SOA CLOUD 2012 SERVICE TECHNOLOGY SYMPOSIUM

World's Largest International Conference Dedicated to the Exploration of SOA, Cloud Computing and Modern Service Technologies

September 24-25, 2012 | London, UK

ServiceTechSymposium.com | ServiceTech.co

Designing Composite Services Using BPMN 2.0 as a Visual Programming Language

Track: New SOA and Service-Orientation Practices & Models

Presented September 24, 2012

Lloyd Dugan – Founder LAB Derivations (BPMN4SCA), and BMA'S Senior BPMN SME for the U.S. Department of Defense (DoD), Deputy Chief Management Office (DCMO)

Nathaniel Palmer – Chief BPM Strategist and BPM/SOA Practice Director at SRA, and Executive Director of the Workflow Management Coalition (WfMC.org)

Outline

- BPMN & SOA: Creating an Opportunity To Use BPMN for Services Modeling

- BPMN & Services: Visual Programming With BPMN for Service Orchestration

- BPMN & SCA: Translating BPMN Into Service Composition Representations

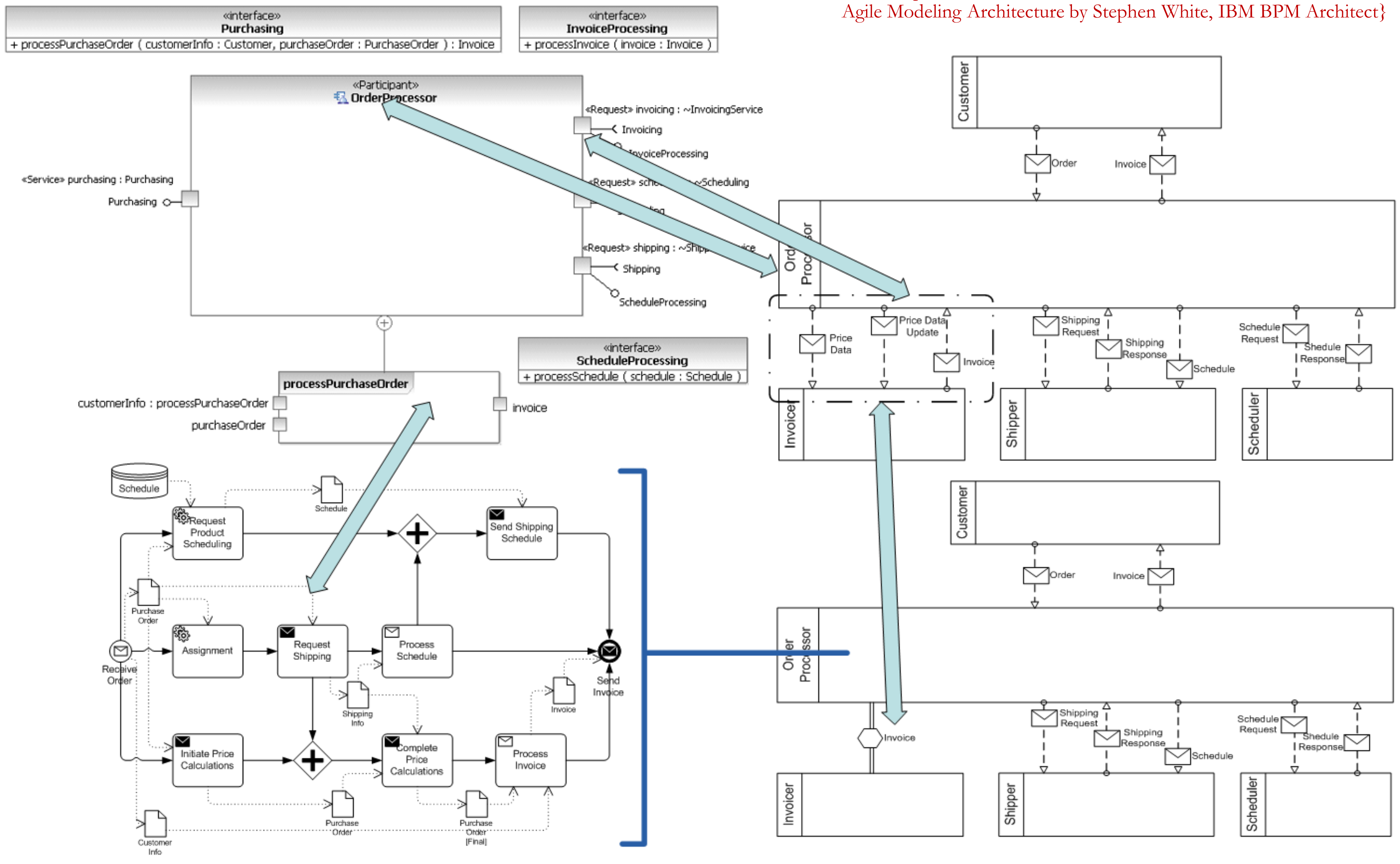
BPMN & SOA: Creating an Opportunity To Use BPMN for Services Modeling

Background

- BPMN 2.0 specification formally released in January 2011 by the OMG
  - Additional notational elements and new behaviors
  - Meta Object Facility (MOF) for BPMN in UML
  - Serialization into BPMN XML per defined schema (XSD)
  - Executable BPMN XML includes schema for item aware elements
  - Conformance classes introduced: Descriptive, Analytic, and Common Executable
  - Services and interfaces model explicitly incorporated
- Future of BPMN {Source: "Future of BPMN" presented on 9/6/2012 at BPM 2012 Conference by Stephen White, IBM BPM Architect}
  - Case Management (for more unstructured processes)
  - Service Level Modeling (for more structured processes)
- Service Modeling Issues To Resolve
  - Which representation alternatives to use: SoaML (OMG), SCA (OASIS), SOA ontology (The Open Group), BPMN ontology (currently being put forward to the OMG by the DCMO)
  - BPMN MOF, schema, and rules are…challenging to interpret and realize, leading to inconsistent interpretations (no reference model or implementation exists)
  - Concept of executable BPMN complicates discussion, particularly with BPMS pure play vendors (with engine only) vs. BPMS platform vendors (also with middleware) competing for market share
  - Attribution of BPMN XML elements is problematic due to tooling limitations and lack of clarity in certain cases as to spec's intentions and meanings

Early BPMN Mapping To SoaML

{Source: "Supporting MDA through Integration of BPMN and SoaML" presented on 9/30/2010 at the 5th International Forum on Agile Modeling Architecture by Stephen White, IBM BPM Architect}



