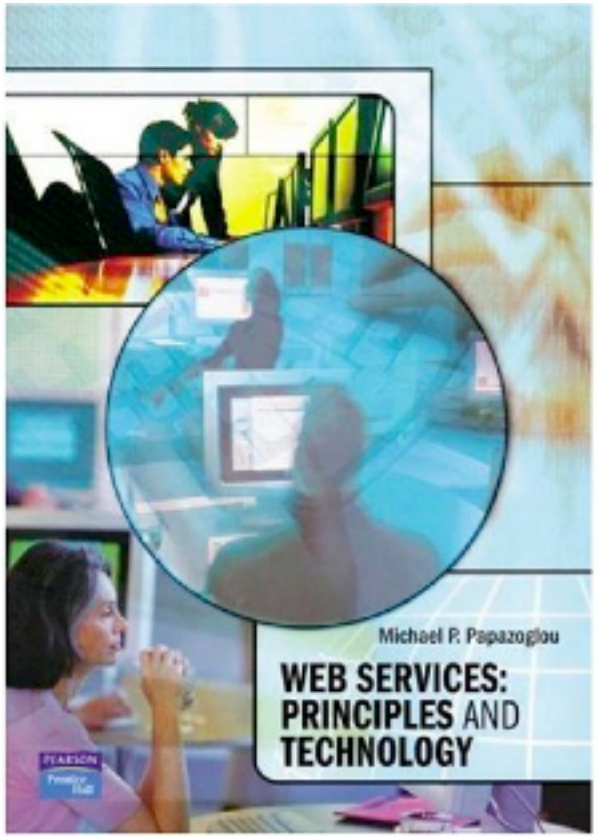
Web Services:
Principles & Technology

Service Analysis & Design

Mike P. Papazoglou
mikep@uvt.nl



UNIVERSITEIT VAN TILBURG

Topics

- *Web services development*
- *Properties of service development methodology*
- *Qualities of service development methodology*
- *Web services development lifecycle*
- *Service analysis*
- *Service design*
- *Service construction*

Web services development methodology

- Introducing a thin SOAP/WSDL/UDDI veneer atop of existing applications or components that implement Web services is by now widely practiced by the software industry.
- A ***methodology*** is of critical importance to specify, construct, refine and customize highly volatile business processes from internally and externally available Web services.
  - A sound methodology helps avoiding the pitfalls of deploying an uncontrolled maze of services & provides a solid foundation for service enablement so that *Web services can be efficiently used in SOA-based business applications*.

Problems with current methodologies

- OOAD, CBD & BPM only address part of the requirements of service-oriented computing applications.
- OOAD, CBD & BPM are not grounded on SOA principles & fail to address the three key elements of an SOA: *services, service assemblies* (composition), and *components realizing services*.

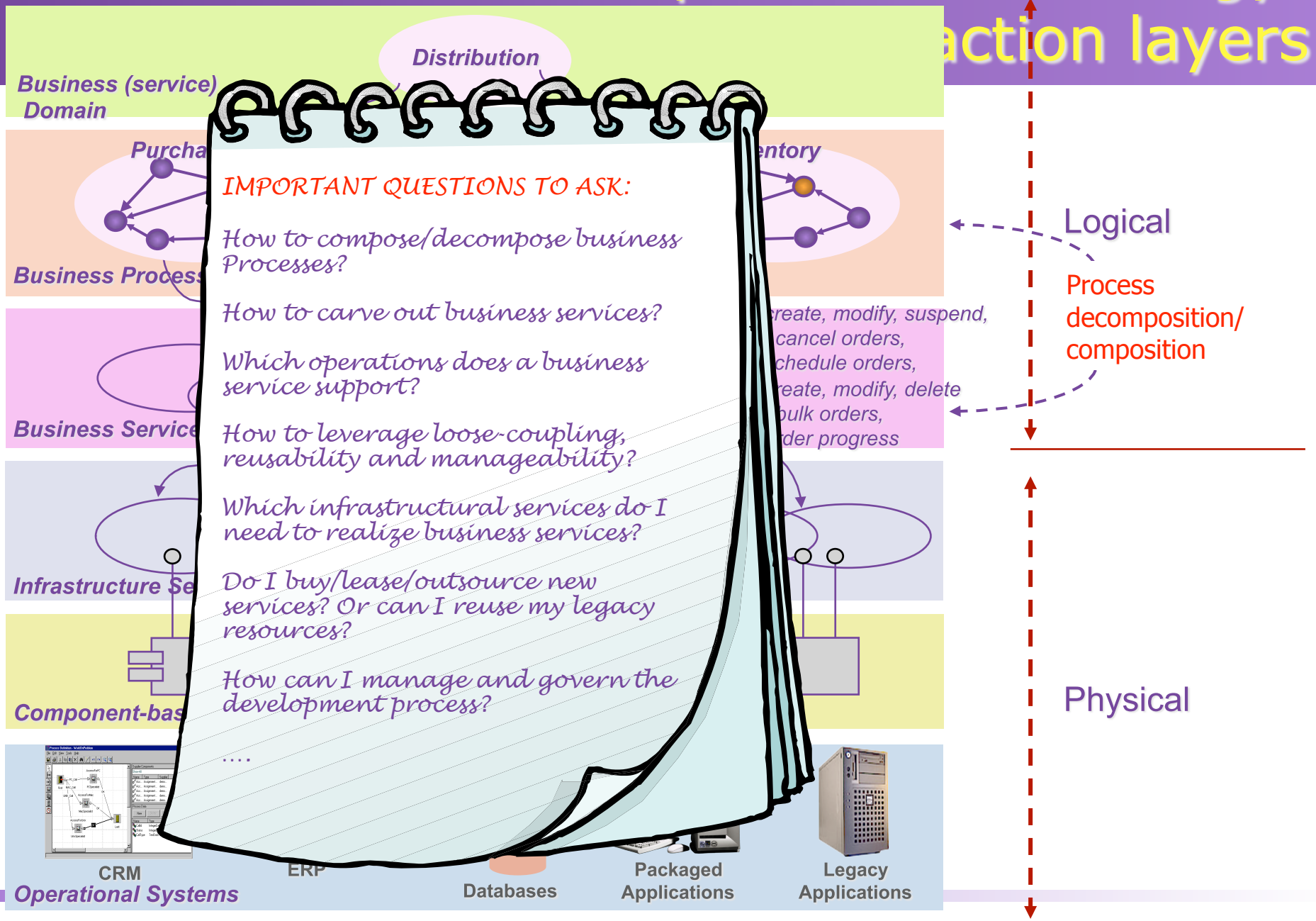
- They also do not support:
  - distributed service development & deployment
  - service provisioning
  - service management

