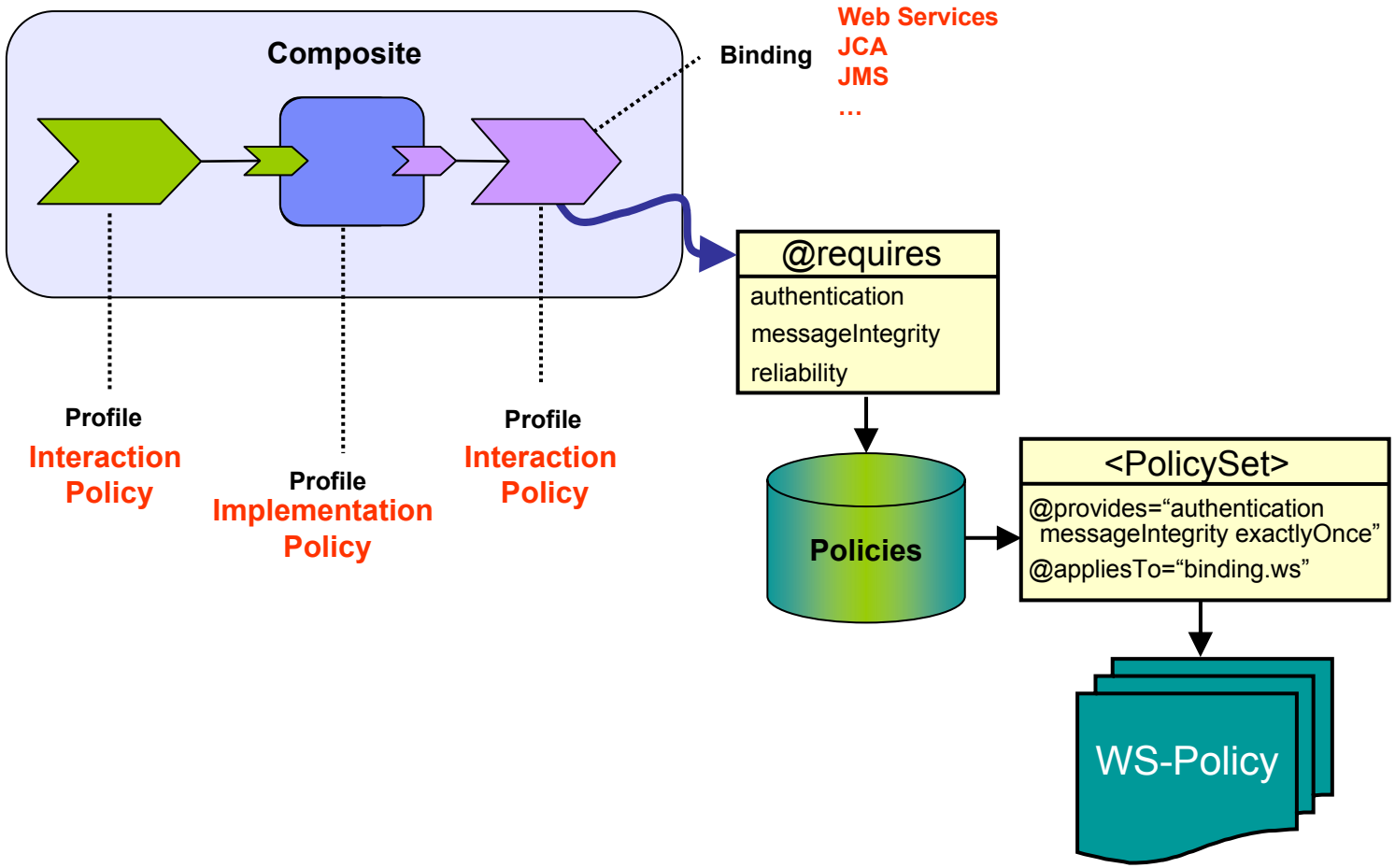
- Binding specifications
  - Web services, JMS, JCA
  - Potential additional binding specifications
    - HTTP with REST and other patterns
    - FTP, SMTP, others



SCA Policy-Framework

- Intents
  - Abstract QoS requirements
  - Intents for security, reliability and transactionality
  - SCA Implementations can define other intents
- Policy sets
  - Map intents to policies
- Usage of intents and policy sets
  - Interaction policies
  - Implementation policies

SCA Policy-Framework



SCA Intents Provided by Bindings

- The binding technology can satisfy intents
  - E.g. JMS supports reliable messaging
- Binding type declarations specify intents satisfied
  - @type = the element name used for the binding
  - @mayProvide = intents that are provided only if the service or reference requires it
  - @alwaysProvides = intents that are always provided

```
<bindingType
    type="sca:binding.jms"
    mayProvide="sca:atLeastOnce sca:atMostOnce sca:ordered"
    alwaysProvides="sca:jms" />
```

- The intents provided by a binding are vendor specific

SCA Java

- Component implementation types
  - Java POJOs (implementation.java)
  - Spring Beans (implementation.spring)
  - EJB (implementation.ejb)
- Common annotations and API
  - Metadata service contracts based on Java interfaces
  - Metadata component type information
  - Metadata container-component implementation contract
  - Asynchronous interactions and callbacks
  - APIs (few) for accessing assembly information

SCA Java Component

```
@Remotable
public interface AccountService{

   public AccountReport getAccountReport( String customerID );

}

public class AccountServiceImpl implements AccountService {

   @Property
   private String currency = "USD";

   @Reference
   private AccountUpgradeService accountUpgradeService;
   @Reference
   private StockQuoteService stockQuoteService;

   public AccountReport getAccountReport( String customerID ) {
      ...
   }
   ...
}
```

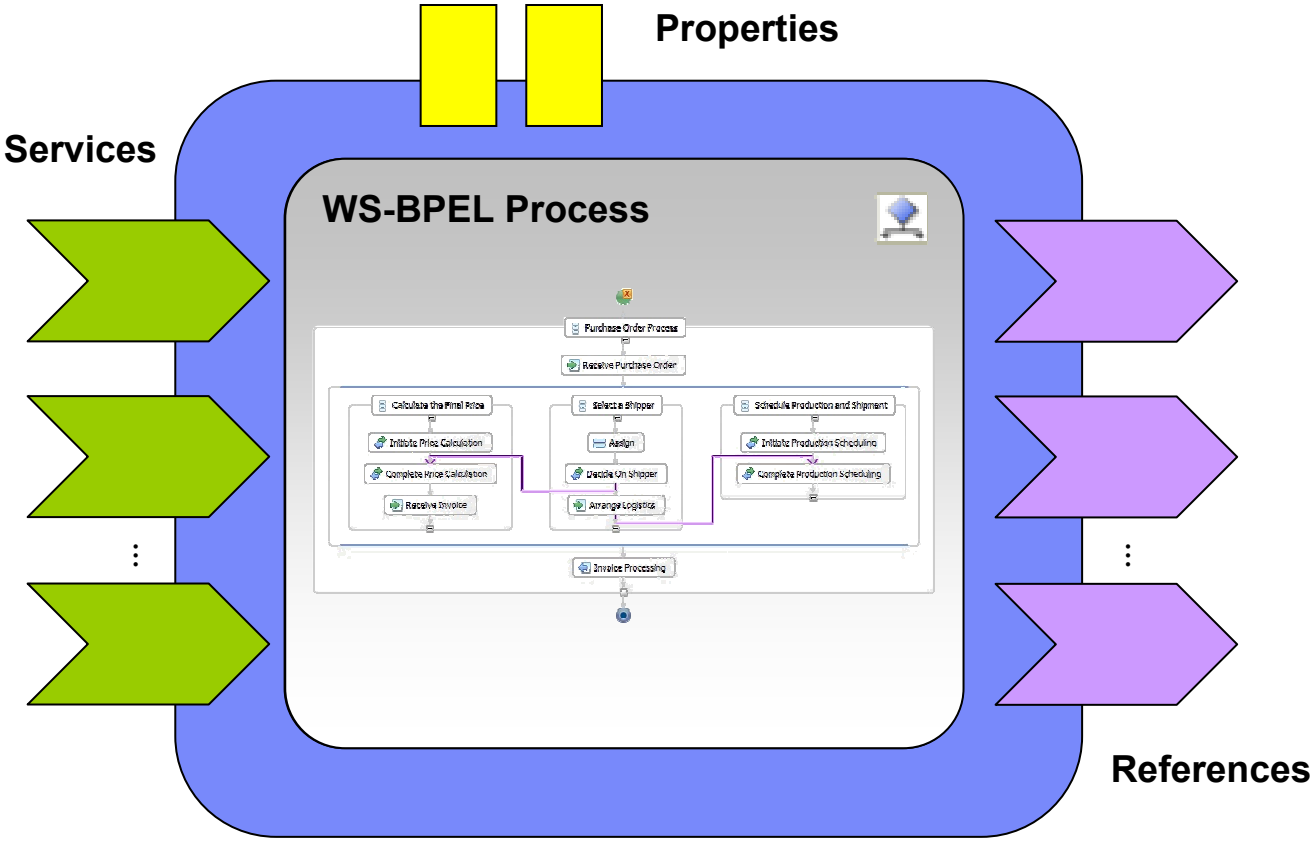
SCA C / C++

- Component implementation types
  - C++ classes (implementation.cpp)
  - Set of C functions (implementation.c)
- Annotations and API
  - Metadata service contracts
  - Metadata component type information
  - Metadata container-component implementation contract
  - Asynchronous interactions and callbacks
  - APIs (few) for accessing assembly information

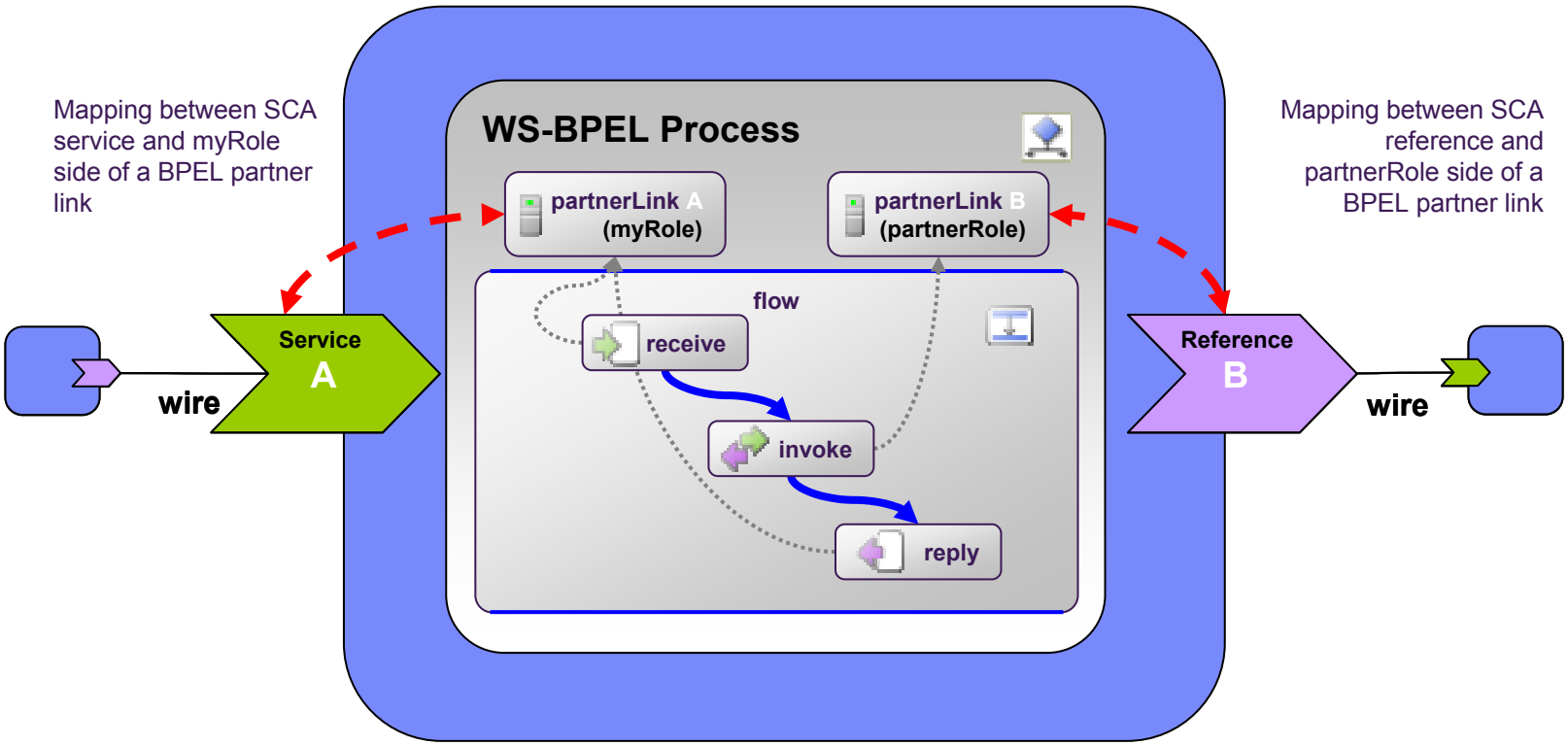
SCA BPEL

- Component implementation type
  - WS-BPEL 2.0 business process (implementation.bpel)
  - Optionally support BPEL4WS 1.1 processes
- Use cases
  - Use any valid WS-BPEL 2.0 process definition as the implementation of a component within SCA
  - Use WS-BPEL to implement any SCA Component Type that uses only WSDL interfaces to define services and references
    - Optionally use SCA specific extensions within the process definition
  - Create a WS-BPEL process definition that uses SCA extensions and generate an SCA Component Type and use that type within an SCA assembly