«interface» Purchasing
+ processPurchaseOrder (customerInfo : Customer, purchaseOrder : PurchaseOrder) : Invoice
«interface» InvoiceProcessing
+ processInvoice (invoice : Invoice)
«Participant» OrderProcessor
«Request» invoicing : ~InvoicingService
Invoicing
InvoiceProcessing
«Request» scheduling : ~Scheduling
«Service» purchasing : Purchasing
Purchasing
«Request» shipping : ~ShippingService
Shipping
ScheduleProcessing
«interface» ScheduleProcessing
+ processSchedule (schedule : Schedule)
processPurchaseOrder
customerInfo : processPurchaseOrder
purchaseOrder
invoice
Schedule
Request Product Scheduling
Send Shipping Schedule
Purchase Order
Assignment
Request Shipping
Process Schedule
Receive Order
Send Invoice
Shipping Info
Invoice
Initiate Price Calculations
Complete Price Calculations
Process Invoice
Customer Info
Purchase Order [Final]
Customer
Order
Invoice
Order Processor
Price Data
Price Data Update
Shipping Request
Shipping Response
Schedule
Schedule Request
Shedule Response
Invoicer
Shipper
Scheduler

Variations of Executable BPMN

- Modeler's Definition:
  - BPMN model can be handed off for "execution" as an unambiguous process for performers to follow or as an application design to be developed and implemented
- BPMS's Definition:
  - BPMN model is the application design, and the BPMS implements its functionality per the configuration (mostly) and some scripting (maybe) of modeled elements
- BPMN Spec's Definition:
  - Fully attributed BPMN model in XML form that is also the internal representation of the application, and the BPMS executes it per the spec's rules
- Proposed Definition (pitched as "execution-lite"):
  - Specifically-attributed BPMN model in XML form that contains the service stubs and system structure of the designed process that translate to service representations

BPMN as Visual Programming

Major Improvement	Examples from the Spec	Impact on Executability
Richer notational set and associated modeled behaviors	New or improved event types and behaviors, and addition of reusable Call Activity	Comprehensively covers programmable functions and system behaviors
Better support for abstraction of executable functionality	Requirement of WSDL for service calls, and addition of Business Rule Task	Enhances the role of orchestration and makes it more stateless
Better representation of components in process execution	Improved mapping to BPEL, and new attributes of messages and data objects	Enables alignment with service representations (e.g., SCA component assembly)
Structured capture and standardization of implementation detail	Use of standards (WSDL, XPath, XSD), and serialized XML for model attributes	Design-time model can be exported or interpreted as a run-time construct

Result: BPMN Is Now a Visual Programming Language!

BPMN & Services: Visual Programming With BPMN for Service Orchestration

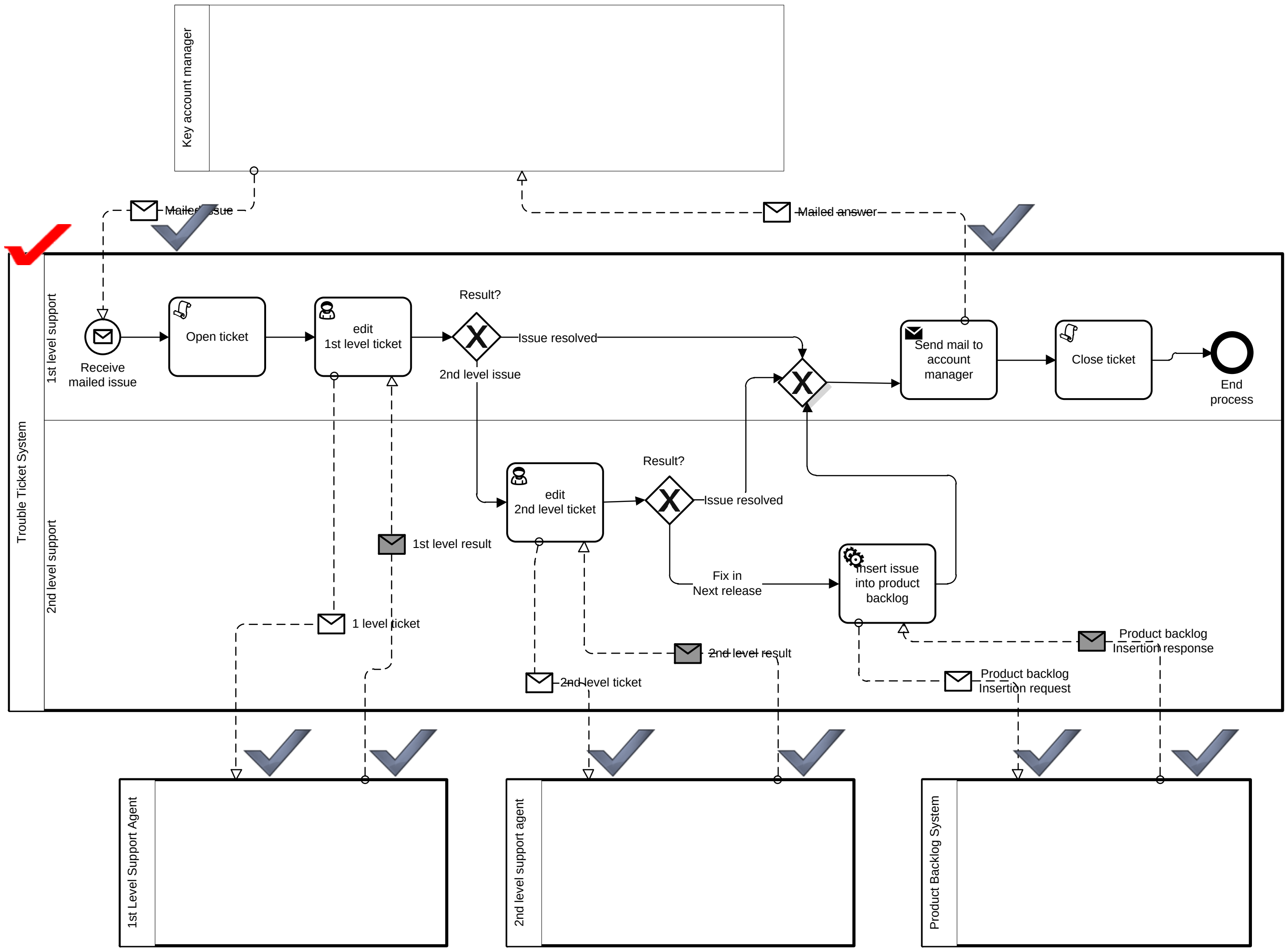
Focus on "Lean BPMN" that Relies on the Analytic Conformance Class and Selected Attributes from the Common Execution Conformance Class

System as a Pool in BPMN

System Pool as the Focus

Expanded System Pool in the collaboration diagram provides the functional detail of what the system is doing