- These practices *fail* when they attempt to develop service-oriented solutions *while being applied independently of each other*.

Topics

- *Web services development*
- *Properties of service development methodology*
- *Qualities of service development methodology*
- *Web services development lifecycle*
- *Service analysis*
- *Service design*
- *Service construction*

SOA development methodology: action layers



Important milestones

- *Re-use existing functionality*
- *Minimize costs of disruption*
- *Employ an incremental mode of integration*
- *Provide increased flexibility*
- *Provide scalability*
- *Provide (design for) compliance with standards*

Service oriented design and development principles

- Service coupling
- Service cohesion
- Service granularity

Service coupling

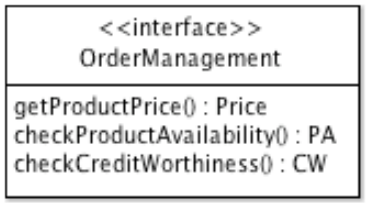
Coupling is the degree of interdependence between any two business processes.

- ***Representational service coupling:*** Services should not depend on specific representational or implementation details and assumptions of one another. Leads to:
  - *Service interchangeability*
  - *Service versioning*
- ***Identity Service coupling:*** Connection channels between services should be unaware of who is providing the service.
- ***Message (protocol) coupling:*** A sender of a message should rely only on those effects necessary to achieve effective communication.

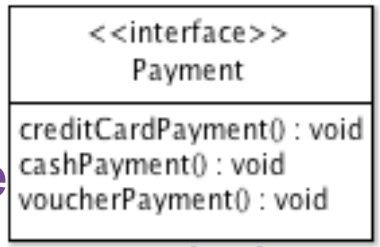
Service cohesion

Cohesion is the degree of the strength of functional relatedness of operations within a service.

- ***Functional Service Cohesion:*** performs one and only one problem-related task and contain only services necessary for that purpose.



- ***Communicational Service Cohesion:*** activities & services use the same input and output messages. Leads to cleanly decoupled business processes.



- ***Logical Service Cohesion:*** services all contribute to tasks of the same general category. They perform a set of independent but logically similar functions (alternatives).

Service granularity

- **Service granularity** is the scope of functionality exposed by a service.

- Services may come at two levels of granularity:
  - A coarse-grained interface might be the complete processing for a given service, e.g., “SubmitPurchaseOrder”
    - the message contains all business information needed to define a PO
    - A fine-grained interface might have separate operations for: “CreateNewPO”, “SetShippingAddress”, “AddItem”, etc.

- *Fine-grained services* usually provide basic data access or rudimentary operations.
- *Coarse-grained services* are composed from finer grained services.

Service granularity concerns

- A service interface (WSDL port type) should generally contain more than one operation, supporting a particular business activity

- The frequency of message exchange is an important factor. Sending & receiving more info. in a single request is more efficient than sending many fine-grained messages.
  - *Several redundant, fine-grained services lead to increased message traffic & tremendous overhead & inefficiency.*
  - *A small collection of coarser-grained services - each of which implements a complete business process - that are usable in multiple scenarios is a better option.*
- Heuristics identify the right level of granularity for services.