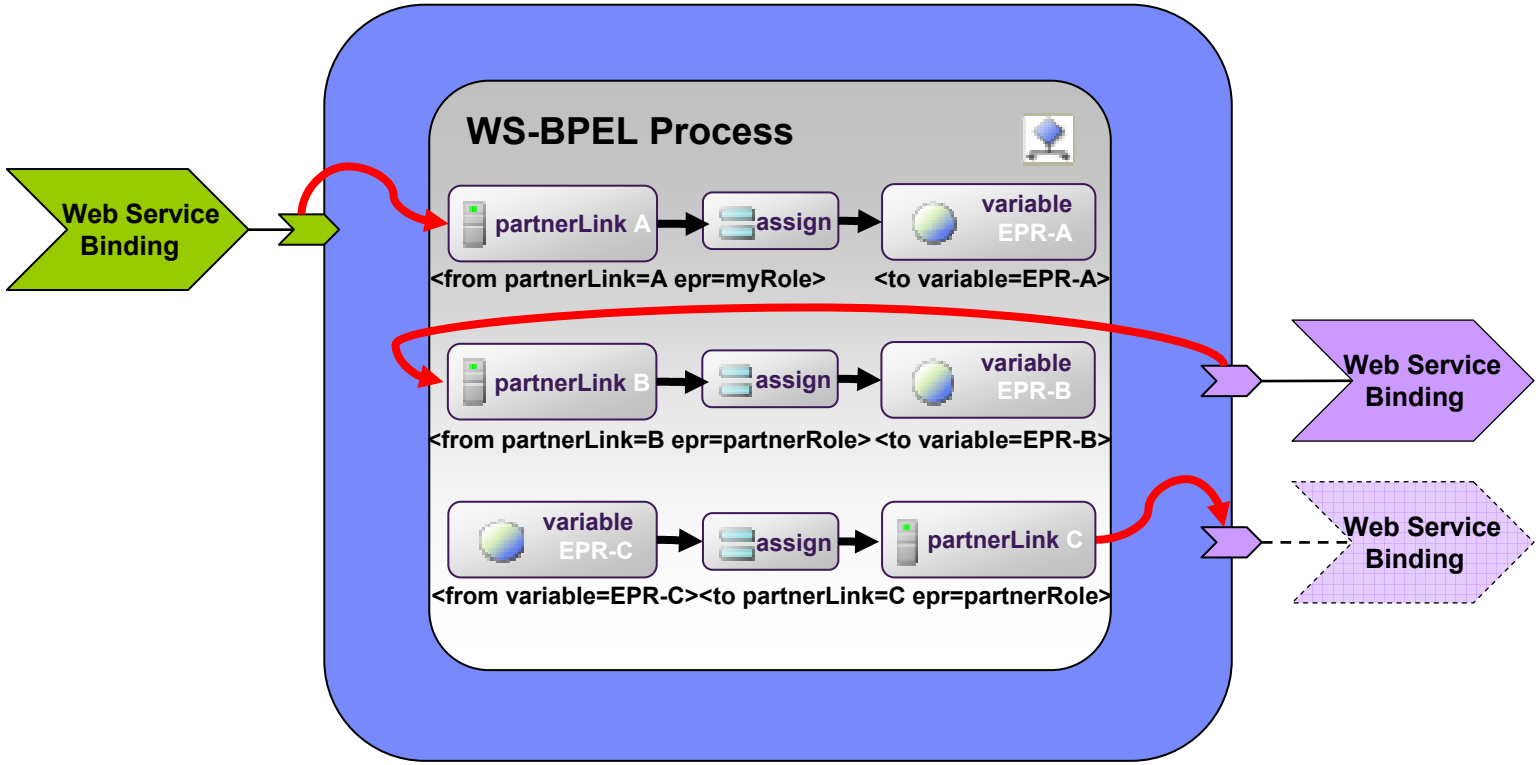
SCA BPEL Component



SCA Services and References



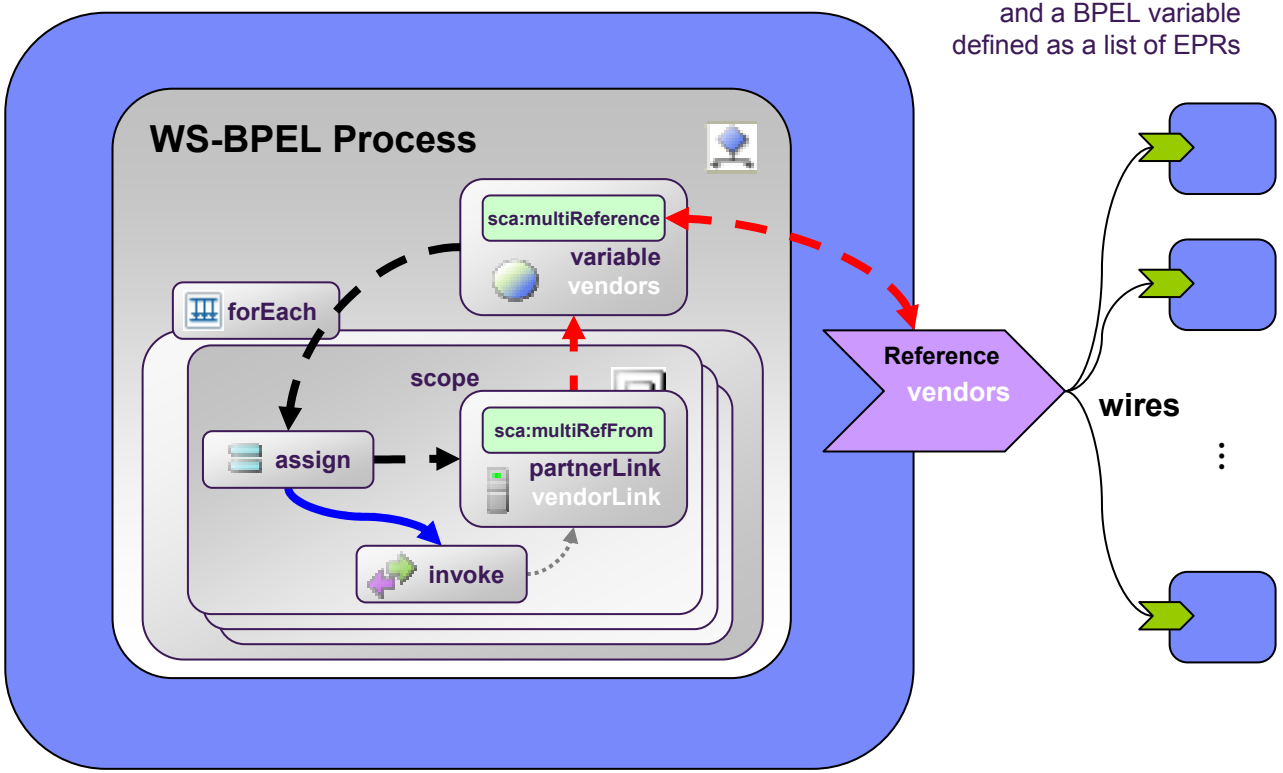
SCA BPEL EPR Assignment



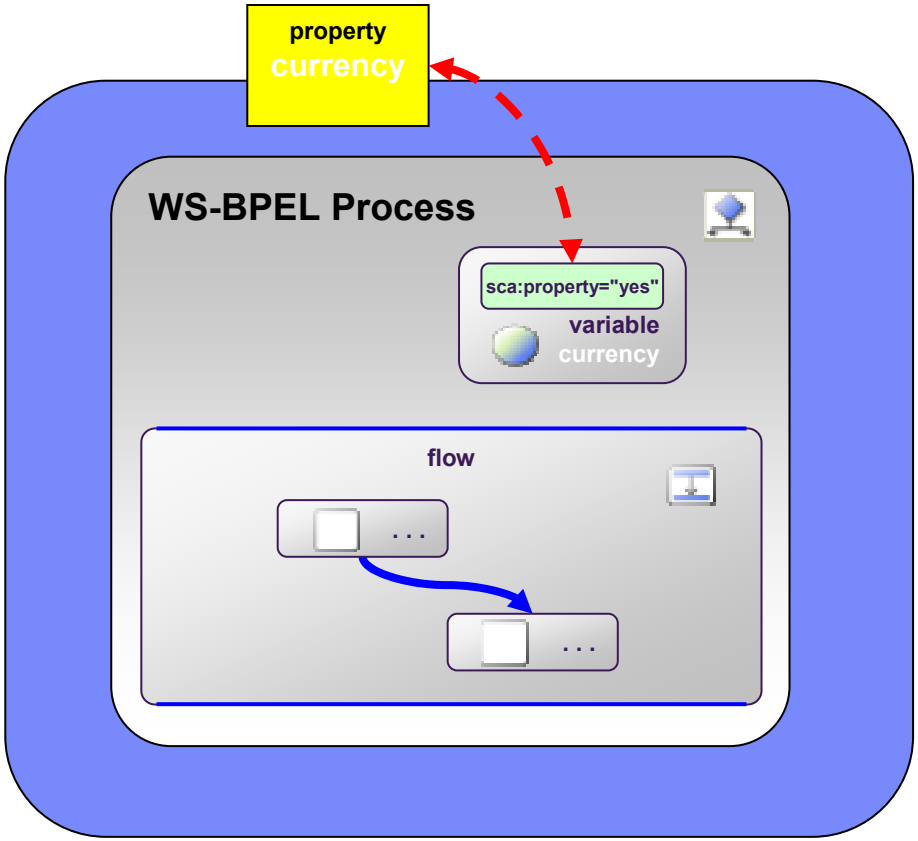
SCA Multi-Valued References

Modeling pattern using a parallel forEach activity with a local partner link

Mapping between SCA multi-valued reference and a BPEL variable defined as a list of EPRs



SCA Properties



Mapping between SCA property and a BPEL variable defined with the same name and type

Complementary Technologies

- Similarities between SCA and WS-BPEL
  - Formal language based on XML
  - Describe a business service implemented by composing together other business services
  - Describe service interactions types by WSDL port types
- SCA describes the structure of an application
  - Components within the business application
  - Services offered by components
  - Service references components depend on
  - Connections between components
  - Endpoint addresses and communication methods
  - Policies applied to components and to the connections between them
- WS-BPEL describes the logic of a business process
  - Sequences of operations which are performed to execute an individual business process
  - Services provided and consumed through partnerLinks – abstract interfaces which must be connected to actual endpoints and communication methods through configuration

SCA Resources

- OASIS Open CSA
  http://www.oasis-opencsa.org/
- Apache Tuscany
  http://tuscany.apache.org/
- Open SOA Collaboration
  http://osoa.org/display/Main/Home
- SCA V1 specifications
  http://osoa.org/display/Main/Service+Component+Architecture+Specifications
- Papers and other SCA information
  http://osoa.org/display/Main/SCA+Resources
- OASIS Webinar downloads
  http://www.oasis-open.org/events/webinars/

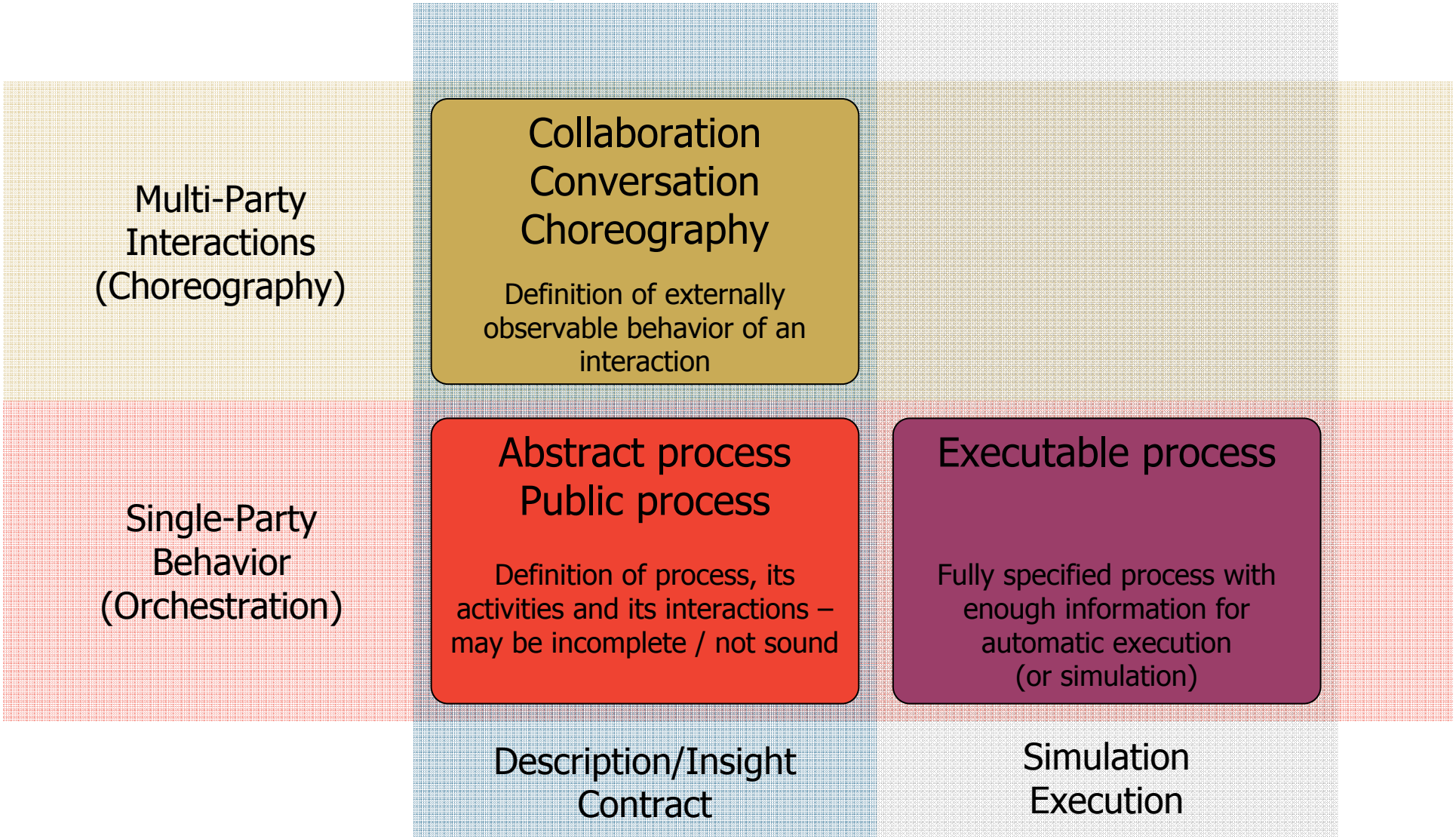
Web Service Orchestration and WS-BPEL

- Part I – Web Service Orchestration
  - Motivation and Overview
  - Business Process Modeling Styles
- Part II – The WS-BPEL 2.0 Standard
  - WS-BPEL 2.0 Concepts and Language Elements
- Part III – WS-BPEL 2.0 Extensions
  - WS-BPEL Extension for People (BPEL4People)
  - WS-BPEL Extension for Subprocesses (BPEL-SPE)
  - WS-BPEL Extension for Java (BPELJ)
- Part IV – Additional Related Standards
  - Service Component Architecture (SCA)
  - Business Process Modeling Notation (BPMN)