Collapsed Process Pools in the collaboration diagram provide the interfaces between the activities of the hidden processes (or other systems) and the supporting system as seen by the System Pool itself



{Source: Adapted from "BPMN v2.0 by Example" (PDF) at http://www.omg.org/spec/BPMN/20100601}

Modeling Systems With BPMN

System Concept	Equivalent BPMN Representation
Execution context and framework	• Pool defines the controlling framework for the system • Lanes segment system resource (performer) responsibilities • System interfaces are realized as boundary-crossing moments with other participants
Control flow and routing logic	• Sequence flow and conditional expression logic define order of execution • Gateways and branches define inbound and outbound execution paths
Procedural and modular logic	• Service task, send task, receive task, and business rules task define specific and germane operations that are executed via invocation • User task defines an operation that is executed via one or more user interfaces • Script task defines an operation that manipulates data that is executed internally • Subprocess and Call Activity define a set of operations that are executed separately • Start events and end events define the start and end conditions for execution • Intermediate events define operations based on event type that are executed • Boundary events add operations to the attaching activities that are executed
Data I/O declaration	• Data objects as input or output with corresponding data sets as input or output, respectively, by way of corresponding data associations • Messages and signals define data input or output as data payloads
Integration interfaces (APIs)	• Service task, send task, receive task, business rules task, and user task define service interfaces to abstracted and/or external components • Message event and signal event define service interfaces with other systems

BPMN Modeling Principles

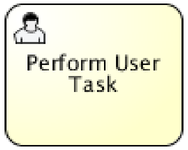
- Separate By Performers
- Separate By Functionality
- Separate By State Transitions

Ways To **Decompose** Functionality Within a Process Level

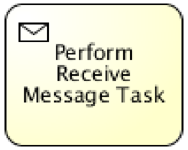
- Use a Subprocess
- Use Abstraction (Service Task, Business Rule Task, User Task, and Send Task + Receive Task)
- Use a Reusable Element (Call Activity)

Ways To **Abstract** Functionality Within a Process Level

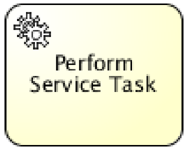
BPMN Tasks as Abstractions



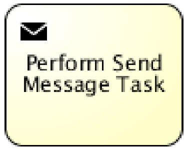
Represents a call to a user interface-based application that represents one or more screens for one or more users, with the hard state change for the transaction being processed realized at the end of the navigation



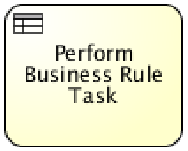
Similar to a catching message event, it represents a call into the process in which the Receive Task is contained, which is received from another pool (participant) as part of a collaboration



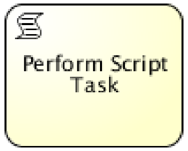
Invokes a service based on an input set to the service provider that synchronously exchanges information with the requesting process



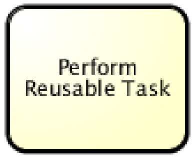
Similar to a throwing message event, it represents a call to another process from out of the process in which the Send Task is contained, which is sent to the other pool (participant) as part of a collaboration



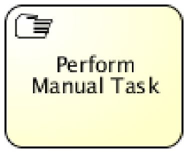
Invokes a service in a manner similar to a Service Task, except that the provider is always a business rules engine that may be native to the platform



Represents the use of an executable script in the BPMN process, based on a declared scripting language, that is good for performing complex data mapping and transformations in-line instead of via abstraction, and thus can manage the in-line state of the data object



Calls a reusable task of one of the other task types, and can be started with various different types of start events and data scopes



Represents a non-automated task that describes an activity that is manually performed and not abstracted though it can be included in an executable design

BPMN Design Approaches

- Granularity Guidance:
  - Each Task/Event executes a single operation
  - Balance operations per optimal realization of coupling and cohesion across tasks/events in the model
- Transactional Boundary for Abstraction Tasks
  - A Task should typically be more ACID-like with respect to usage within the BPMN process sequence of the system design

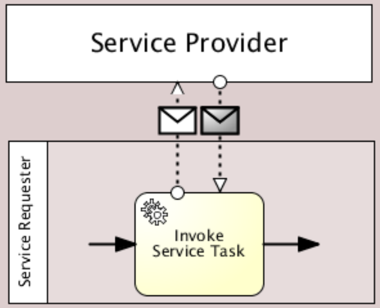
    A: Atomic – task is a single operation task
    C: Consistent – task behavior is consistent
    I: Isolated – task results are isolated
    D: Durable – task results are durable
  - A Task that is more BASE-like should typically be collected with related Tasks into an abstraction task type or into a subprocess

    B + A: Basically Available as an operation
    S: Soft state for the data object
    E: Eventual consistency of the intended result
- Other Design Characteristics To Incorporate
  - *Idempotent*: Process reacts only once for the submitted trigger, no matter how many times the same trigger is sent
  - *State Machine*: Process manages the state of the transaction through a series of stateless moments in the tasks
  - *Data Access Management*: Cache access service or a database access service is invoked via a Service Task
- API Styles {Sources: See http://www.servicedesignpatterns.com for a summary treatment, and http://www.soapatterns.org for a detailed listing of patterns}
  - Message-based APIs for SOAP-style web services for appropriately-named operations of some complexity
  - Resource-based APIs for RESTful services for simply-named operations of CRUD-like actions
  - Custom RPC APIs for proprietary services for specifically-named operations for non-standard interactions

Message Exchange Patterns (1)

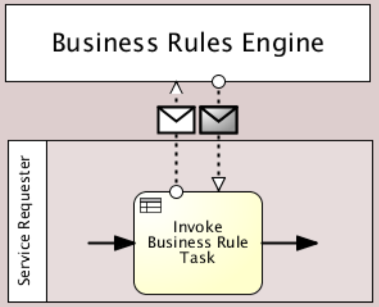
These are Blocking Until Response is Received…So these Should be for Short-running Transactions

Synchronous Request/Response – Service Task



Operation			
SCA Invocation Type = call		Name = Appropriate Name	
Implementation Component			
Type = Web Service, URI, Other		Name = Corresponds to Type	
Messages			
Request to Service		**Response from Service**	
Message Name/ID = Appropriate Name	Type = Initiating ○····✉·▷	Message Name/ID = Appropriate Name	Type = Non-initiating ○····✉·▷
Payload = Inbound Subset per Defined Schema for Service		Payload = Outbound Subset per Defined Schema for Service	
Correlation Key = Same as for **Response**		Correlation Key = Same as for **Request**	