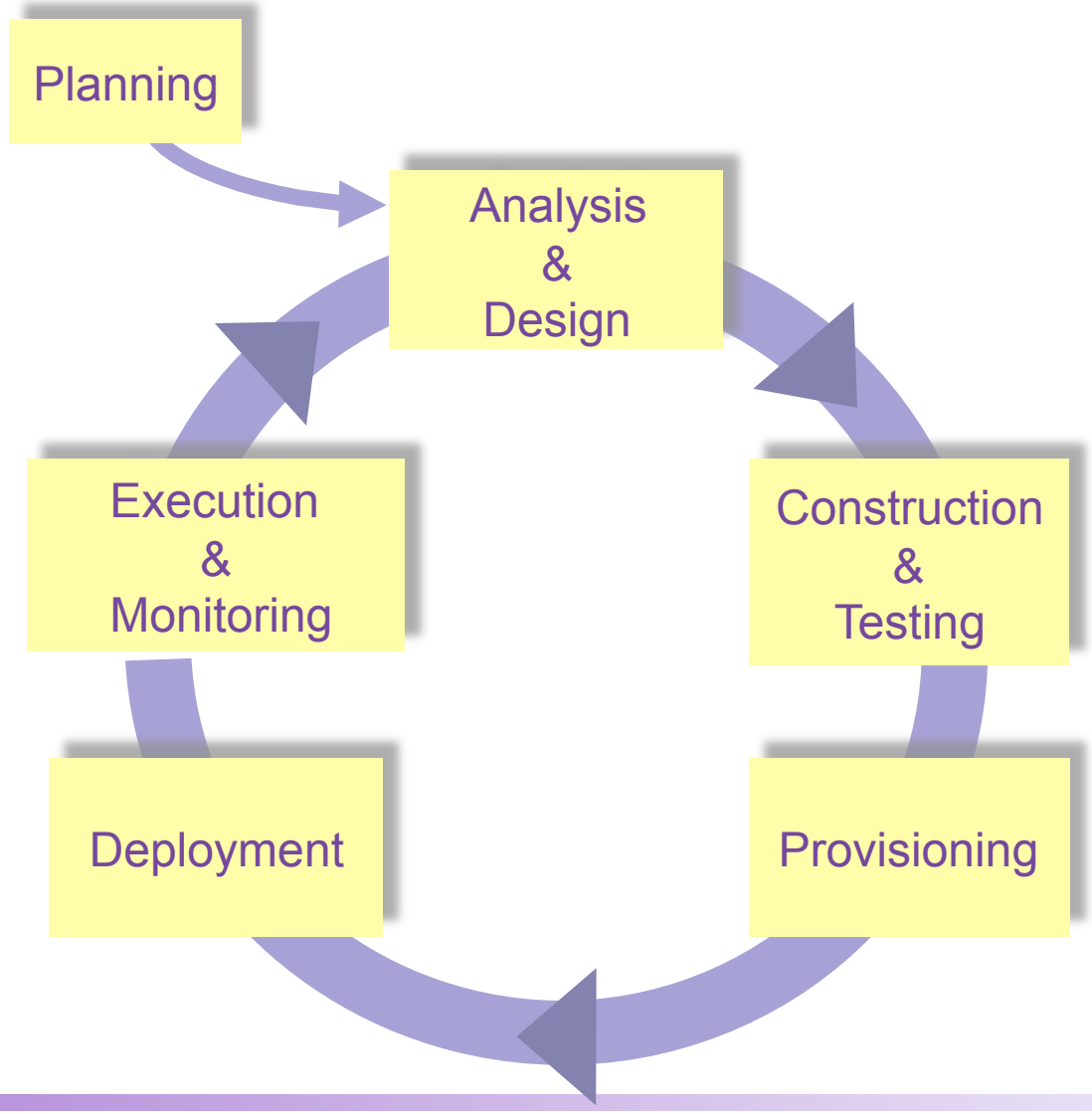
Service oriented design and development concerns

- Manage the entire services lifecycle;
- Establish a platform, programming model and managing services
- Adopt “best-practices” and tools;
- Deliver high-quality SOA solutions

Topics

- *Web services development*
- *Properties of service development methodology*
- *Qualities of service development methodology*
- *Web services development lifecycle*
- *Service analysis*
- *Service design*
- *Service construction*