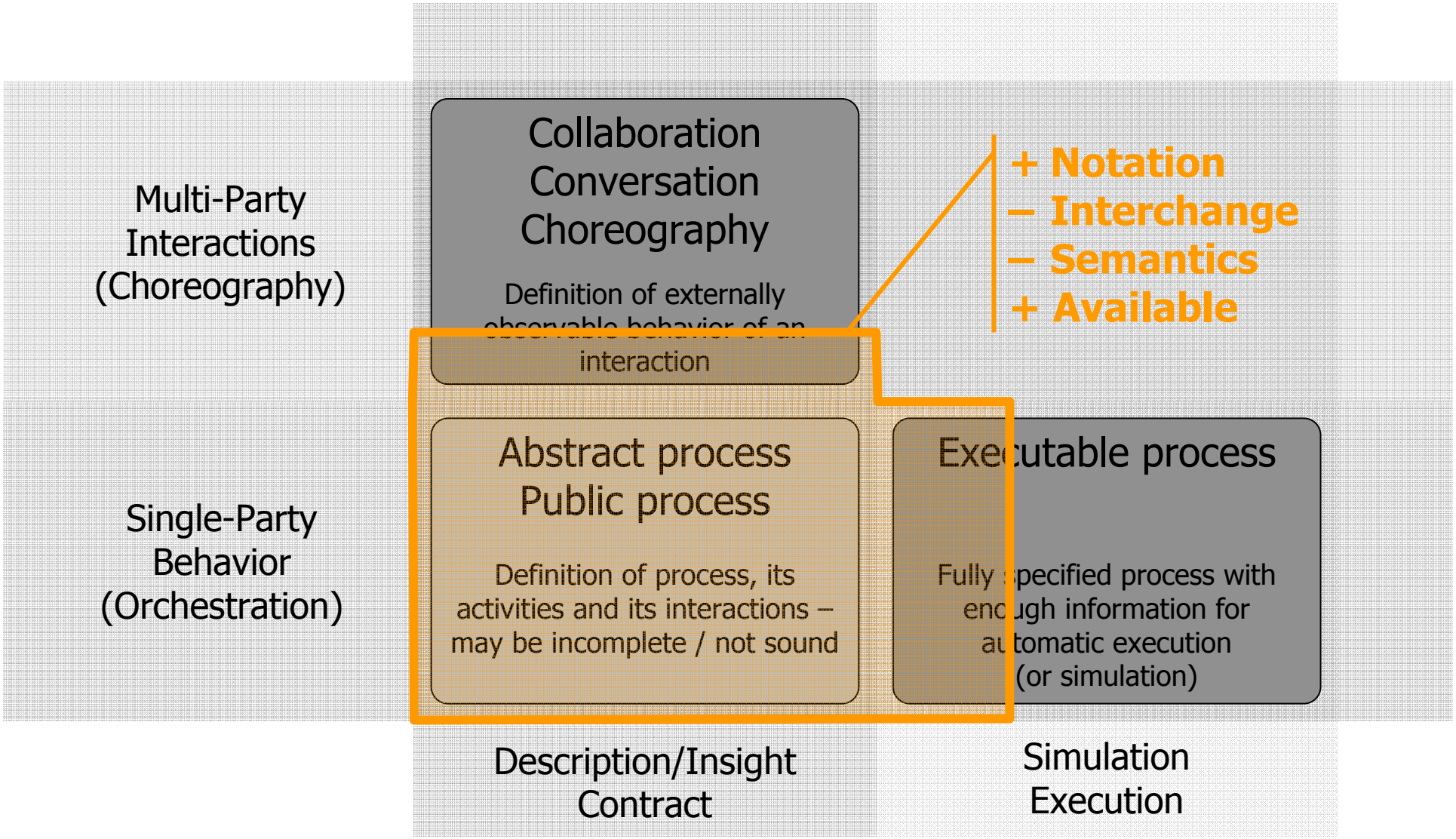
BPMN

- BPMN 1.x
  - Business Process Modeling Notation
  - Current version 1.2 (January 2009)
    - http://www.omg.org/spec/BPMN/1.2/
- BPMN 2.0
  - Business Process Model and Notation
  - In RFP process

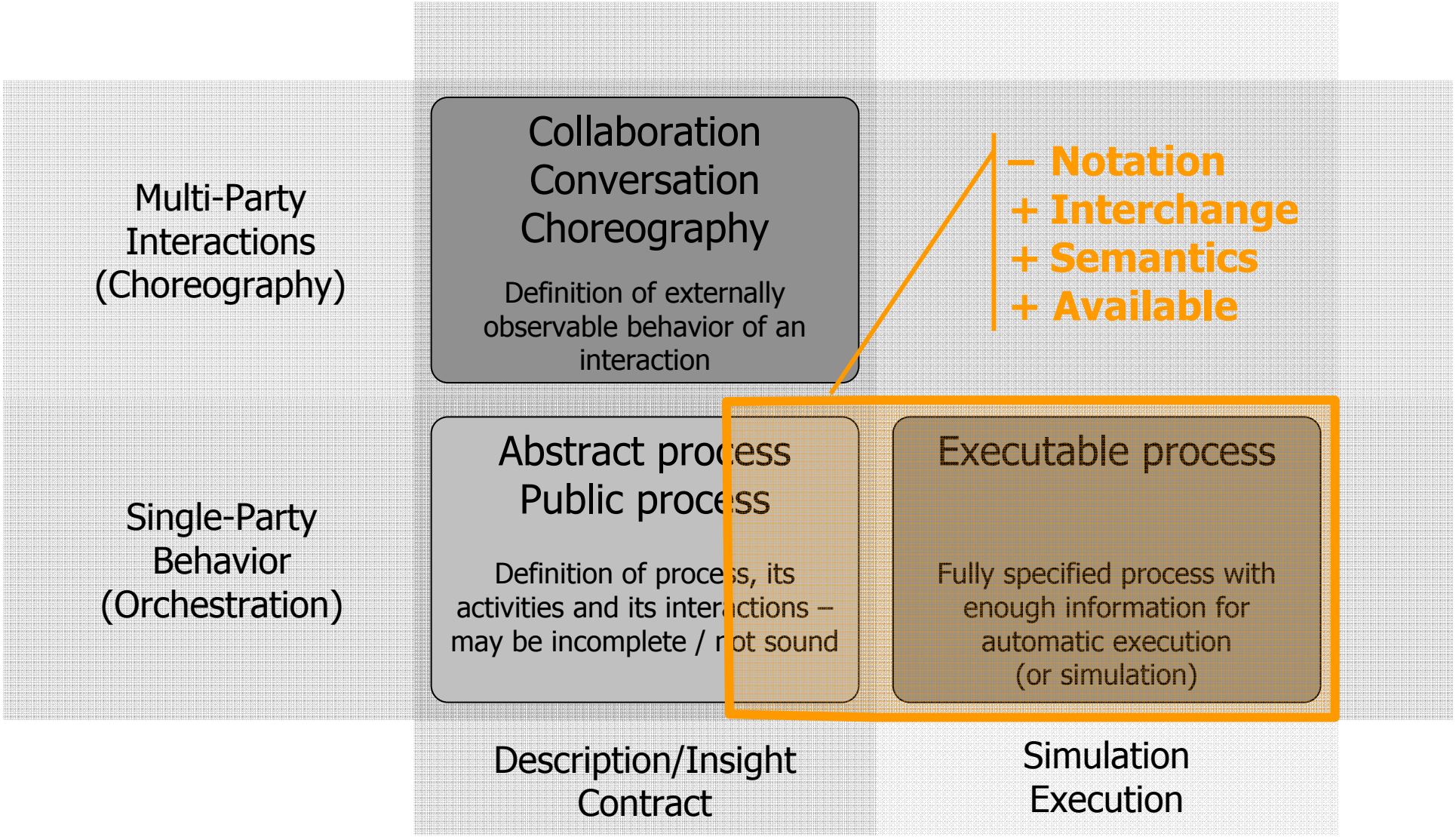
The BPM Modeling Space



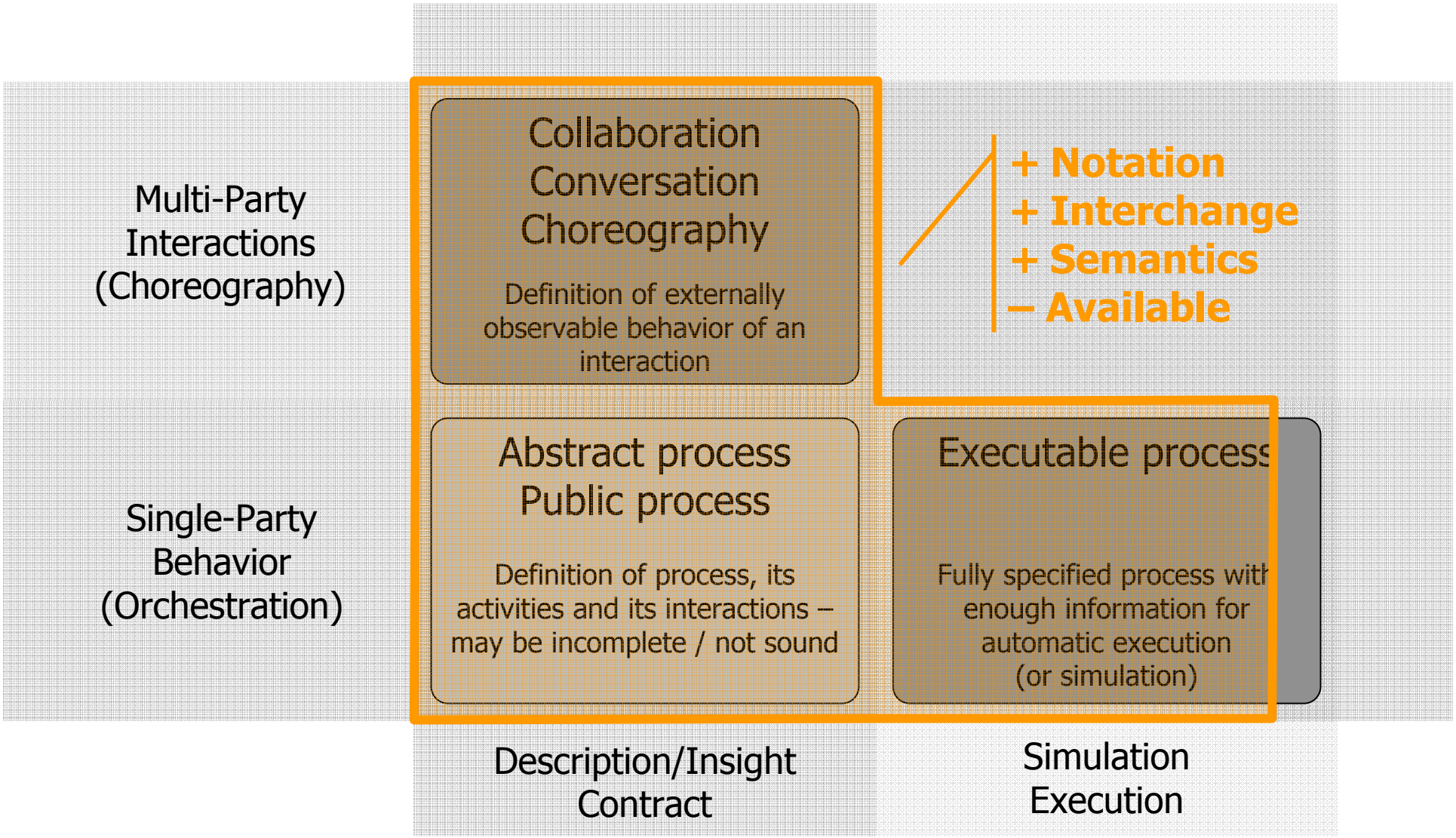
BPMN 1.1


Multi-Party Interactions (Choreography)
Single-Party Behavior (Orchestration)
Collaboration
Conversation
Choreography
Definition of externally observable behavior of an interaction
+ Notation
– Interchange
– Semantics
+ Available
Abstract process
Public process
Definition of process, its activities and its interactions – may be incomplete / not sound
Executable process
Fully specified process with enough information for automatic execution (or simulation)
Description/Insight Contract
Simulation Execution

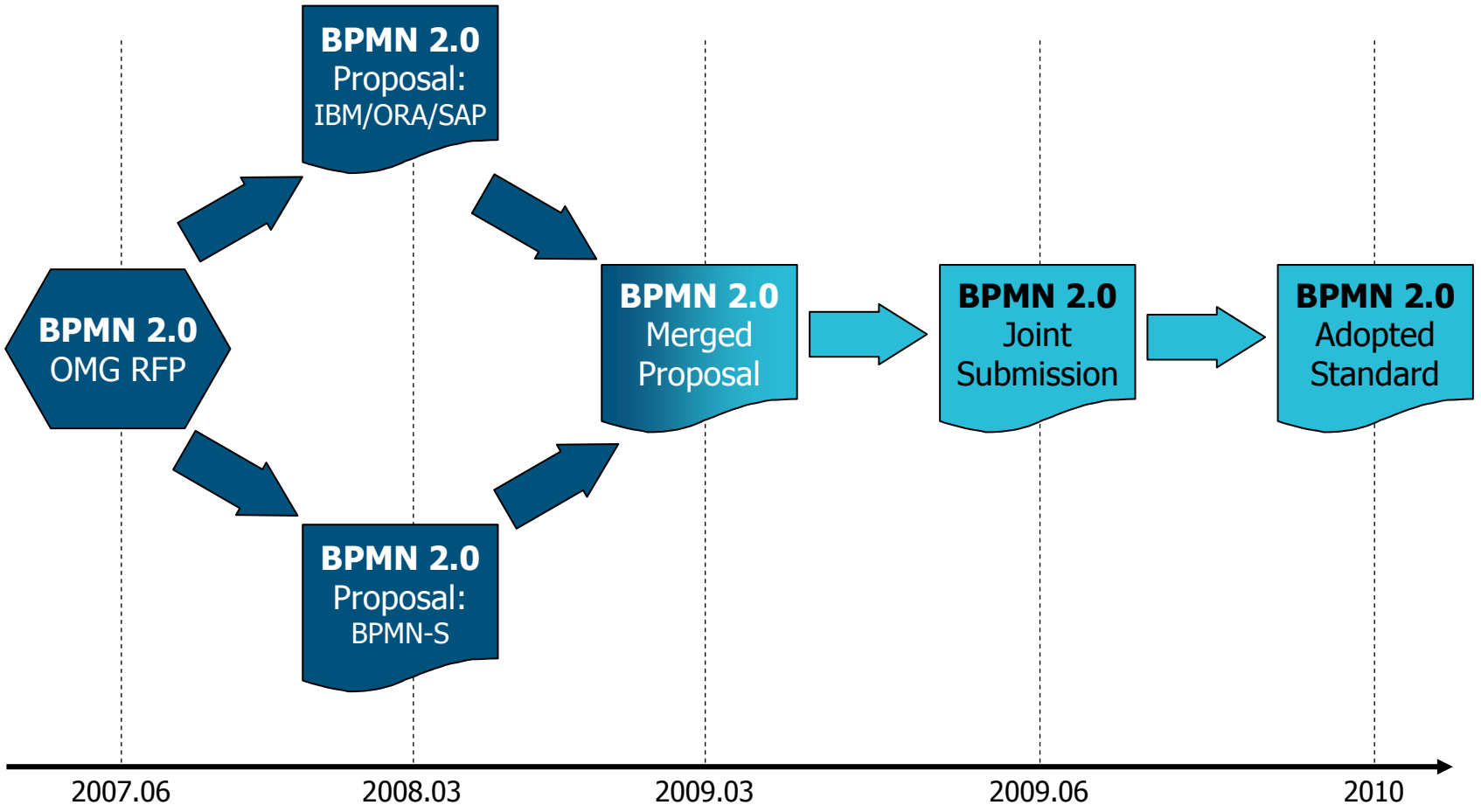
WS-BPEL 2.0


Multi-Party Interactions (Choreography)
Single-Party Behavior (Orchestration)
Collaboration
Conversation
Choreography
Definition of externally observable behavior of an interaction
– Notation
+ Interchange
+ Semantics
+ Available
Abstract process
Public process
Definition of process, its activities and its interactions – may be incomplete / not sound
Executable process
Fully specified process with enough information for automatic execution (or simulation)
Description/Insight Contract
Simulation Execution