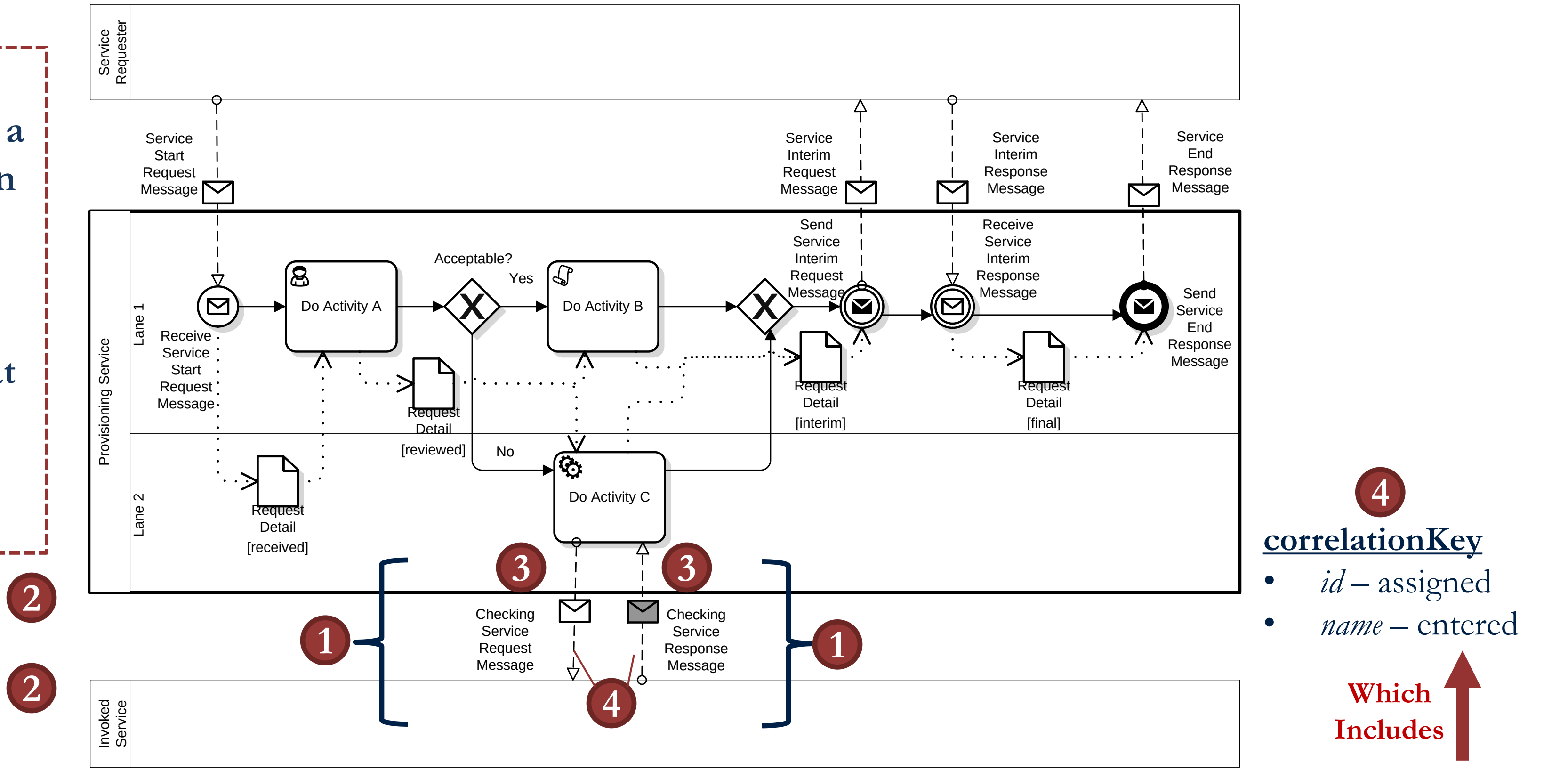
Synchronous Request/Response – Business Rule Task



Operation			
SCA Invocation Type = call		Name = None – Is Assumed	
Implementation Component			
Type = Web Service, URI, Other		Name = Corresponds to Type	
Messages			
Request to Service		**Response from Service**	
Message Name/ID = Appropriate Name	Type = Initiating ○····✉·▷	Message Name/ID = Appropriate Name	Type = Non-initiating ○····✉·▷
Payload = Inbound Subset per Defined Schema for Service		Payload = Outbound Subset per Defined Schema for Service	
Correlation Key = Same as for **Response**		Correlation Key = Same as for **Request**	

BPMN XML Attributes (1)

Correlating Messages in a Collaboration creates the indirection structure in the XML that defines a specific conversation


Service Requester
Service Start Request Message
Service Interim Request Message
Service Interim Response Message
Service End Response Message
Provisioning Service
Lane 1
Lane 2
Receive Service Start Request Message
Do Activity A
Acceptable?
Yes
No
Do Activity B
Do Activity C
Send Service Interim Request Message
Receive Service Interim Response Message
Send Service End Response Message
Request Detail [received]
Request Detail [reviewed]
Request Detail [interim]
Request Detail [final]
Checking Service Request Message
Checking Service Response Message
Invoked Service
1
2
3
4
correlationKey
id – assigned
name – entered
Which Includes

1
collaboration
- *id* – assigned
- *name* – entered

Includes One or More

2
participant
- *id* – assigned
- *name* – entered (is pool name)
- *processRef* – is **process** *id* (if a message flow connects to an element within the process)

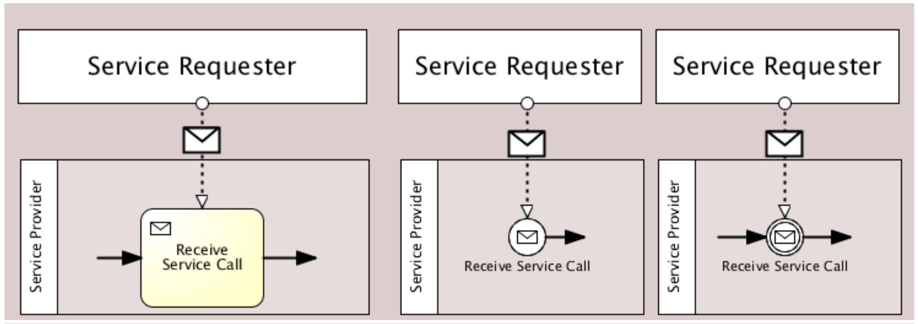
3
messageFlow
- *id* – assigned
- *name* – entered
- **messageRef** – is **message** *id*

4
conversationNode
- *id* – assigned
- *name* – entered
- **messageFlowRef** – is **messageFlow** *id* (one for each message flow)

Message Exchange Patterns (2)