Web services development lifecycle



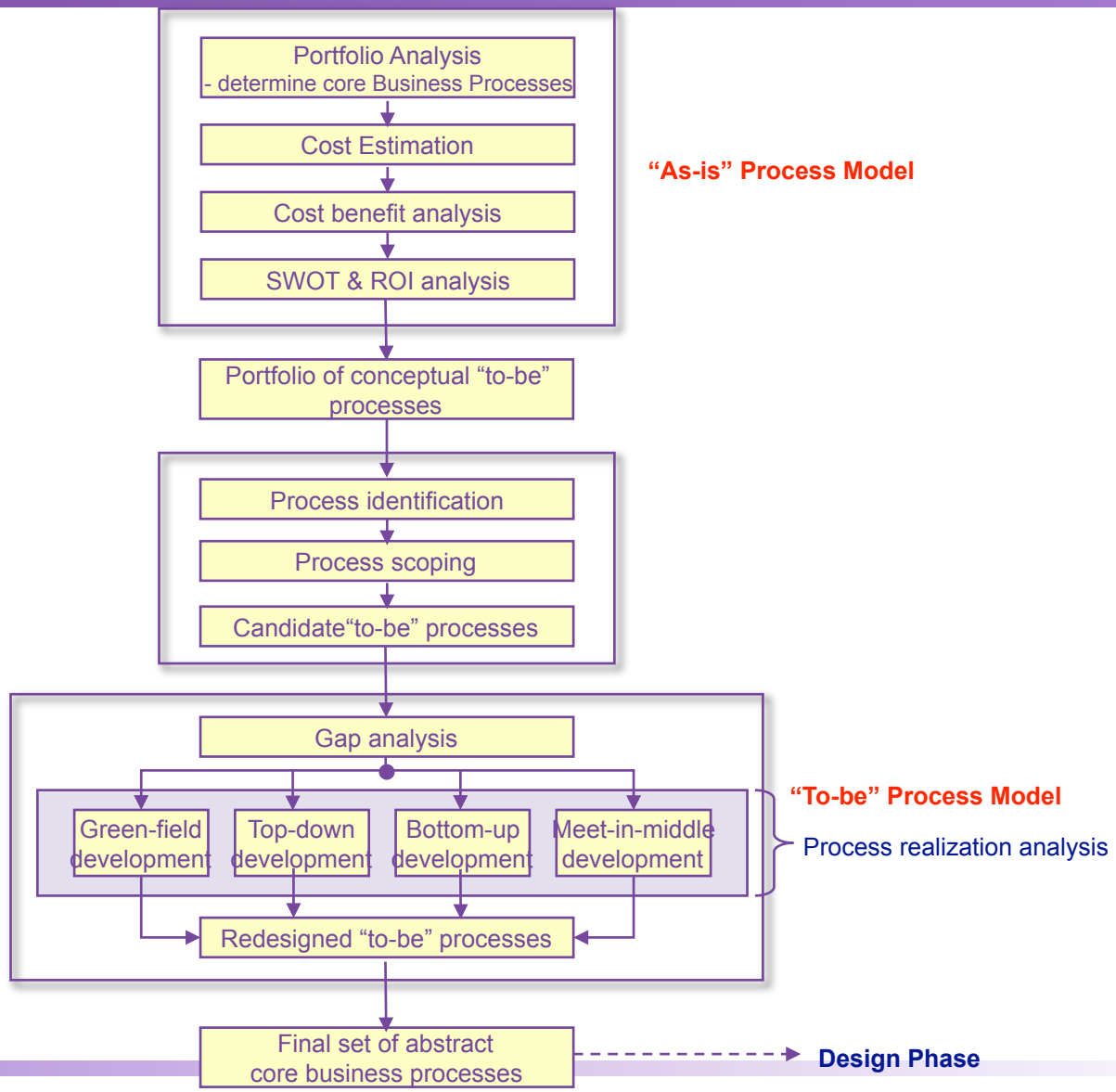
Topics

- *Web services development*
- *Properties of service development methodology*
- *Qualities of service development methodology*
- *Web services development lifecycle*
- *Service analysis*
- *Service design*
- *Service construction*

Service analysis

- Service analysis aims at identifying & describing the processes in a business problem domain; on discovering potential overlaps & discrepancies between processes under construction & available system resources needed to realize services.
- It helps prioritize business processes where SOA can contribute to improvements & offer business value potential.
- It identifies, conceptualizes, and rationalizes business processes as a set of interacting services;
- Develops in-depth understanding of functionality, scope, reuse and granularity of candidate business processes & services.