BPMN 2.0


Multi-Party
Interactions
(Choreography)
Single-Party
Behavior
(Orchestration)
Collaboration
Conversation
Choreography
Definition of externally observable behavior of an interaction
Abstract process
Public process
Definition of process, its activities and its interactions – may be incomplete / not sound
Executable process
Fully specified process with enough information for automatic execution (or simulation)
+ Notation
+ Interchange
+ Semantics
– Available
Description/Insight
Contract
Simulation
Execution

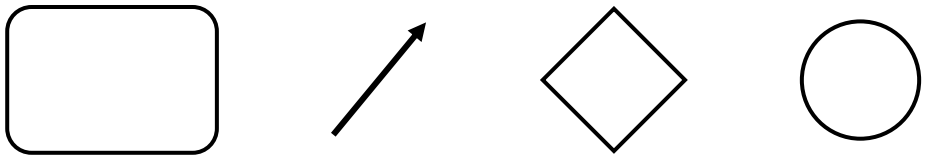
BPMN 2.0 Timeline

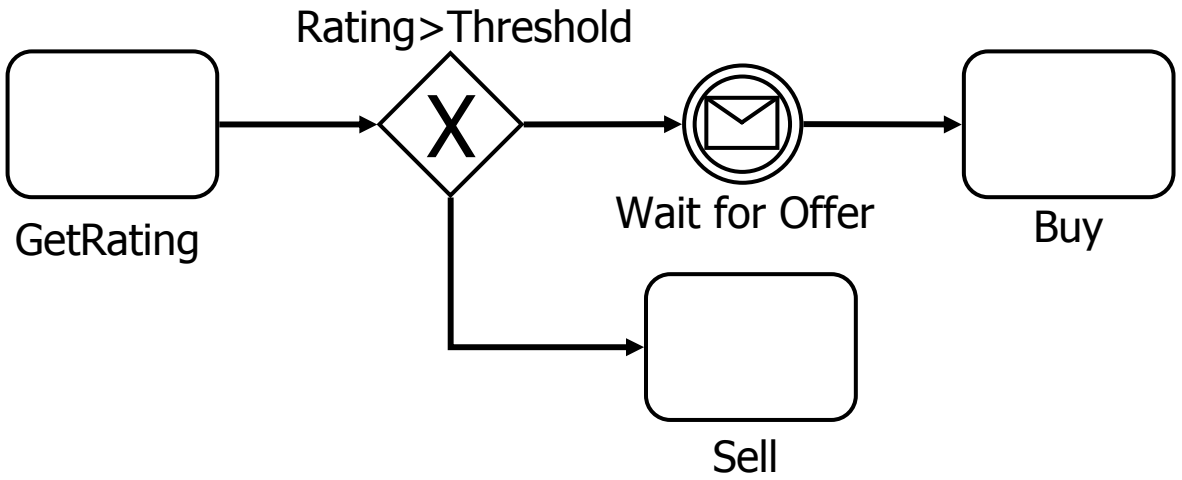


BPMN 2.0 Conformance Classes

- Process Modeling Conformance
  - Support modeling of processes and collaborations
  - Corresponds roughly to the BPMN 1.1 set of capabilities
- Choreography Modeling Conformance
  - Support modeling of choreographies
- Process Execution Conformance
  - Support execution of BPMN processes according to defined execution semantics
- BPEL Process Execution Conformance
  - Support execution of BPMN processes by mapping to BPEL as per the defined BPMN-BPEL mapping.

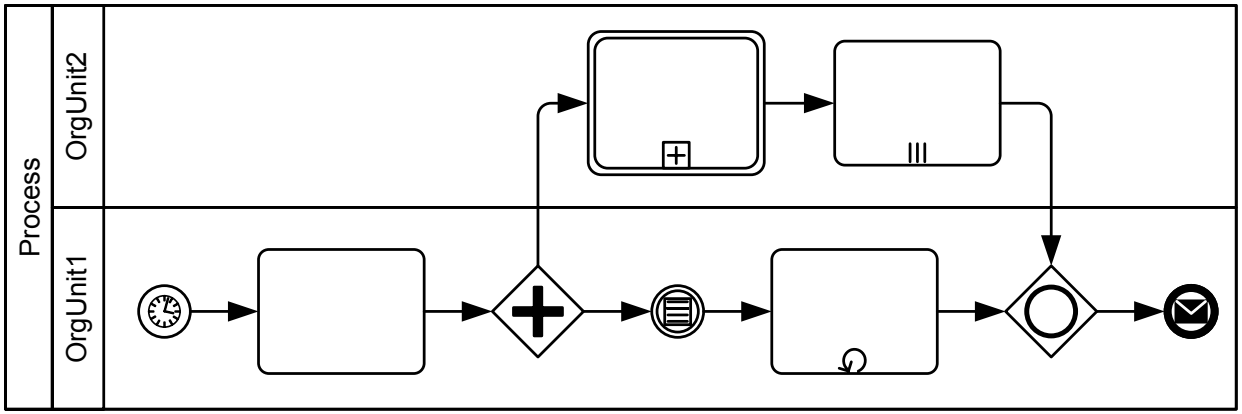
Business Process Modeling Notation





Overall Structure of a BPMN Process

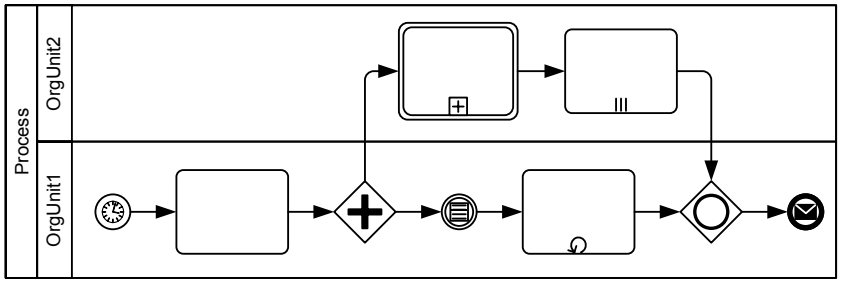
- Lanes
- Activities
  - Structured
  - Atomic → Tasks
- Events
  - Inbound
  - Outbound
- Ordering
  - Sequence Flow
  - Gateways
  - “Adornments”



Lanes

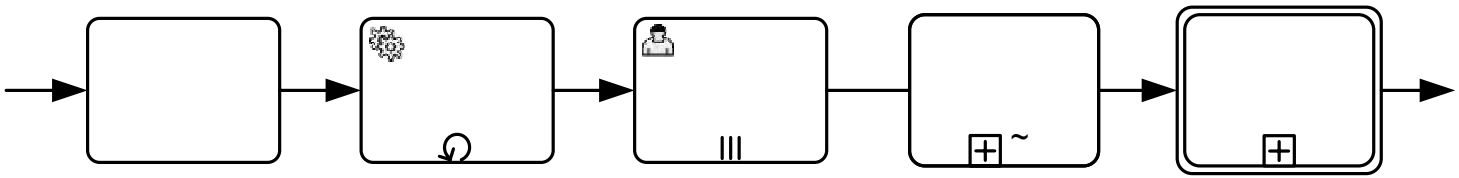
- Sub-partition of a Process
- Used to organize and categorize activities
- Typically represent an organizational unit of an enterprise

- Lanes are a visual construct only
- 2.0 Can be used to structure visualization of activities by *any* suitable attribute

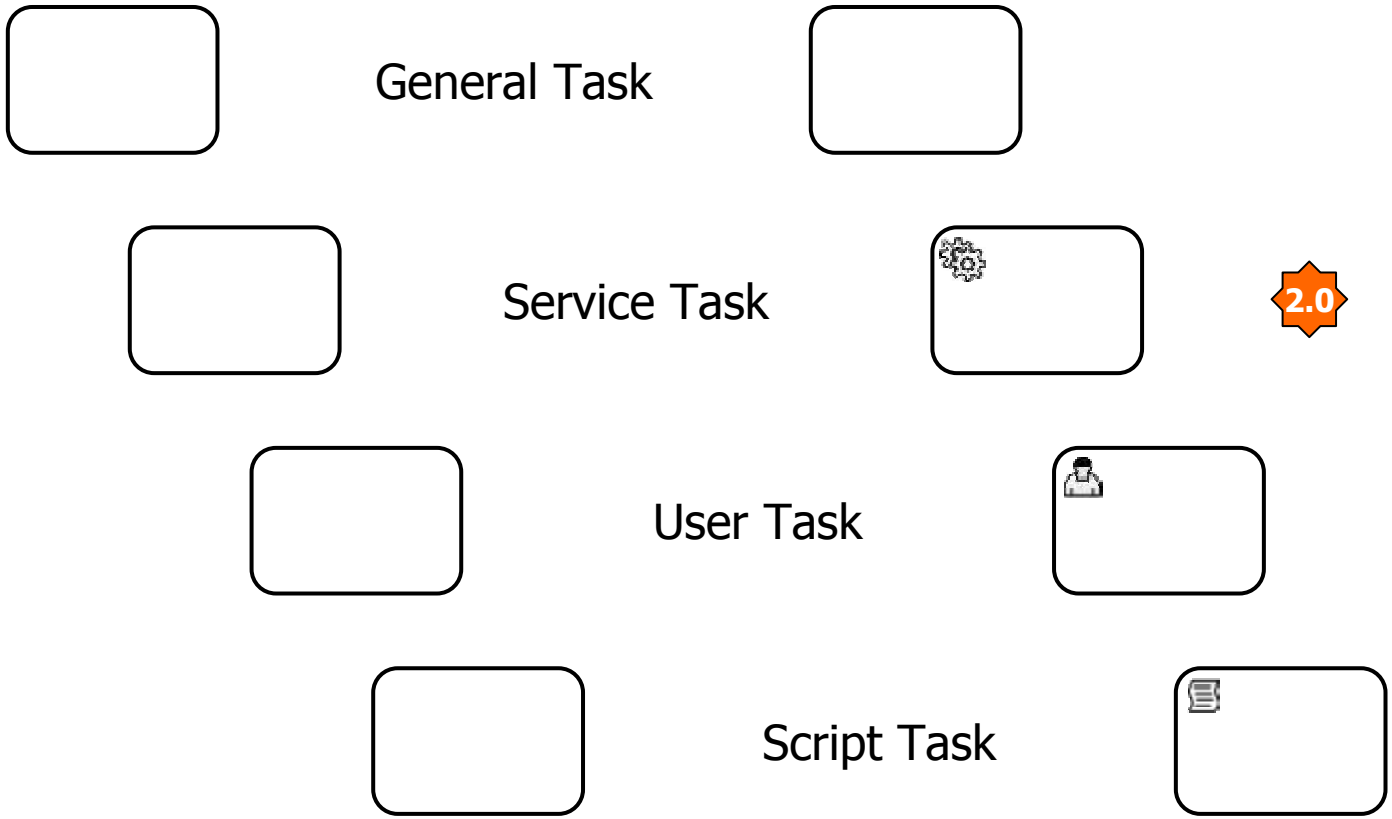


Activities

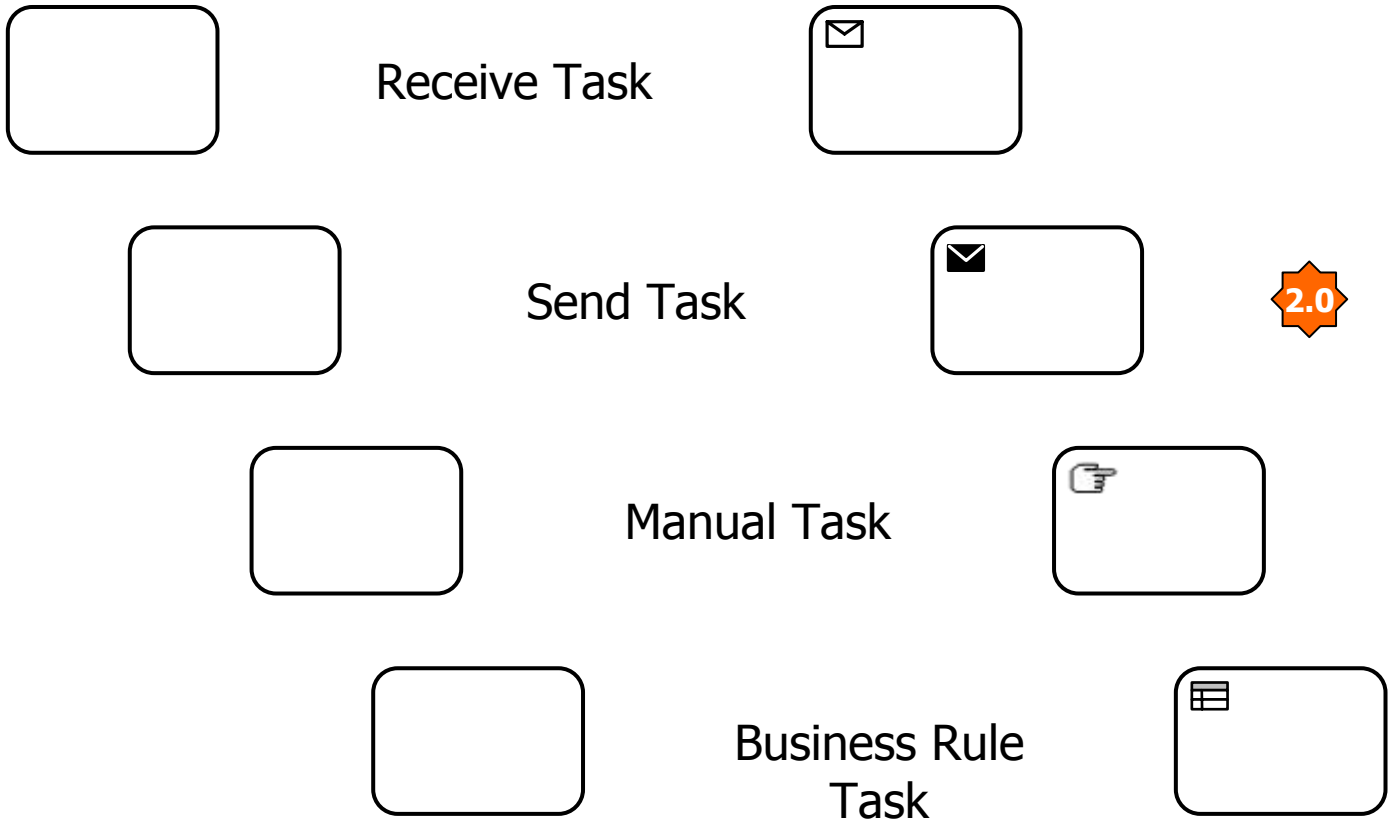
- Represents work performed in a process
- Atomic activities – Tasks
  - General, service, human, script, …
- Compound activities
  - Inline (sub-process) or referenced (2.0 call activity)
- Decorations for looping, multiple instantiation, ad-hoc execution