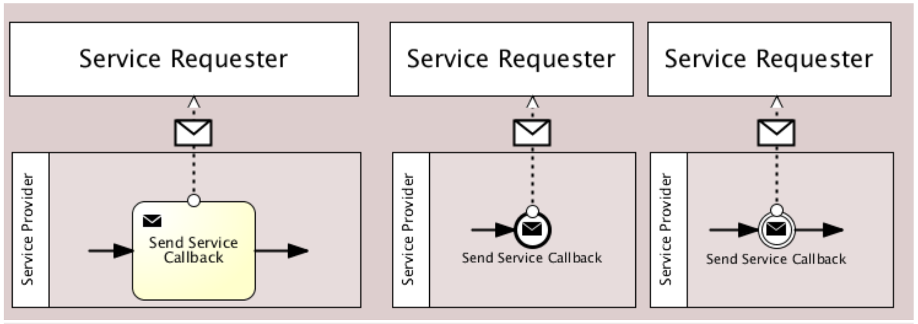
These are not Blocking Until Response is Received…So these Should be for Long-running Transactions

Asynchronous Request/Response – Request



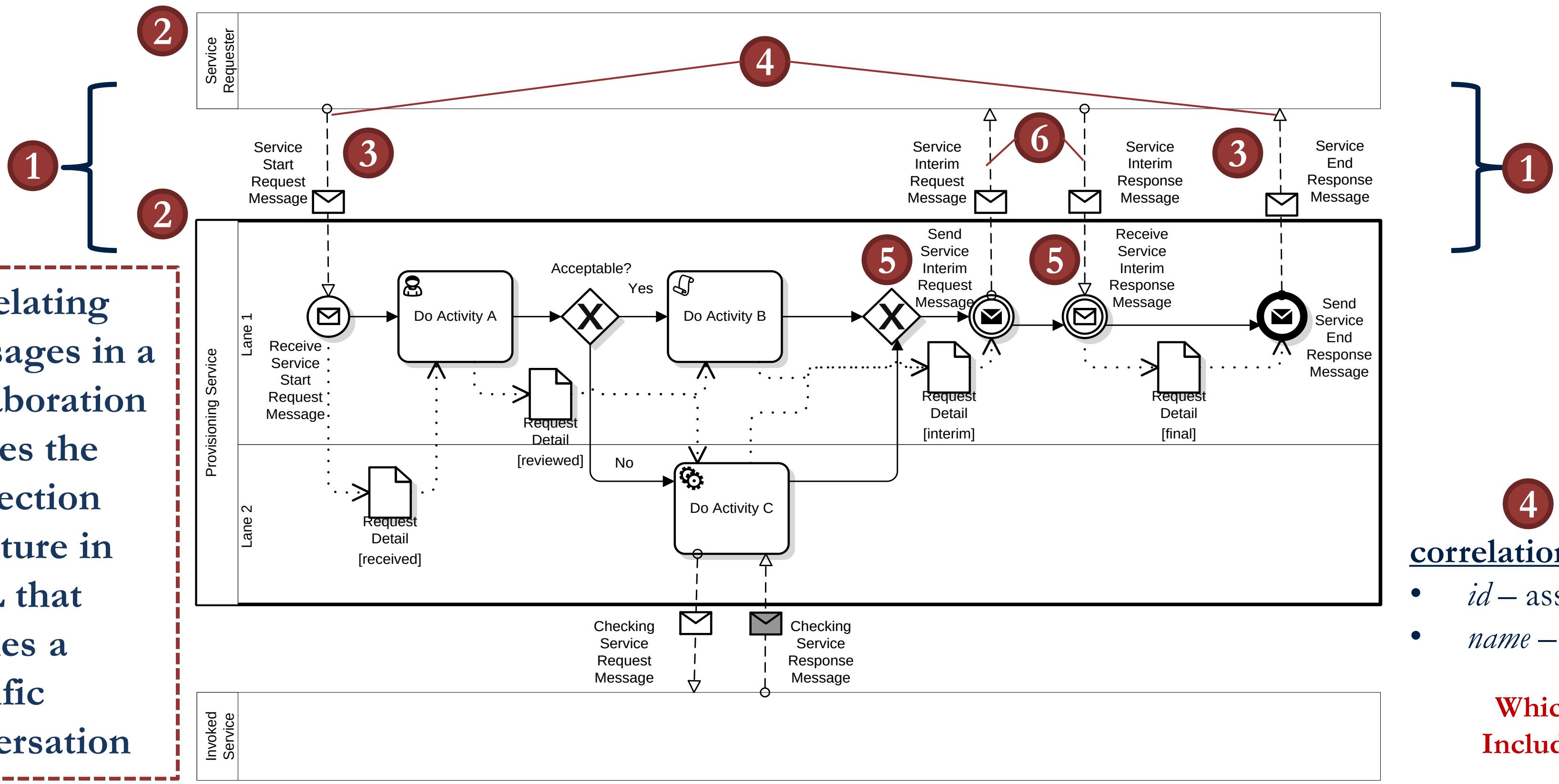
Operation	
SCA Invocation Type = call	Name = Appropriate Name
Implementation Component	
Type = Web Srvc, URI, Other	Name = Corresponds to Type
Messages	
Request to Service	N/A
Message Name/ID = Appropriate Name; Type = Initiating	
Payload = Inbound Subset per Defined Schema for Service	
Correlation Key = Same as for **Response**	

Asynchronous Request/Response – Response



Operation	
SCA Invocation Type = callBack	Name = Appropriate Name
Implementation Component	
Type = Web Srvc, URI, Other	Name = Corresponds to Type
Messages	
N/A	**Response from Service**
	Message Name/ID = Appropriate Name; Type = Initiating
	Payload = Outbound Subset per Defined Schema for Service
	Correlation Key = Same as for **Request**

BPMN XML Attributes (2)



2
Service Requester
4
1
Service Start Request Message
3
Service Interim Request Message
6
Service Interim Response Message
3
Service End Response Message
1
2
Provisioning Service
Lane 1
Lane 2
Receive Service Start Request Message
Do Activity A
Acceptable?
Yes
Do Activity B
5
Send Service Interim Request Message
5
Receive Service Interim Response Message
Send Service End Response Message
Request Detail [reviewed]
Request Detail [interim]
Request Detail [final]
Request Detail [received]
No
Do Activity C
Checking Service Request Message
Checking Service Response Message
Invoked Service

Correlating Messages in a Collaboration creates the indirection structure in XML that defines a specific conversation

4 6

correlationKey
- *id* – assigned
- *name* – entered

Which Includes

1

collaboration
- *id* – assigned
- *name* – entered

Includes One or More

2

participant
- *id* – assigned
- *name* – entered (is pool name)
- *processRef* – is **process** *id* (if a message flow connects to an element within the process)

3 5

messageFlow
- *id* – assigned
- *name* – entered
- **messageRef** – is **message** *id*