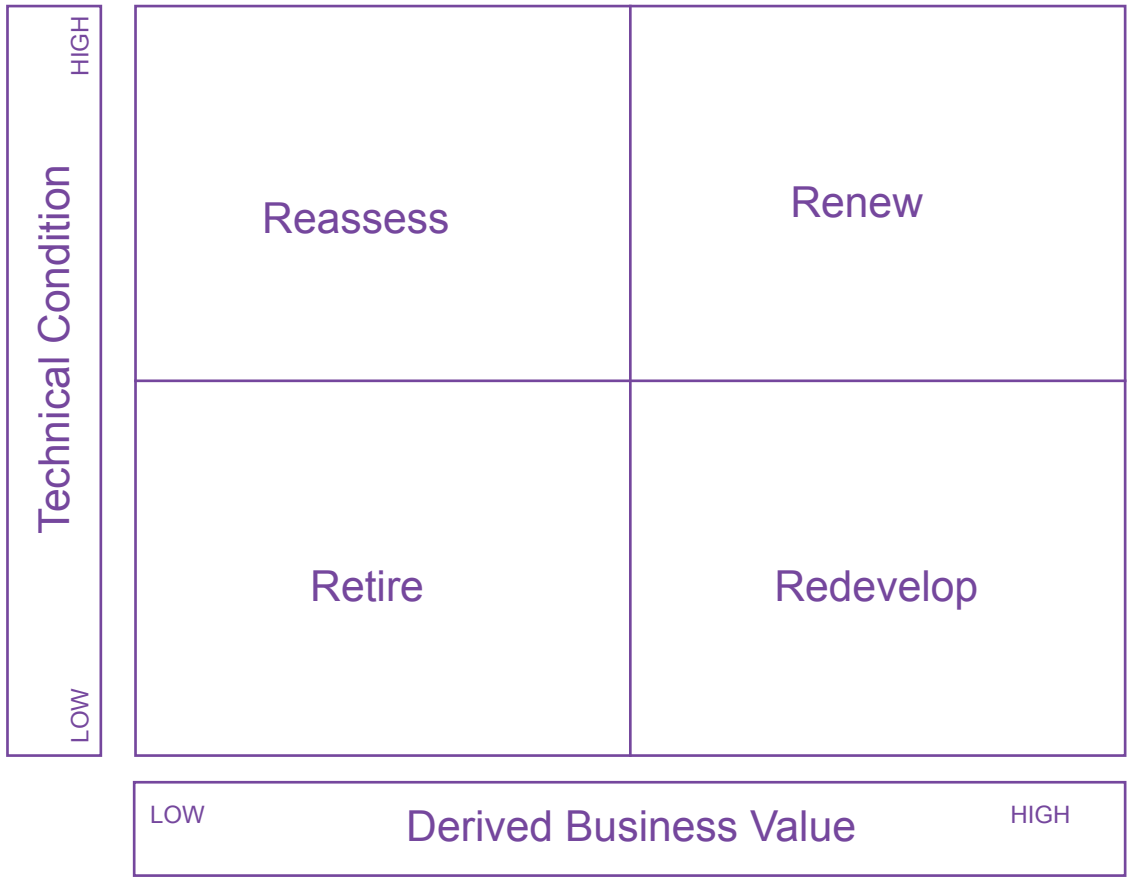
Service analysis steps



Portfolio analysis

- Portfolio analysis requires that apps and business processes that are candidates for re-engineering be prioritized according to their technical quality and business value.
  - Business objectives are derived from business strategies and processes, while technical assets are mapped and weighted against these business objectives and evaluated using a comprehensive set of metrics.
- Current apps and processes are assessed using various techniques including:
  - Reverse-engineering
  - Architectural Knowledge Elicitation
  - Business Process Mapping
  - Process Mining

Portfolio analysis (cntd)

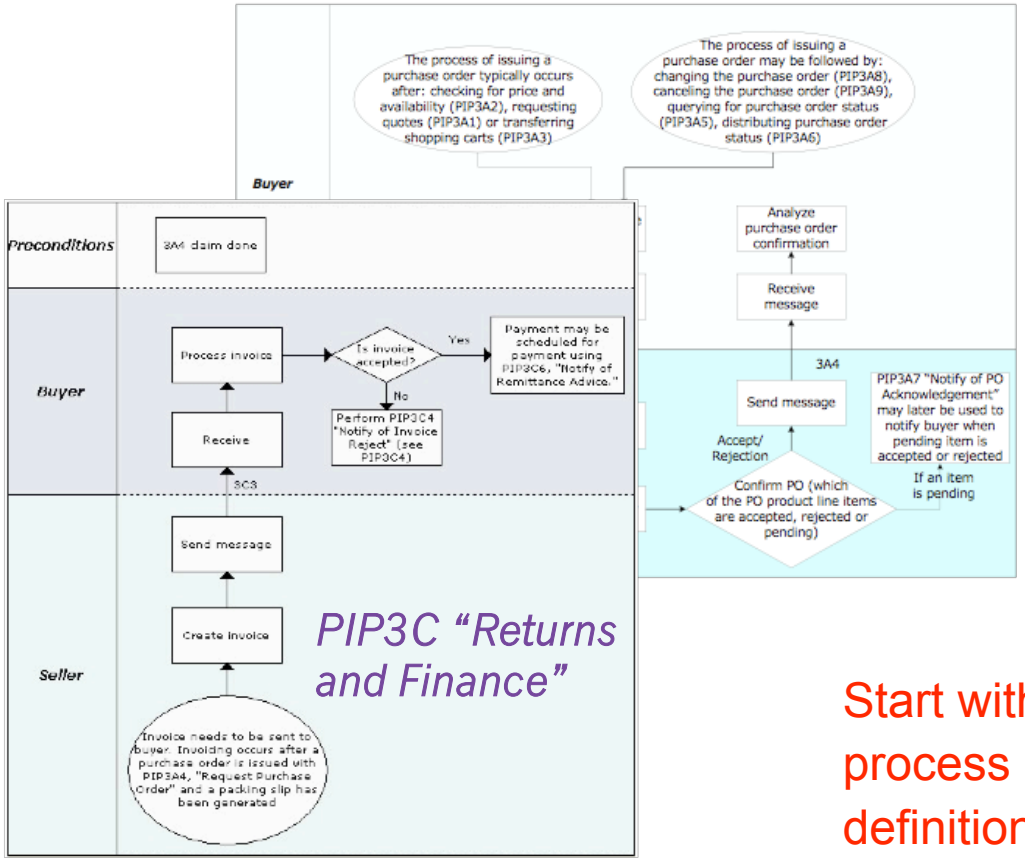


Service identification & scoping

- Service identification inspects enterprise core business entities, ascertains business concepts & leads to the formation of conceptual business processes and business services. Takes into account important issues such as:
  - consolidation,
  - decomposition,
  - reuse, simplification &
  - refactoring of legacy assets
- Service scoping defines the scope of a business process as an aggregation of aspects that include:
  - where the process starts and ends, typical customers,
  - inputs & outputs that customers expect to see,
  - external entities that the process is expected to interface with,
  - different types of events that start an instance of the process.