Tasks



Tasks (continued)



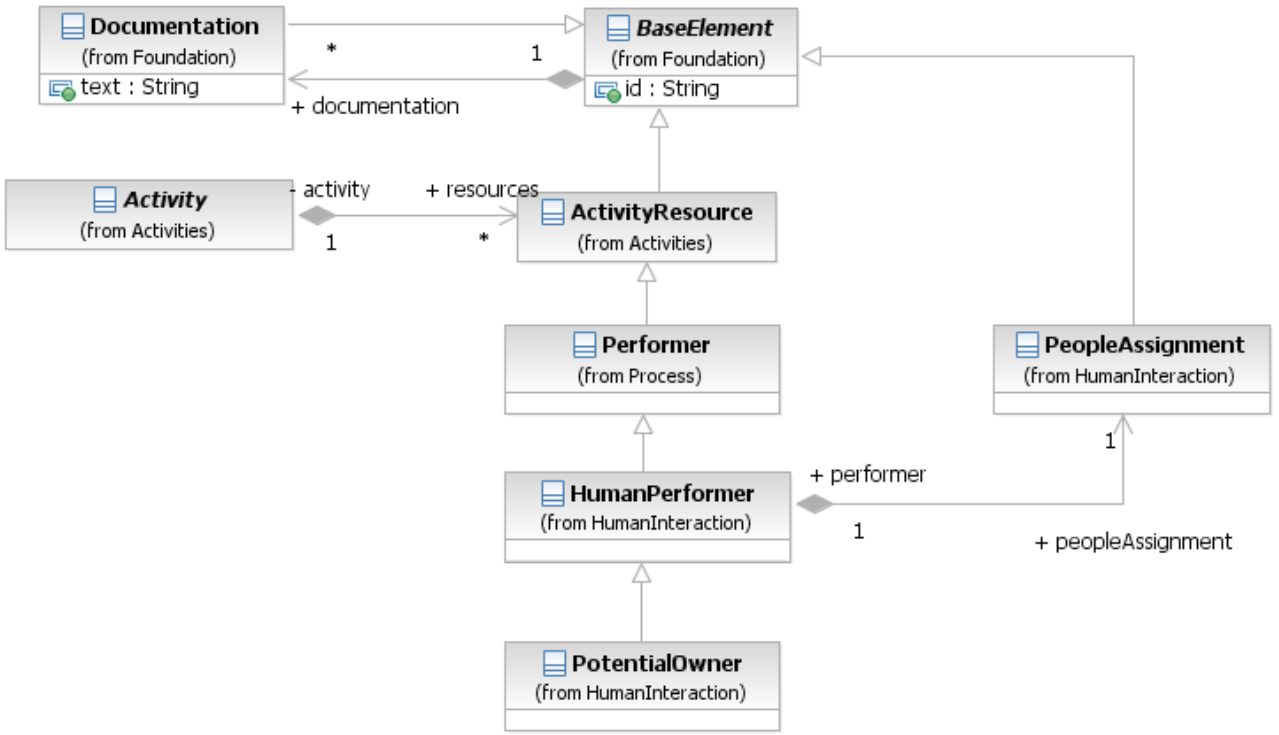
Service Model

- BPMN 2.0 adds a service model to capture service-related information
  - Interface
  - Operation
  - Endpoint
- Captures WSDL 1.1 style information
  - Allows referencing service information from activities and processes
- Alignment with SoaML in progress
  - Similar relationship as BPEL and SCA

2.0

People Assignment Model (1)

- Human Performer identifies human resources
- Potential Owner identifies responsible people
- People Assignment to link to actual people



2.0

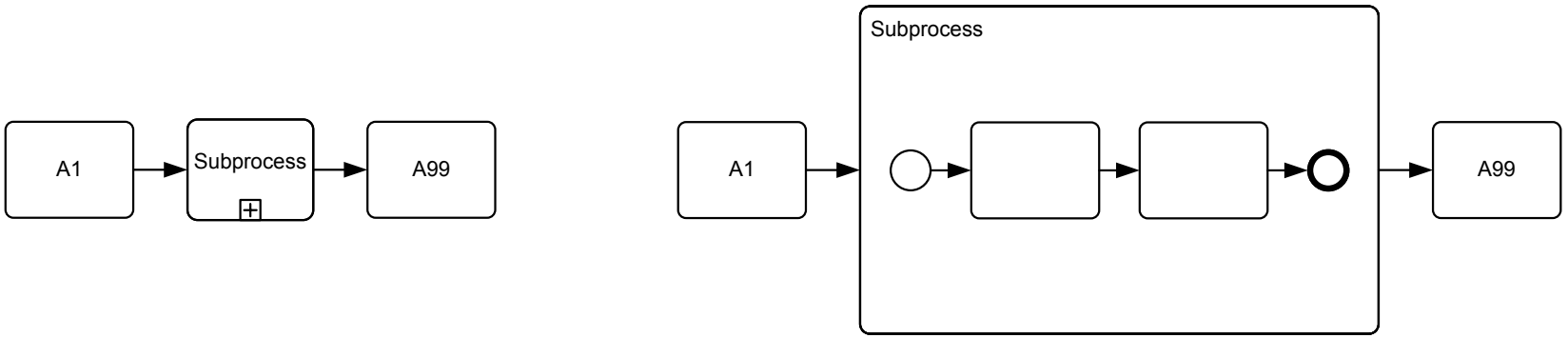
People Assignment Model (2)

Multiple forms of People Assignment

- Literal – list of user IDs
- Expression – eval at runtime, returns list of user IDs
  - Example: getActivityInstanceAttribute(“approve”, “owner”)
- Process Role – abstract group of people
  - Named, possibly parameterized
    - Example: salesManager(region)
  - Evaluated at runtime, binding arguments from process context
    - Example: salesManager(“CHE”)→“Konrad Schmidt”

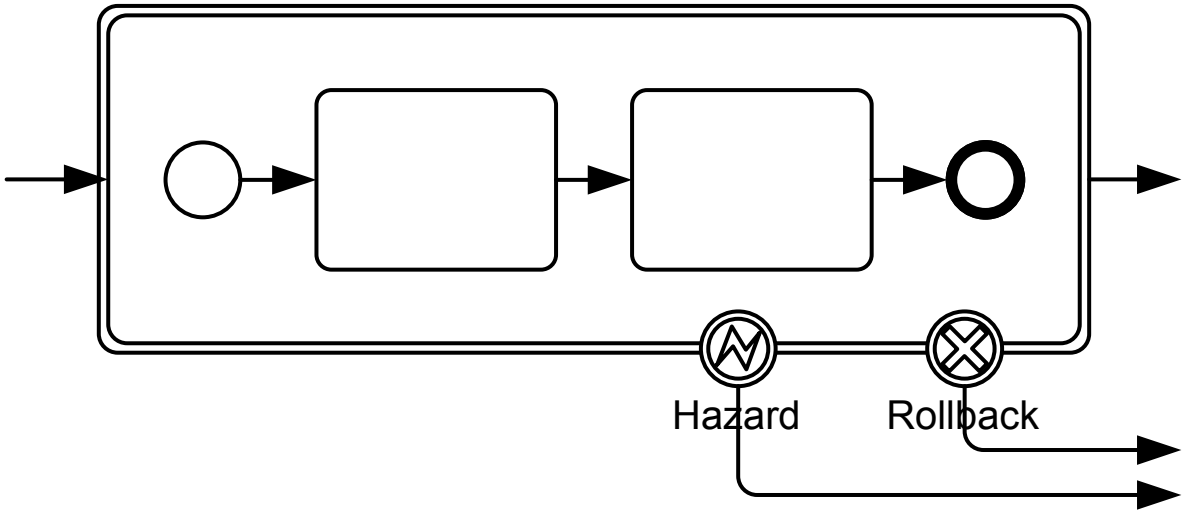
Subprocess

- Activity whose internal structure is a flow
  - Compound activity
  - Recursive aggregation to structure processes
- Can be collapsed/expanded
  - Affects the visual model only
- Provides scoping boundary
  - For data, exception handling, transactions, …



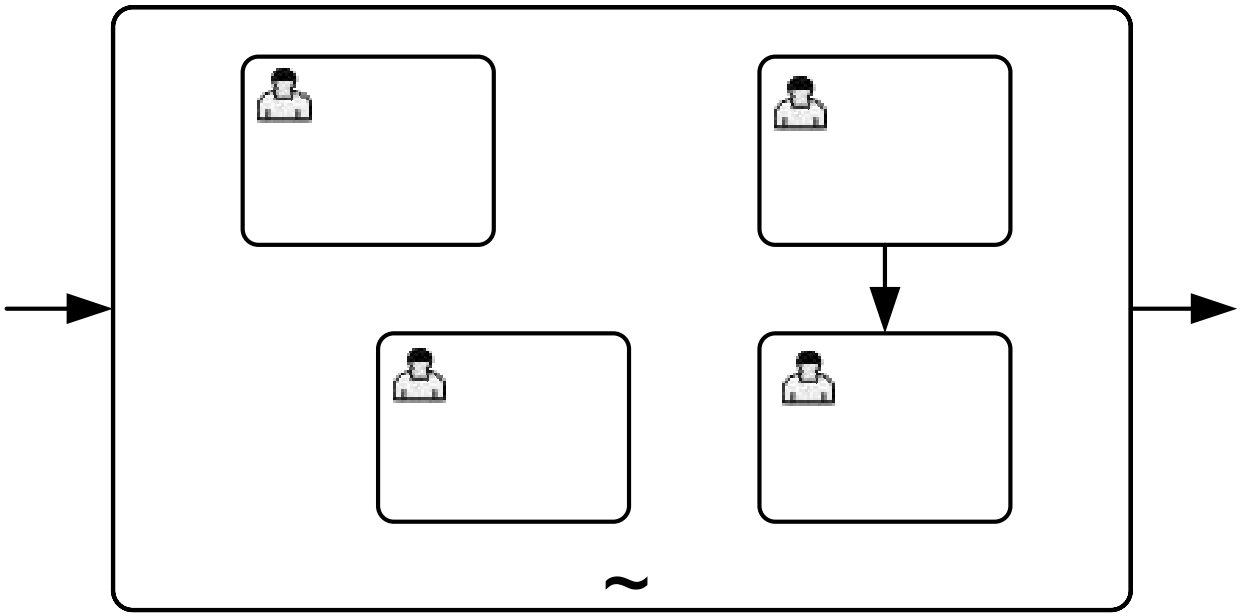
Transaction Subprocess

- A subprocess with transactional behavior of its activities, maintained through a coordination protocol



Ad-hoc Subprocess

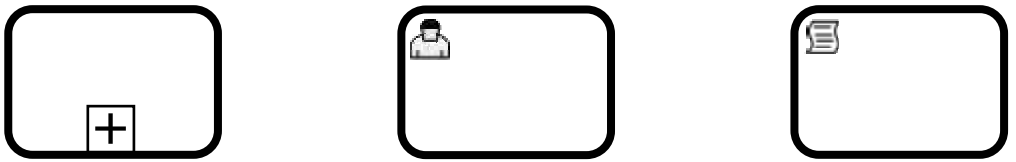
- Used for collaborative processes
- Contained activities can be performed in any order
  - As decided by assigned people



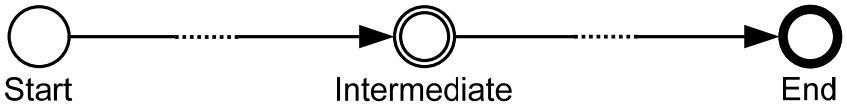
2.0

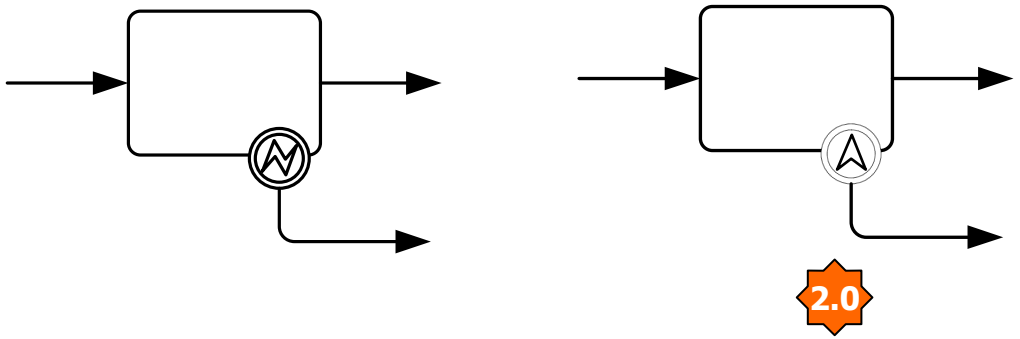
Call Activity

- Activity to call another process, or global task
  - Process/task-aware call with life-cycle coupling
  - Notation: Thick boundary line