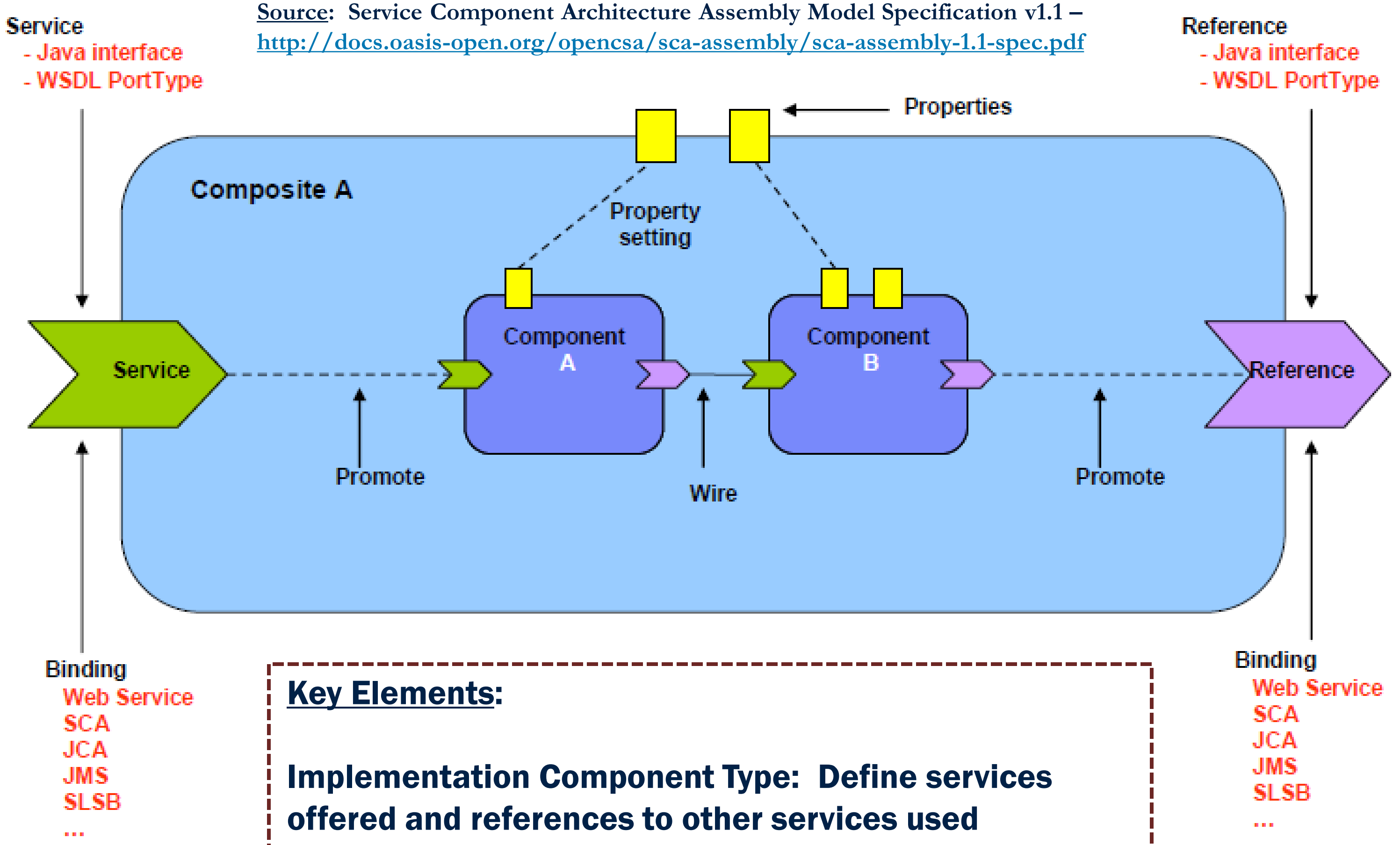
4 6

conversationNode
- *id* – assigned
- *name* – entered
- **messageFlowRef** – is **messageFlow** *id* (one for each message flow)

BPMN & SCA: Translating BPMN Into Service Composition Representations

Service Component Architecture

Source: Service Component Architecture Assembly Model Specification v1.1 – http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec.pdf



Key Elements:

Implementation Component Type: Define services offered and references to other services used

Interface Type: Define one or more business functions with operations and messages

Binding Type: Define access mechanisms and protocols used to call a service

Mapping BPMN To SCA

Key Elements Translated:

Implementation Component – Web Srvc/URI/Other in implementation ref

Interface Type – Message name and type with correlation keys, operations, and WSDL for entire BPMN process

Binding Type – Set to match notionally with Web Srvc/URI/Other in implementation ref

KEY:

Exposed Service(s) =

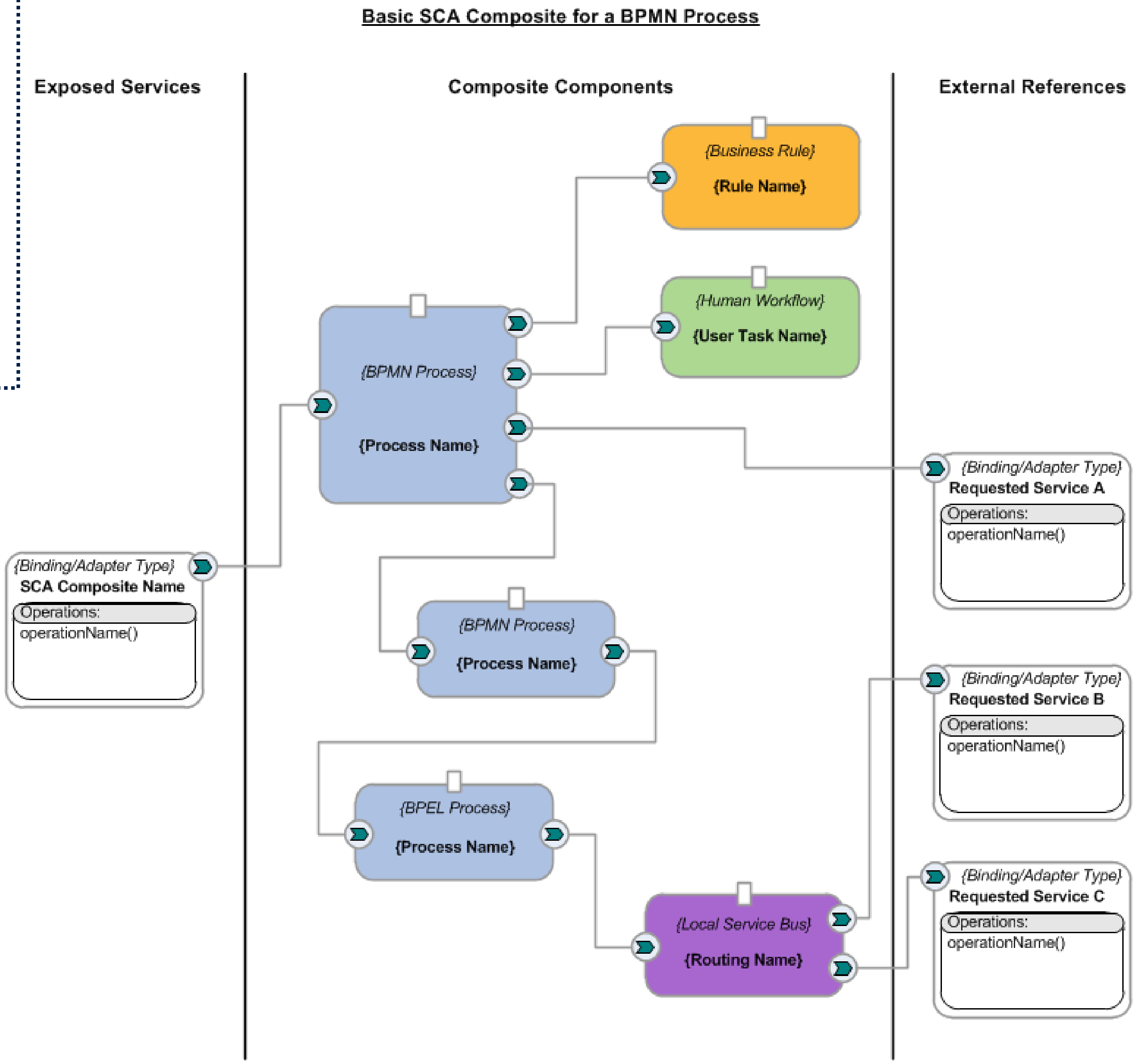
Service interface(s) into the composite's operations