Service identification

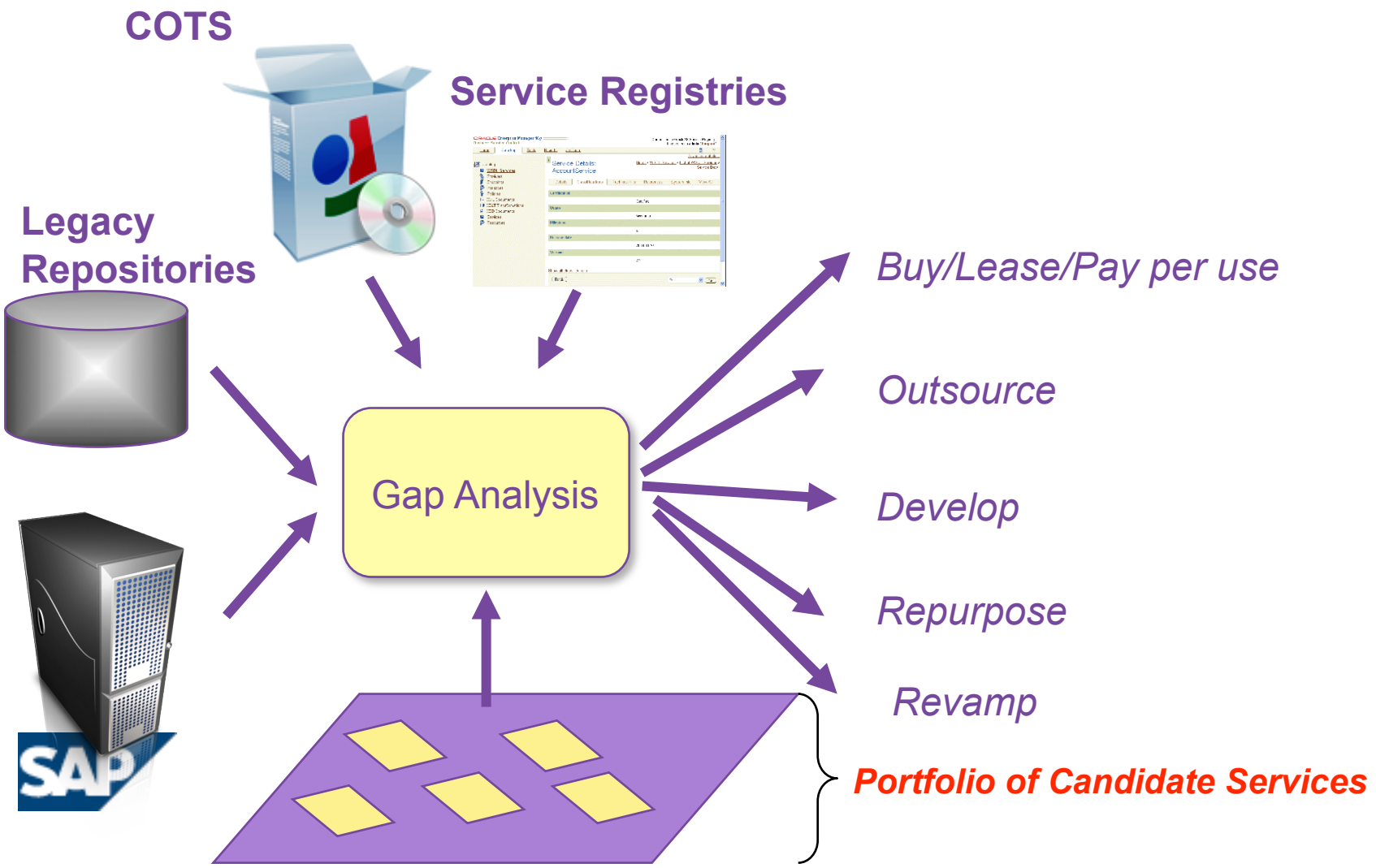
RosettaNet PIP3A4 “Manage Purchase Order”