Events

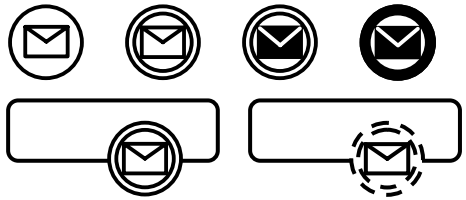




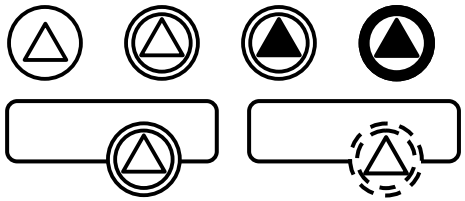
Event Types (1)

- General / unspecified
  - Event details not known or immaterial

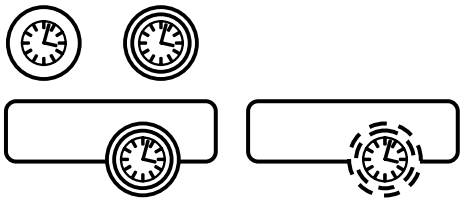
- Message
  - A message is received or sent



- Signal
  - A signal is received or sent

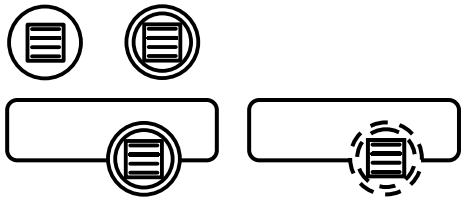


- Timer
  - A timer expires

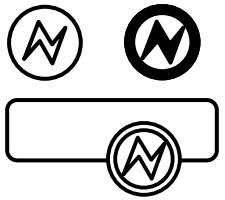


Event Types (2)

- Conditional
  - A condition becomes true

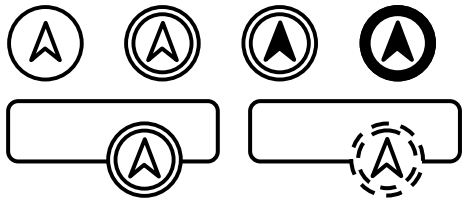


- Error
  - An error is thrown or caught

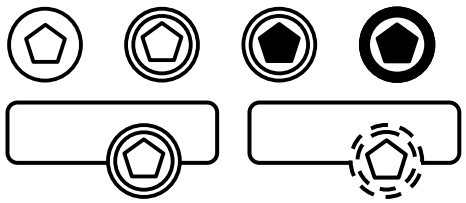


- Escalation
  - An escalation is thrown or caught

2.0

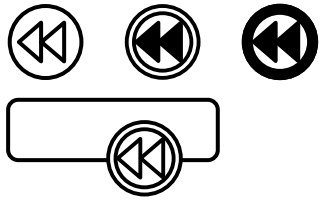


- Multiple
  - A combination of other events is thrown or caught



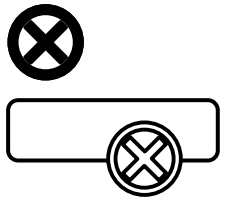
Event Types (3)

- Compensation
  - Compensation is triggered or handled



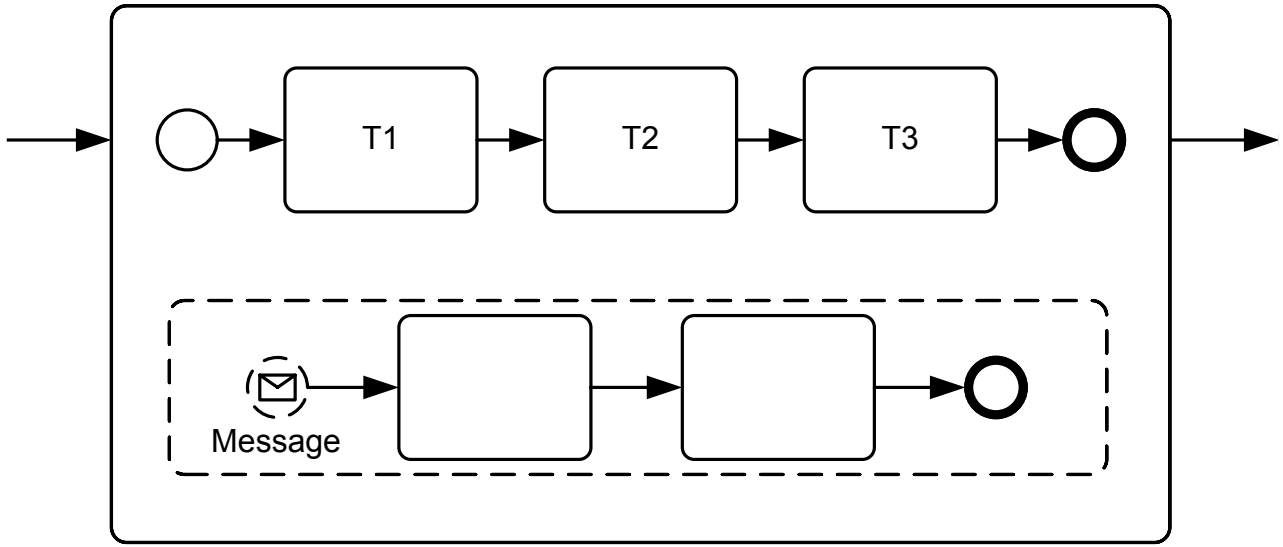
- Terminate
  - Abort the current process

- Cancel
  - Rollback of current transaction is triggered or handled



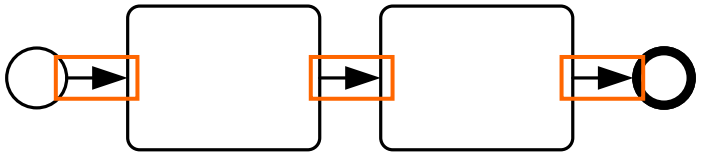
2.0

Event Subprocess

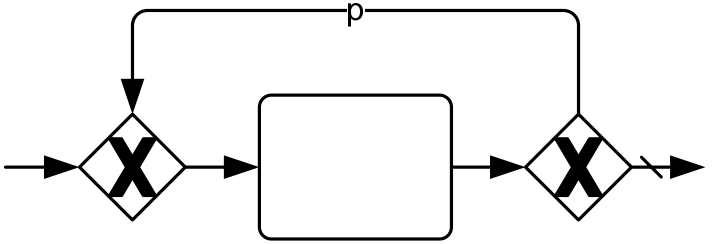


Establishing Order

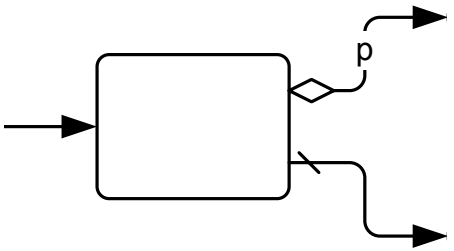
- Sequence flow
  - Perform steps in sequence



- Gateways
  - Split and merge the flow of control
    - Conditional execution
    - Parallel execution
    - Cyclic execution
    - …

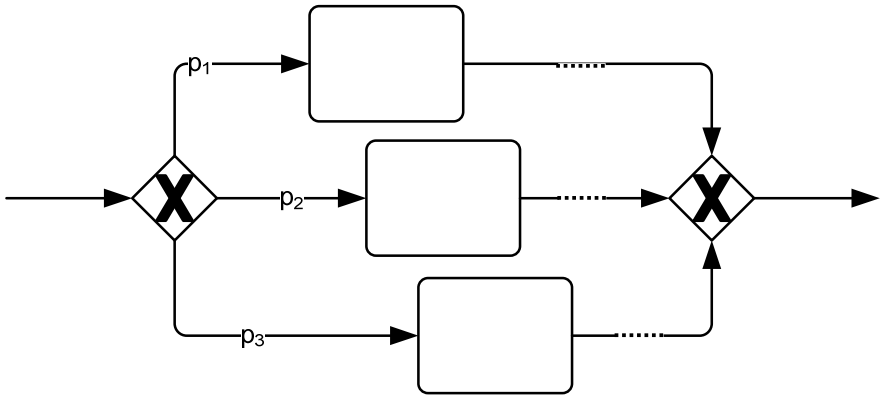


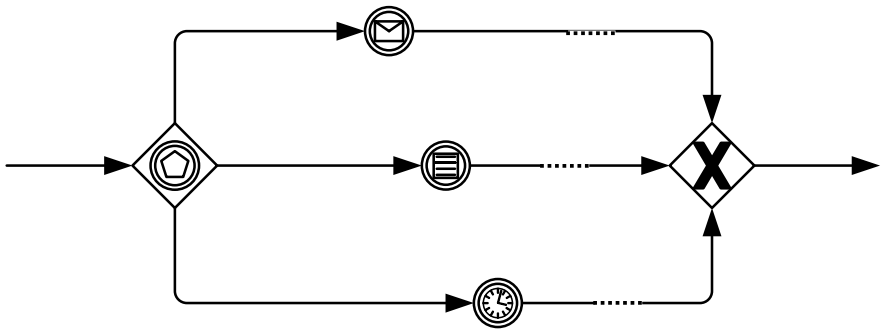
- Shortcuts
  - Implicit IOR in case of multiple outs
  - Implicit XOR in case of multiple ins



Gateway Types (1)

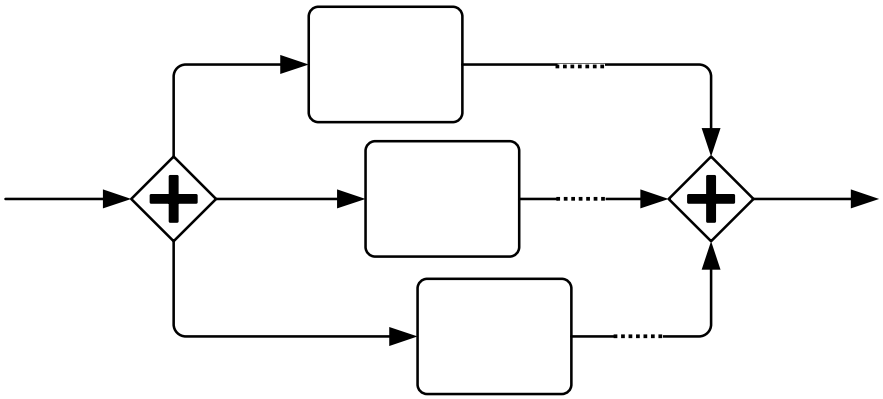
- Exclusive Gateway
  - Selects exactly one branch, the first whose condition is true
  - Merges control flow from several exclusive branches
- Exclusive Gateway – event-based
  - Selects exactly one branch, based on event trigger
  - (Merge as before)

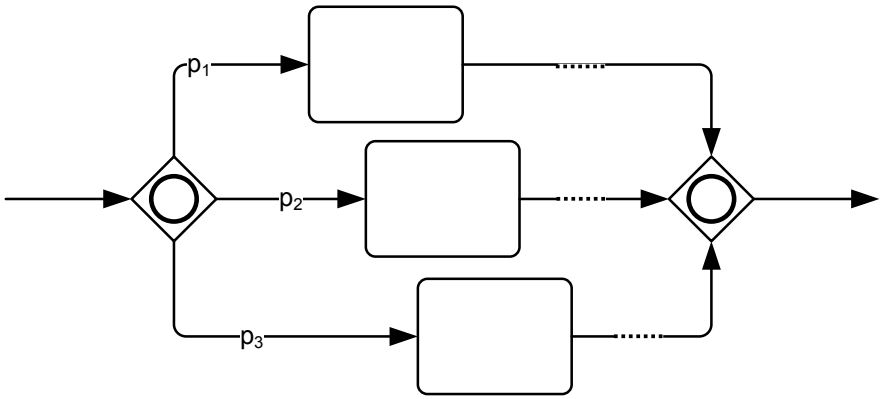




Gateway Types (2)

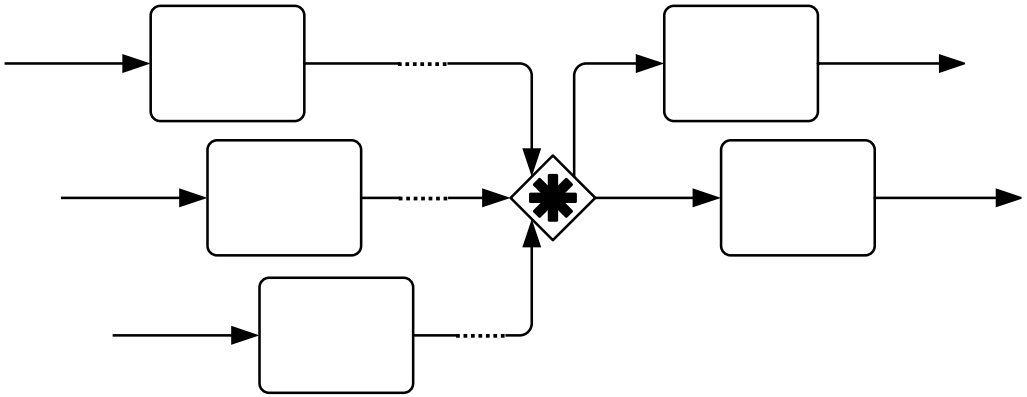
- Parallel Gateway
  - Forks control flow into parallel branches
  - Joins control flow from parallel branches
- Inclusive Gateway
  - Forks control flow into parallel branches, activating only those whose conditions are true
  - Joins control flow from several inclusive branches



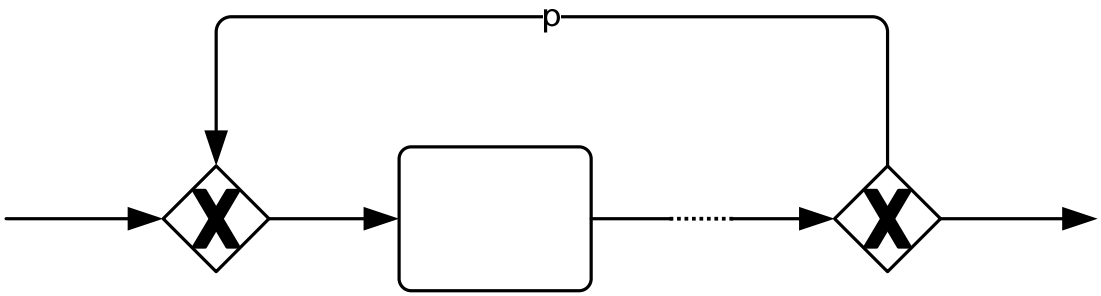


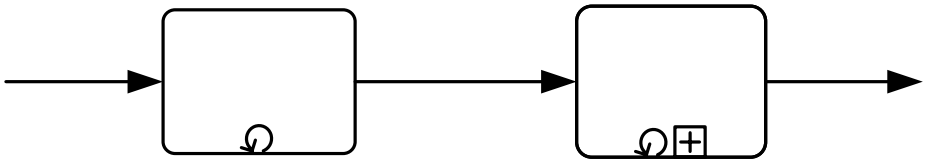
Gateway Types (3)