Composite Component(s) =

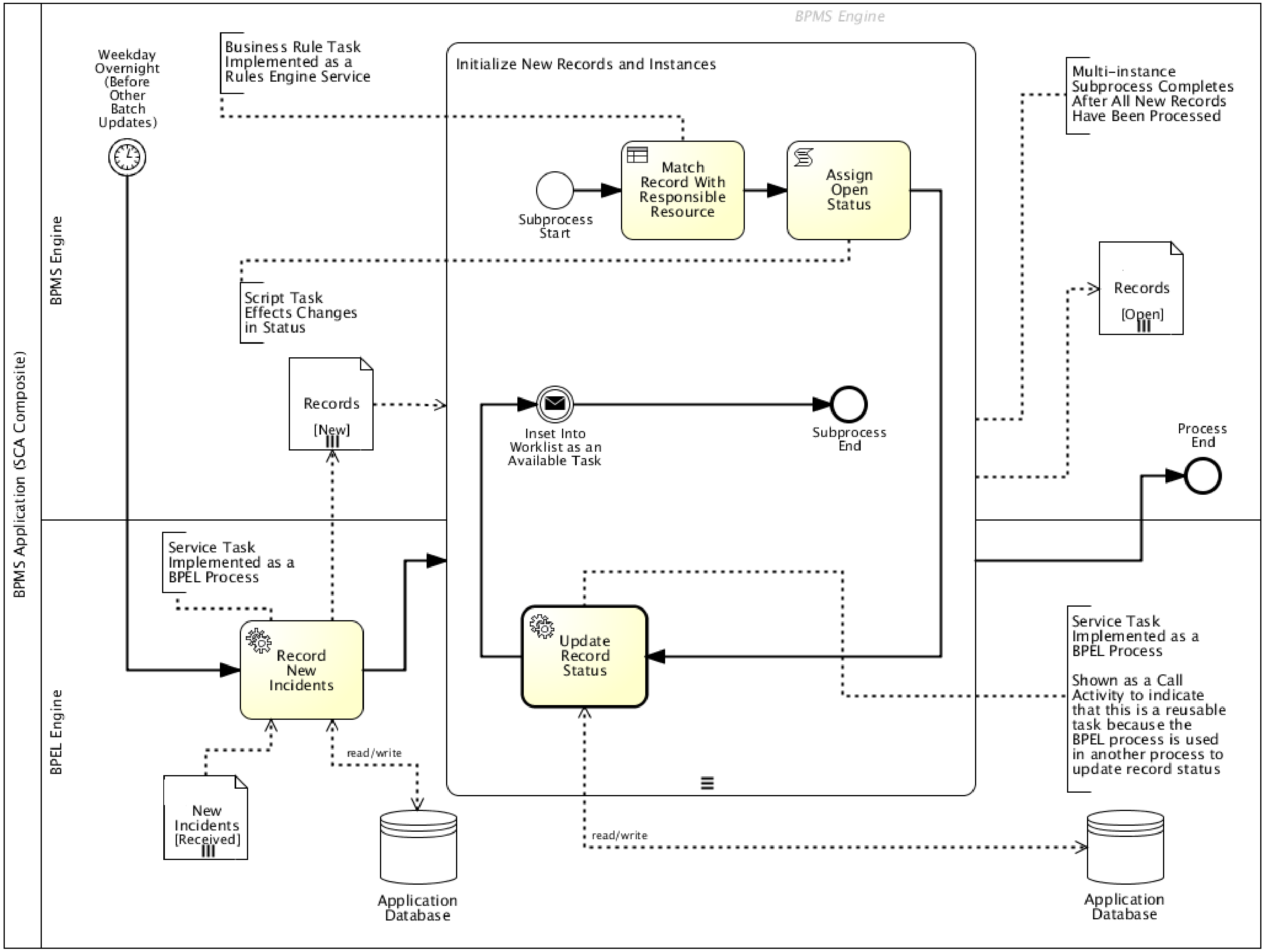
Component(s) wired together in the composite

External Reference(s) =

Service interface(s) into invoked external service(s)