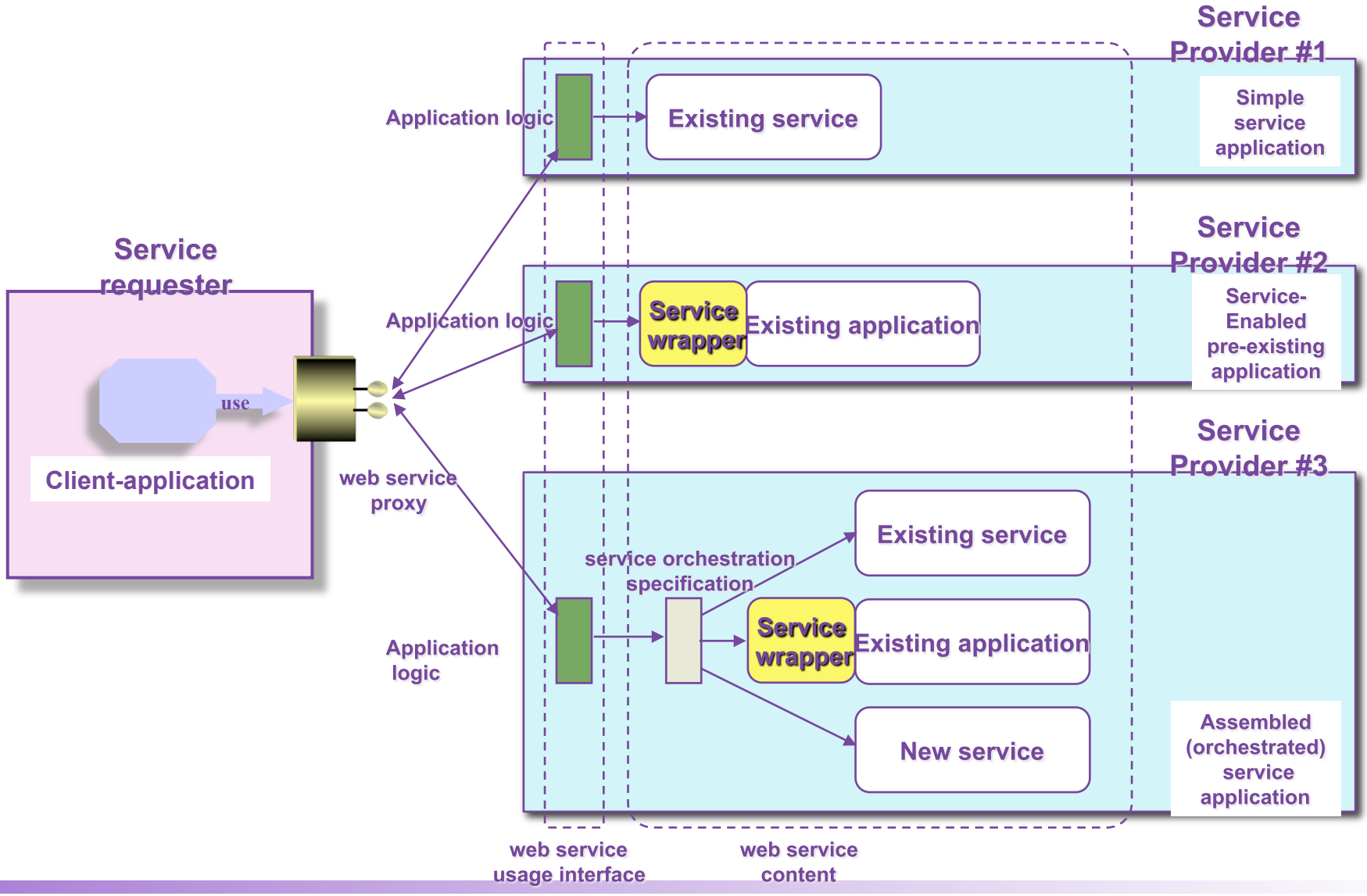
PIP3A, 3C

Start with comparing the abstract business process portfolio with standard process definitions, e.g., RosettaNet

Business service gap analysis



Realization architecture & services portfolio



Business service realization analysis

- Green-field development: describes how new interface for a service will be created.
- Top-down development: a new service can be developed that conforms to an existing service interface.
- Bottom-up development: a new service interface is developed for an existing application.
- Meet-in-the-middle development: this option is used when an already existing service interface - for which an implementation already exists – is partially mapped onto a new service or process definition.

Service realization alternatives

1. Reusing or repurposing already existing Web services, business processes or business process logic.
3. Developing new Web services or business processes logic from scratch.
5. Purchasing/leasing/paying per use for services.
6. Outsourcing service design and implementation of Web services or (parts of) business processes.
7. Using wrappers and/or adapters to revamp existing enterprise (COTS) components or existing (ERP/legacy) systems.
   - Revamping software components including database functionality or legacy software results in introducing service-enabling implementations for these systems in the form of adapters or wrappers.

Topics

- *Web services development*
- *Properties of service development methodology*
- *Qualities of service development methodology*
- *Web services development lifecycle*
- *Service analysis*
- *Service design*
- *Service construction*

Service design concerns

- Service design requires developers to define related, well-documented interfaces for all conceptual services identified by the analysis phase, prior to constructing them.
- The design phase encompasses the steps of:
  - singular service specification,
  - business process specification, and
  - policy specification for both singular services and business processes.
- Service design is based on a twin-track design approach that provides two production lines – one along the logical part and one along the physical part of the SOA – and considers both functional and non-functional service characteristics.

Service design

- Concerns: design for reuse, design for composition and granularity
- Specification: structural specification, behavioral specification, service programming style and policy specification.
- Choice of “right” standards, e.g., orchestration vs. choreography

Specifying service types

```
<wsdl:types>
      <xsd:complexType name = "PIP3A4PurchaseOrderRequest">
            <xsd:sequence>
                  <xsd:element ref = "PurchaseOrder"/>
                  <xsd:element ref = "fromRole"/>
                  <xsd:element ref = "toRole"/>
                  <xsd:element ref = "thisDocumentGenerationDateTime"/>
                  <xsd:element ref = "thisDocumentIdentifier"/>
                  <xsd:element ref = "GlobalDocumentFunctionCode"/>
            </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name = "PurchaseOrder">
            <xsd:sequence>
                  <xsd:element ref = "deliverTo" minOccurs = "0"/>
                  <xsd:element ref = "comment" minOccurs = "0"/>
                  <xsd:element ref = "packListRequirements" minOccurs = "0"/>
                  <xsd:element ref = "ProductLineItem" maxOccurs = "unbounded"/>
                  <xsd:element ref = "GlobalShipmentTermsCode"/>
                  <xsd:element ref = "RevisionNumber"/>
                  <xsd:element ref = "prePaymentCheckNumber" minOccurs = "0"/>
                  <xsd:element ref = "QuoteIdentifier" minOccurs = "0"/>
                  <xsd:element ref = "WireTransferIdentifier" minOccurs = "0"/>
                  <xsd:element ref = "AccountDescription" minOccurs = "0"/>
                  <xsd:element ref = "generalServicesAdministrationNumber"
                                    minOccurs = "0"/>
                  <xsd:element ref = "secondaryBuyerPurchaseOrderIdentifier"
                                    minOccurs = "0"/>
                  <xsd:element ref = "GlobalFinanceTermsCode"/>
                  <xsd:element ref = "PartnerDescription" maxOccurs = "unbounded"/>
                  <xsd:element ref = "secondaryBuyer" minOccurs = "0"/>
                  <xsd:element ref = "GlobalPurchaseOrderTypeCode"/>
            </xsd:sequence>
      </xsd:complexType>
</wsdl:types>
  … … … …
<message name="PurchaseOrderRequest">
      <part name="PO-body" type="tns:PIP3A4PurchaseOrderRequest"/>
</message>
```

Type definitions for POs
& PO requests in RosettaNet.