- Complex Gateway
  - Only exists as converging gateway
    - Mandatory standard output, optional “reset” output
  - Used to support complex synchronization behavior
    - Arbitrary condition on incoming connectors for standard output
      - E.g., n out of m
    - IOR for remaining connectors for reset output
      - E.g. remaining (m-n) out of m



Loops





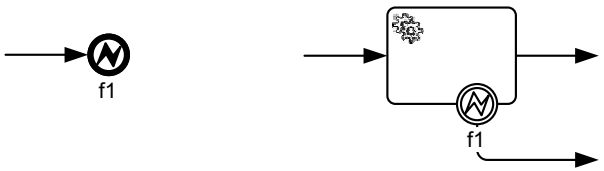
Multi-Instance

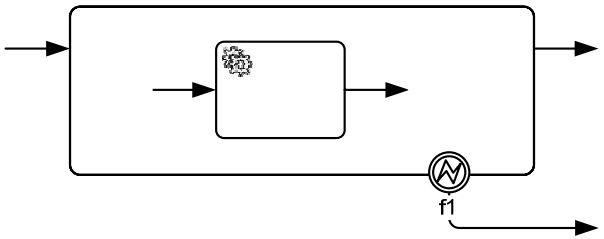
Multiple instantiation of an activity at runtime

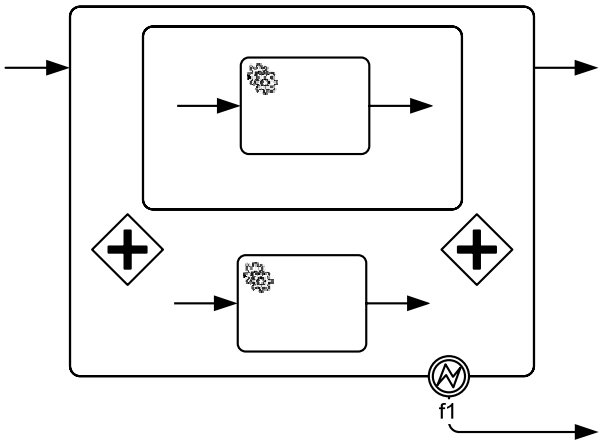
- Data-driven – via integer expression or cardinality of data array
- Can be performed sequentially or in parallel
- Different behavior alternatives regarding completion and aborting still running instances
- Multi-instance behavior can be specified for any activity

Error Handling

- Errors are raised internal to an activity, or explicitly
- Errors are caught on a boundary
  - … task
  - … sub-process
    - Ongoing parallel work is terminated
- Uncaught errors are propagated up the scope tree
  - Termination is propagated down



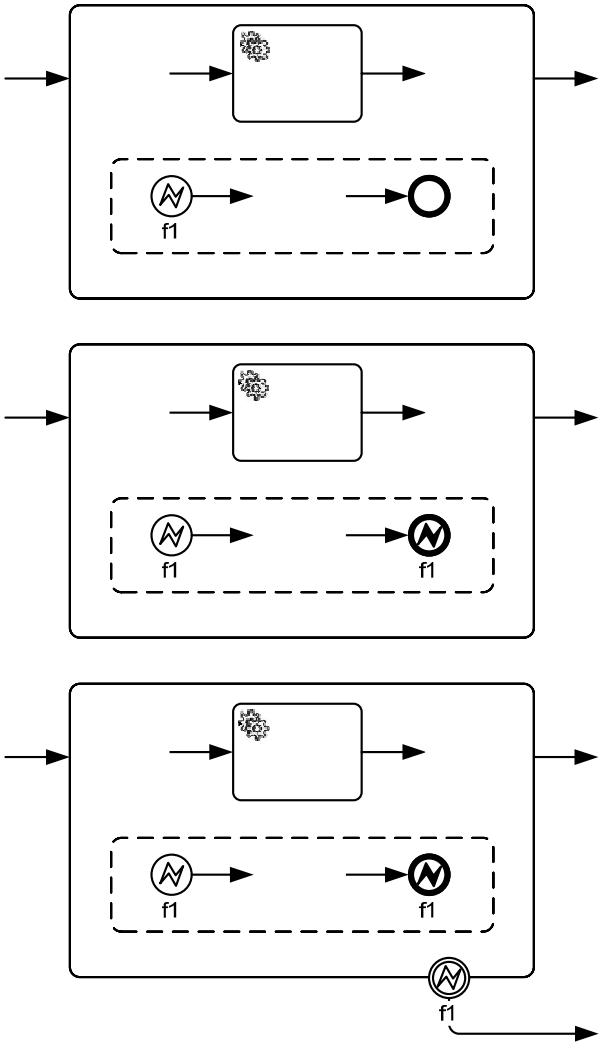




2.0

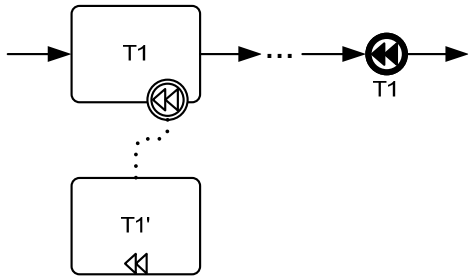
Error Handling (2)

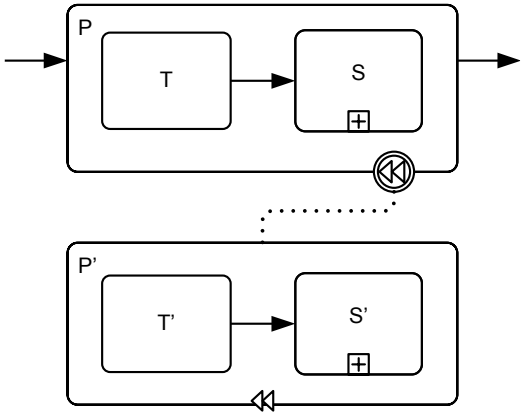
- Inline error handler
  - … can “absorb” error
  - … can rethrow error
- Upwards propagation extended
  - Inline handlers are considered first
  - Immediately enclosed boundary handlers second



Compensation (1)

- Compensation pairs
  - Activity can be associated with “undo” activity
- Compensation triggered via event
  - Will trigger execution of associated
- No access to subprocess context
  - Only black-box compensation

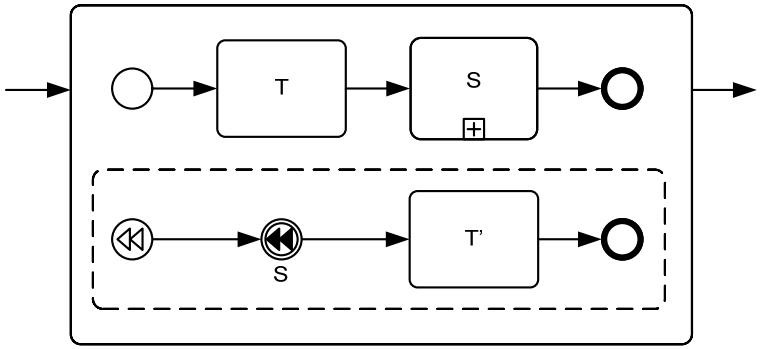


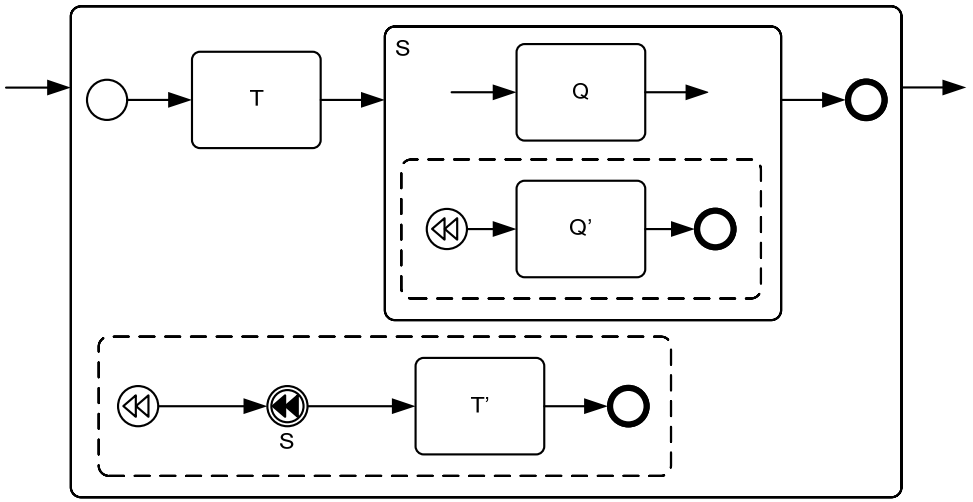


2.0

Compensation (2)

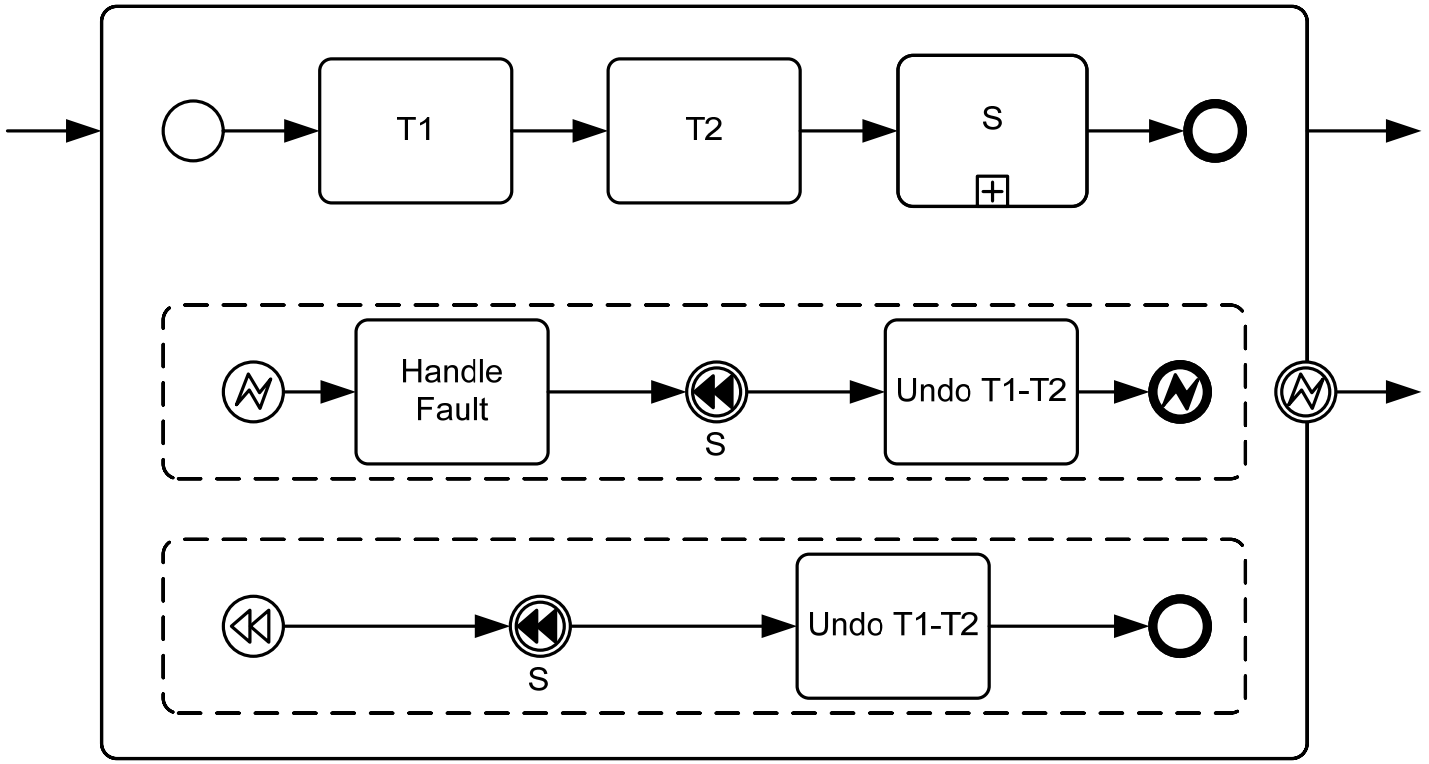
- Compensation “Event Sub-process” (Handler)
  - Access to subprocess context – white-box compensation
  - Allows for recursive compensation definition
- Compensation Triggering
  - From a compensation handler
  - From an error handler





2.0

Error Handling and Compensation



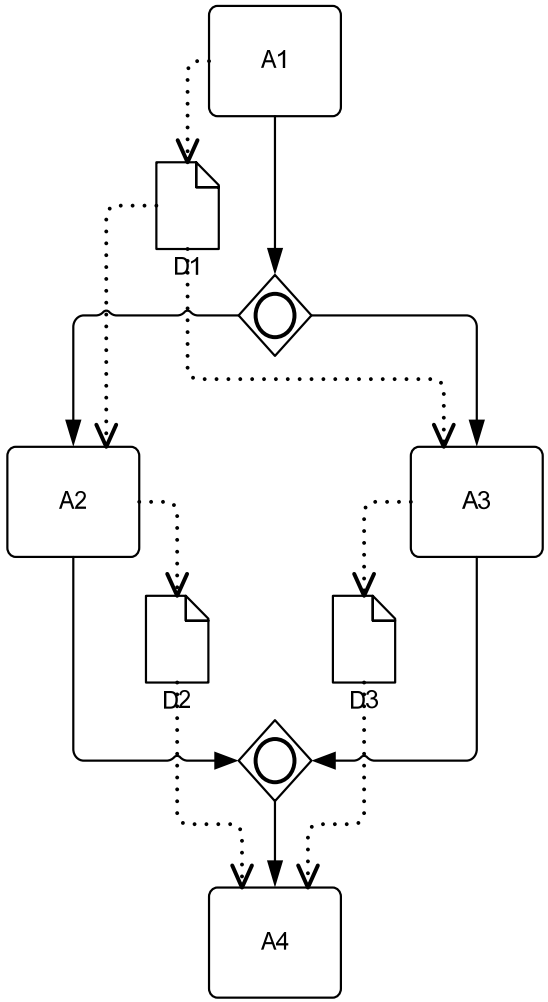
2.0

Error Handling and Compensation

- Successful completion of subprocess
  - Preparation for possible compensation, snapshot of data is taken
- Unsuccessful completion of subprocess
  - *Presumed abort:* no side effects may persist
  - Error handler possibly needs to take care
- Compensation
  - Successful completion is undone
  - Snapshot data is made available to handler