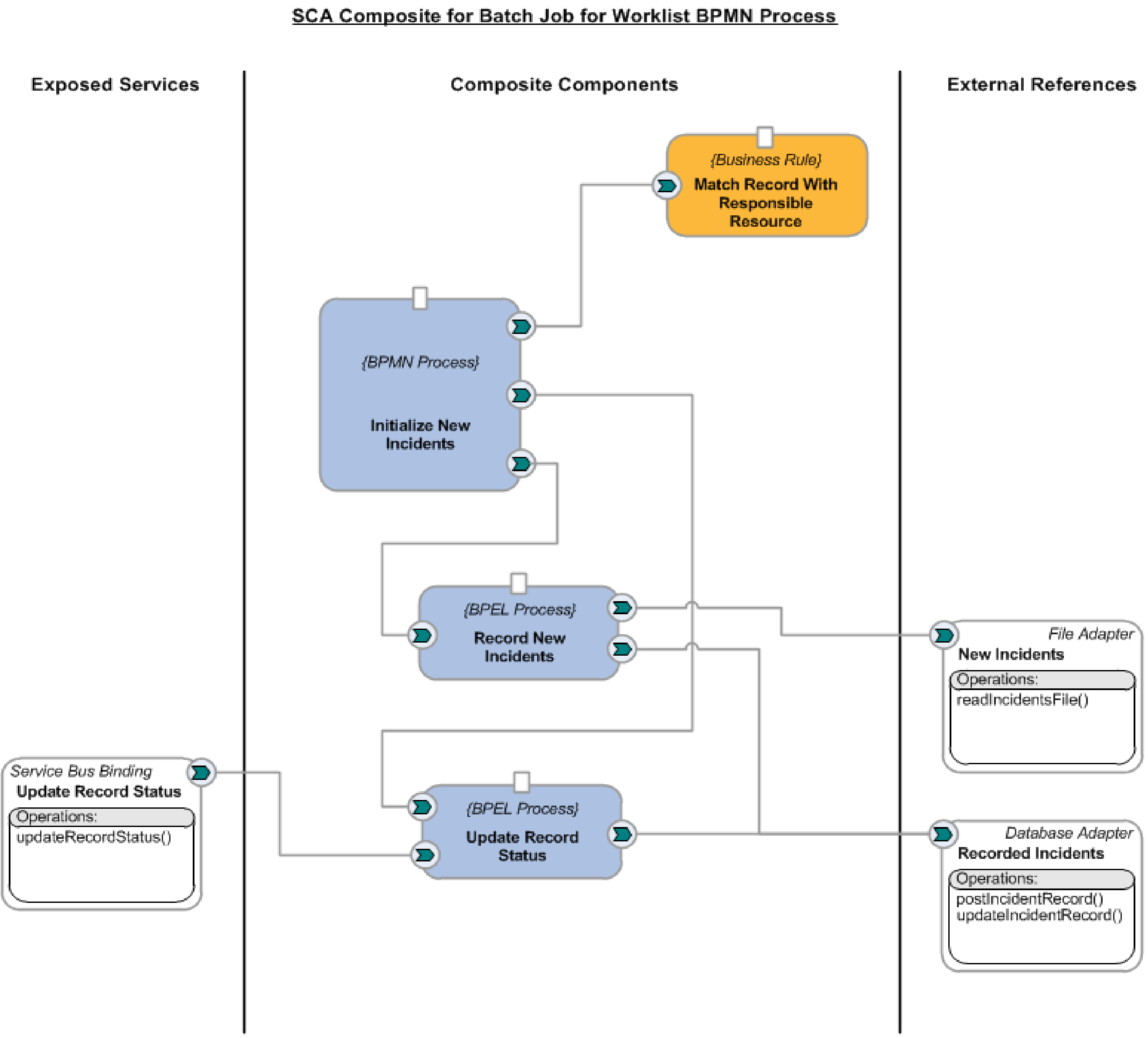
Mapping Example – BPMN View


BPMS Application (SCA Composite)
BPMS Engine
BPEL Engine
BPMS Engine
Weekday Overnight (Before Other Batch Updates)
Business Rule Task Implemented as a Rules Engine Service
Initialize New Records and Instances
Subprocess Start
Match Record With Responsible Resource
Assign Open Status
Multi-instance Subprocess Completes After All New Records Have Been Processed
Records [Open]
Script Task Effects Changes in Status
Records [New]
Inset Into Worklist as an Available Task
Subprocess End
Process End
Service Task Implemented as a BPEL Process
Record New Incidents
Update Record Status
Service Task Implemented as a BPEL Process
Shown as a Call Activity to indicate that this is a reusable task because the BPEL process is used in another process to update record status
read/write
read/write
New Incidents [Received]
Application Database
Application Database

Batch form of the design uses a "flat" model with abstracted components

Maximizes flexibility and reuse in the initialization of the records and setting up for use in the worklist as part of the downstream workflow

Mapping Example – SCA View


SCA Composite for Batch Job for Worklist BPMN Process
Exposed Services
Composite Components
External References
{Business Rule} Match Record With Responsible Resource
{BPMN Process} Initialize New Incidents
{BPEL Process} Record New Incidents
{BPEL Process} Update Record Status
File Adapter New Incidents
Operations: readIncidentsFile()
Database Adapter Recorded Incidents
Operations: postIncidentRecord() updateIncidentRecord()
Service Bus Binding Update Record Status
Operations: updateRecordStatus()

BPMN process provides superstructure for process execution sequence and control – as a private service (no external invocation)

Invoked rules engine (as shown) is within the SCA domain, and so is shown as wired locally

Middle BPEL process reads from batch file to create new DB records, which uses adapter services

Reusable BPEL process is exposed as a public service of the composite

Conclusions and Q&A

Now that BPMN Is also a Visual Programming Language...

We Need Design Patterns and Modeling Guidance!

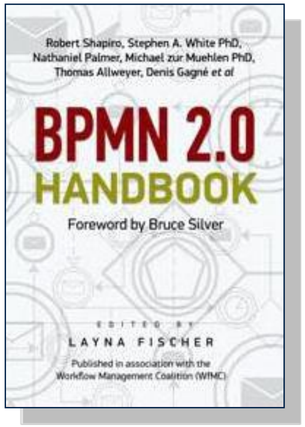
+

Now that BPMN Is Expressible in XML Form…

We Can Translate Model To Service Representations!

Presenters/Authors

- Lloyd Dugan (Presenter)
  - Chief Architect, Business Management Associates (BMA)
    and
    Independent Contractor for Executable BPMN (LAB Derivations/BPMN4SCA)
  - Senior BPMN SME of DoD's DCMO, Co-author of BPMN 2.0 Handbook
  - ldugan@bizmanagers.com or ldugan.bpmn4sca@gmail.com or lloyd.dugan.ctr@osd.mil
- Nathaniel Palmer
  - Chief BPM Strategist and BPM/SOA Practice Director for SRA International
    and
    Executive Director, Workflow Management Coalition (WfMC)
  - Co-author BPMN 2.0 Handbook
  - Nathaniel_Palmer@sra.com



Robert Shapiro, Stephen A. White PhD, Nathaniel Palmer, Michael zur Muehlen PhD, Thomas Allweyer, Denis Gagné *et al*

BPMN 2.0 HANDBOOK

Foreword by Bruce Silver

EDITED BY LAYNA FISCHER

Published in association with the Workflow Management Coalition (WfMC)

See description of book at http://futstrat.com/