Specifying service interfaces

```
  … … … …
<portType name="CanReceive3A42_PortType">
     <!-- name of operation is same as name of message -->
     <operation name="PurchaseOrderRequest">
           <output message="tns:PurchaseOrderRequest"/>
     </operation>
     <operation name="ReceiptAcknowledgement">
           <output message="tns:ReceiptAcknowledgment"/>
     </operation>
</portType>

<portType name="CanSend3A42_PortType">
     <!-- name of operation is same as name of message -->
     <operation name="PurchaseOrderConfirmation">
           <input message="tns:PurchaseOrderConfirmation"/>
     </operation>
     <operation name="ReceiptAcknowledgment">
           <input message="tns:ReceiptAcknowledgment"/>
     </operation>
     <operation name="Exception">
           <input message="tns:Exception"/>
     </operation>
</portType>
   … … … …
```

Interface definitions for PO requests in RosettaNet.

Specifying business processes

- Describe the process structure:
  - Identify and group activities that implement a business process;
  - Describe activity dependencies, conditions or synchronization
  - Describe implementation of the business process.
- Describe roles.
- Capture non-functional process concerns, e.g., security and transactions.

Process flow

```
<sequence>
     <receive   partner="Manufacturer" portType="lns:PurchaseOrderPortType"
                      operation="Purchase"   variable="PO"
                      createInstance="yes" >
    </receive>
 <flow>
      <links>
            <link name="inventory-check"/>
            <link name="credit-check"/>
      </links>
     <!– Check inventory –->
     <invoke partner="inventoryChecker"
                   portType="lns:InventoryPortType"
                   operation="checkInventory"
                   … … …
              <source linkName="inventory-check"/>
          …
     <invoke/>
     <!– Check  credit –->
     <invoke partner="creditChecker"
                   portType="lns:CreditCheckPortType"
                   operation="checkCredit"
                   … … …
              <source linkName="credit-check"/>
          ….
     <invoke/>
     <!– Issue bill once inventory and credit checks are succesful –->
     <invoke partner="BillingService"
                   portType="lns:BillingPortType"  operation="billClient"
                   inputVariable="billRequest"  outputVariable="Invoice" >
                   joinCondition="getLinkStatus("inventory-check") AND
                                        getLinkStatus("credit-check")" />
              <target linkName="inventory-check"/>
              <target linkName="credit-check"/>
          ….
     <invoke/>
 </flow>
    ...
 <reply partnerLink="Purchasing" portType="lns:purchaseOrderPT"
                           operation="Purchase" variable="Invoice"/>
</sequence>
```

BPEL process flow for PO process

Defining roles

```
<portType name="PurchaseOrderPortType">
    <operation name="Purchase">
    </operation>
</portType>
```

Purchase process WSDL PortType

```
<partnerLinkType name="PurchaseOrderPLT">
  <role name="PurchaseService">
     <portType name="PurchaseOrderPortType"/>
  </role>
</partnerLinkType>

<partnerLinkType name="CreditCheckPLT">
  <role name="CreditChecker">
     <portType name="CreditCheckPortType"/>
  </role>
  <role name="CreditCheckRequester">
     <portType name="CreditCheckCallbackPortType"/>
  </role>
</partnerLinkType>
```

Purchase process partner link types

```
<partnerLinks>
    <partnerLink name= "Purchasing" partnerLinkType="PurchaseOrderPLT"
      myRole= "PurchaseService" />

    <partnerLink name="CreditChecking" partnerLinkType="CreditCheckPLT"
      myRole="CreditRequester" partnerRole="CreditChecker"/>

    <partnerLink name="Billing" partnerLinkType="BillingPLT"
      myRole="BillRequester" partnerRole="Biller"/>
</partnerLinks>
```

Purchase process partner link declarations

BPEL roles for PO process

Specifying service policies

Service policies could specify three sets of constraints:

- *Business constraints:* such as operating ranges, regulatory and legal constraints, or standards established in specific vertical industries.
- *Technology constraints:* based on choices, decisions, & commitments to specific technologies in current & continued use in the enterprise infrastructure
  - e.g., choice of specific application packages, legacy applications, commitments to industry specific standards, etc.
- *Runtime quality constraints:* services aspects that are directly related to system dynamics e.g., performance, scalability, transactional integrity, security, and fault tolerance.
  - In SOAs runtime qualities are captured by SLAS.

Services integration model

- A services integration model facilitates the design of a service integration strategy to solve integration and interoperability problems mainly between interacting enterprises. This strategy considers:
  - service design models, policies, SOA governance options & organizational & industry best practices and conventions.
- The SOA integration model is based on service integration principles:
  - *service relationship principle:* regards services relationships founded in business processes in terms of service producers & consumers during service design.
  - *service transportation principle:* regards message interceptors or brokers interposed between service consumers & providers.
  - *service delivery principle:* describes the mechanisms which service intermediaries, consumers & producers use to deliver messages to endpoints.
  - *process flow principle:* describes how service conversations and behavioral activities in which service providers & consumers are engaged in are influenced by business, technical and environmental requirements.

Topics

- *Web services development*
- *Properties of service development methodology*
- *Qualities of service development methodology*
- *Web services development lifecycle*
- *Service analysis*
- *Service design*
- *Service construction*

Constructing a service