Data Flow

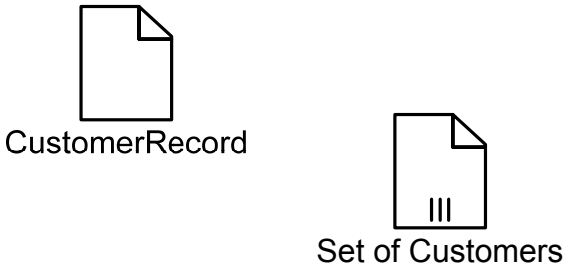
- Data Object
  - Used to store data
  - Pre-defined structure definition
  - Scoped, with standard visibility rules and life-time
- Data association 2.0
  - Describe data flow between data objects and flow nodes (activities or events)



“Item-aware Elements” (1)

Data Object

- Pre-defined (modeled) structure definition
- Visualized in process diagram
- Contained in enclosing (sub-) process
- Single instance or collection



Property

- Pre-defined (modeled) structure definition
- Not visualized
- Contained in (sub-) process or activity

2.0

"Item-aware Elements" (2)

Data Input

- Input of an activity
- Not visualized

Input Set

- Collection of inputs forming a valid set

Example Input Set

- { customerID
  | (customerName, address, SSN) }

Data Output

- Output of an activity
- Not visualized

Output set

- Collection of outputs forming a valid set

Example Output Set

- { customerRating
  | error
  }

2.0

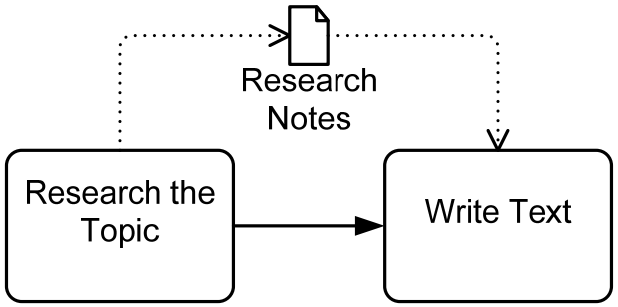
Data Association

Input data association

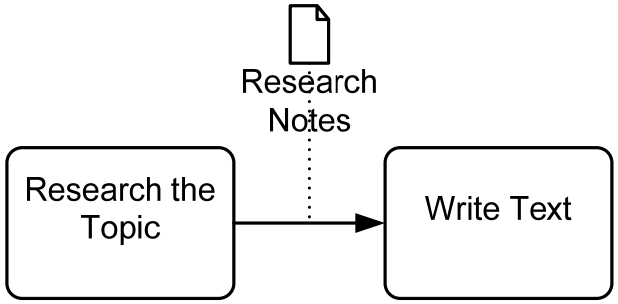
- Provide data for inputs of an activity
- Possibly multiple sources: data objects, properties, expressions
- Transformation specified as expression

Output data association

- Update data objects or properties after completion of an activity
- Sources: activity's data outputs
- Transformation specified as expression



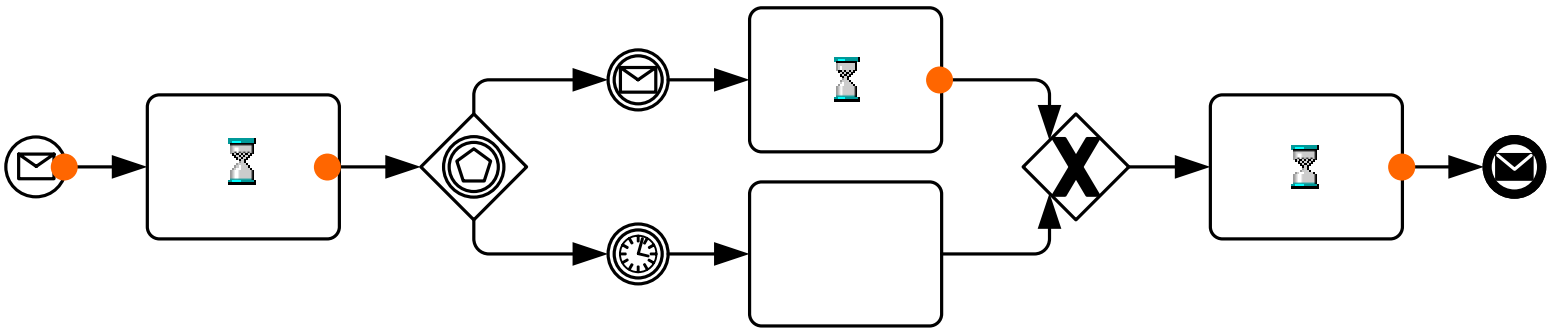
Data input and data output association



Shortcut notation

Execution Semantics

- Execution Semantics is described by means of token flow
  - Tokens flow along sequence flow
  - Tokens are produced and consumed by activities, events and gateways

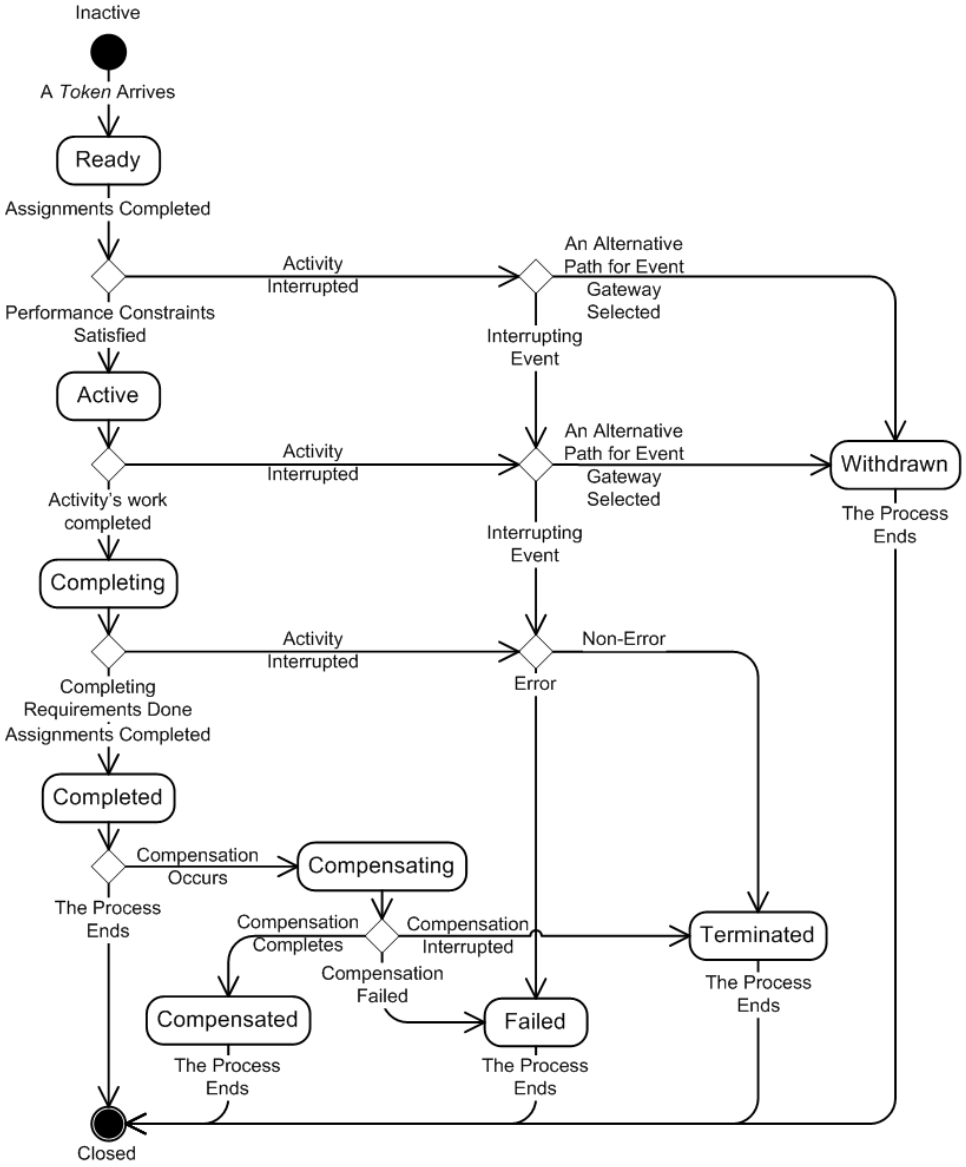


Process Life-cycle

- Instantiation …
  - … when start event is triggered
    - Exception: If start event is part of an existing conversation, that conversation is joined
- Termination …
  - … when all of the following holds
    - All start events have been triggered (one for each group in case of starting event-based gateways)
    - No remaining token
    - No active activity

2.0

Activity Life-cycle



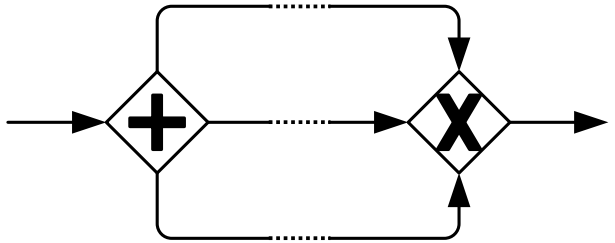
Soundness

Sound process:

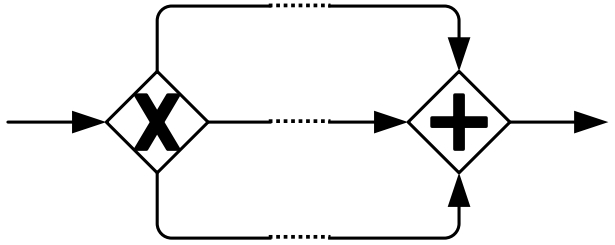
- No lack of synchronization
- Deadlock free

BPMN does not require processes to be sound

- Lack of synchronization



- Deadlock



Mapping from BPMN to BPEL

- Pre-req: Process is sound
- Basic mapping
  - Syntactical mapping: Basic BPMN blocks are mapped to BPEL constructs
    - Semantics are preserved
  - Applies to subset of all sound BPMN processes
- Extended mapping
  - Builds on basic mapping
  - Any mapping of BPMN to BPEL that preserves the BPMN semantics
  - Spec describes a number of specific patterns
  - Extensible by vendors, allows usage of vendor extensions

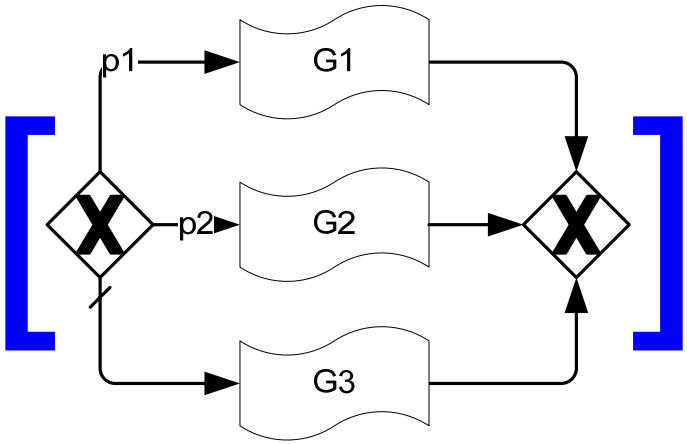
2.0

BPEL Mapping

- Recursive description, driven by BPMN syntax structure
- Notation:
  - [BPMN fragment] = BPEL fragment

2.0

BPEL Mapping Example – If-Elseif-Else

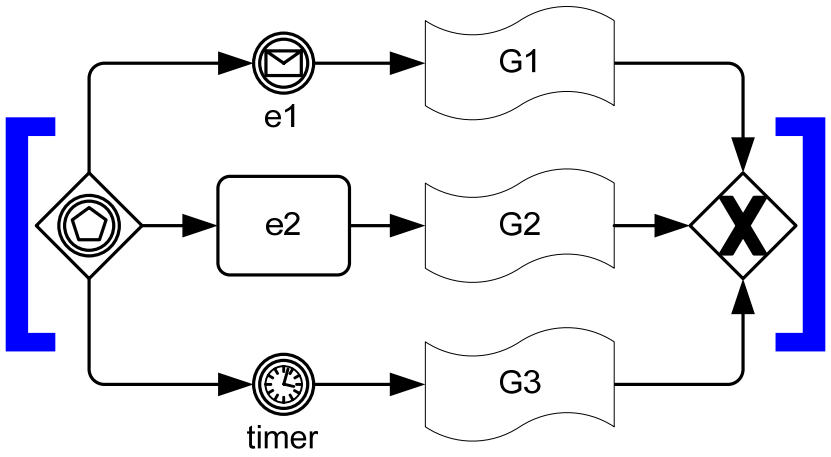


=

```
<if><condition>[p1]</condition>
  [G1]
<elseif><condition>[p2]</condition>
  [G2]
</elseif>
<else>
  [G3]
</else>
</if>
```

2.0

BPEL Mapping Example – Pick



=

```
<pick createInstance="[instantiate? 'yes':'no']">
  <onMessage partnerLink="[e1-operation-interface]"
       operation="[e1-operation]">
    [G1]
  </onMessage>
  <onMessage partnerLink="[e2-operation-interface]"
       operation="[e2-operation]">
    [G2]
  </onMessage>
  <onAlarm>
    [timer-spec]
    [G3]
  </onAlarm>
</pick>
```

Web Service Orchestration and WS-BPEL

- Part I – Web Service Orchestration
  - Motivation and Overview
  - Business Process Modeling Styles
- Part II – The WS-BPEL 2.0 Standard
  - WS-BPEL 2.0 Concepts and Language Elements
- Part III – WS-BPEL 2.0 Extensions
  - WS-BPEL Extension for People (BPEL4People)
  - WS-BPEL Extension for Subprocesses (BPEL-SPE)
  - WS-BPEL Extension for Java (BPELJ)
- Part IV – Additional Related Standards
  - Service Component Architecture (SCA)
  - Business Process Modeling Notation (BPMN)

Web Service Orchestration and WS-BPEL

Thank You

QUESTIONS ?

dieterkoenig@de.ibm.com