Comparing WSDL-based and ebXML-based Approaches for B2B Protocol Specification

ICSOC'03, Trento, December 2003

Martin Bernauer
Gerti Kappel
>Gerhard Kramler
{lastname}@big.tuwien.ac.at

Business Informatics Group

Institute of Software Technology and Interactive Systems
Vienna Universiy of Technology
Favoritenstraße 9-11/188-4 . 1040 Vienna, Austria
Tel.: +43 (1) 58801 - 18828, Fax: +43 (1) 58801 - 18896
http://www.big.tuwien.ac.at

Outline

- Motivation
- Framework for Comparison
- Overview of Approaches
- *foreach* Framework Layer
  - Languages
  - Concepts and Evaluation
- Summary, Conclusion, and Future Work

Motivation

- Why B2B Protocol Specification?
  - Application-to-Application integration across organizations
  - EDI based on minimal infrastructure → expensive
  - Custom protocols
  - Generic infrastructure (validation, monitoring, integration support, etc.)
- Why this Comparison?
  - Two approaches proposed for the same problem domain, but different concepts are used
    - Web Services have good tool support, dynamic development (policies, transactions, etc.)
    - ebXML provides strong conceptual background (UN/CEFACT Modeling Methodology)
  - What are the differences? Which approach is "better"?

Framework for Comparison (1)

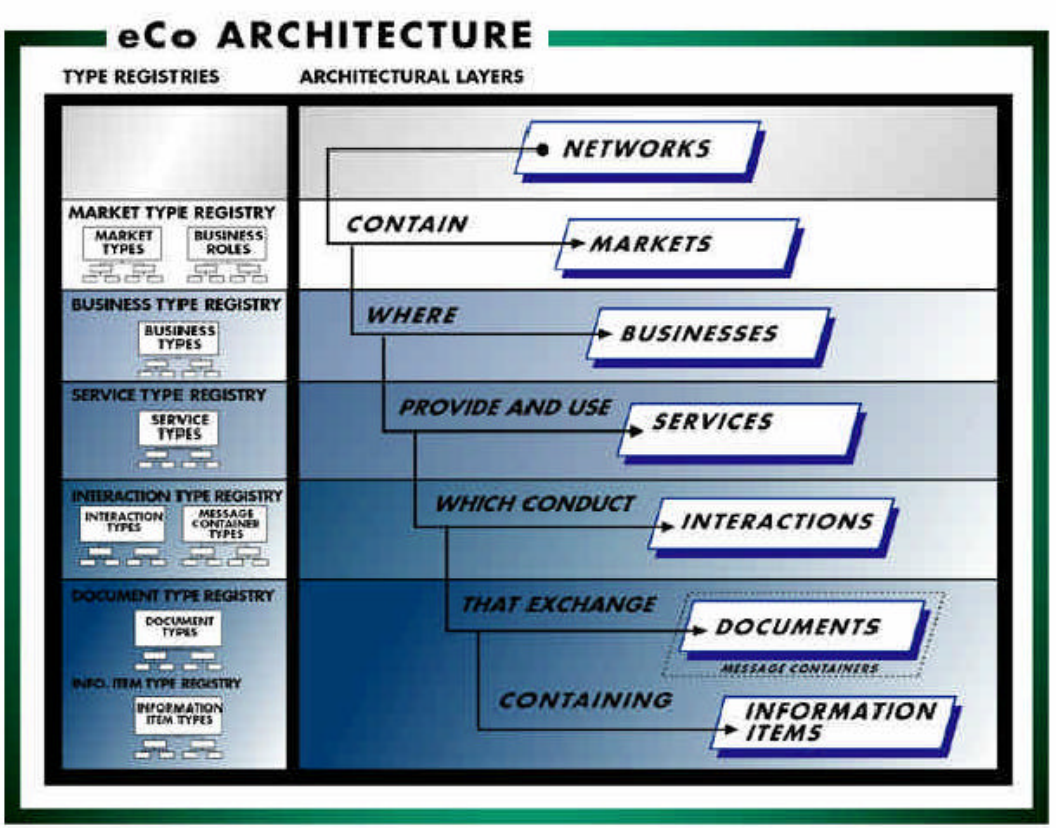
Two Base Frameworks

- eCo Framework
  - Conceptual architecture for electronic commerce interoperability
  - Description of an e-commerce system at different layers
    - Information Items, Documents, …
  - http://eco.commerce.net/rsrc/eCoSpec.pdf (1999)

- Aspects of Workflow Modeling
  - Reduce complexity of workflow models by considering different aspects of a workflow independently
    - Functionality, Behavior, Transactions, …
  - Aspects identified by S. Rausch-Schott and S. Jablonski
  - Security as additional aspect for B2B protocols added

Framework for Comparison (2)

eCo Layers



Framework for Comparison (3)

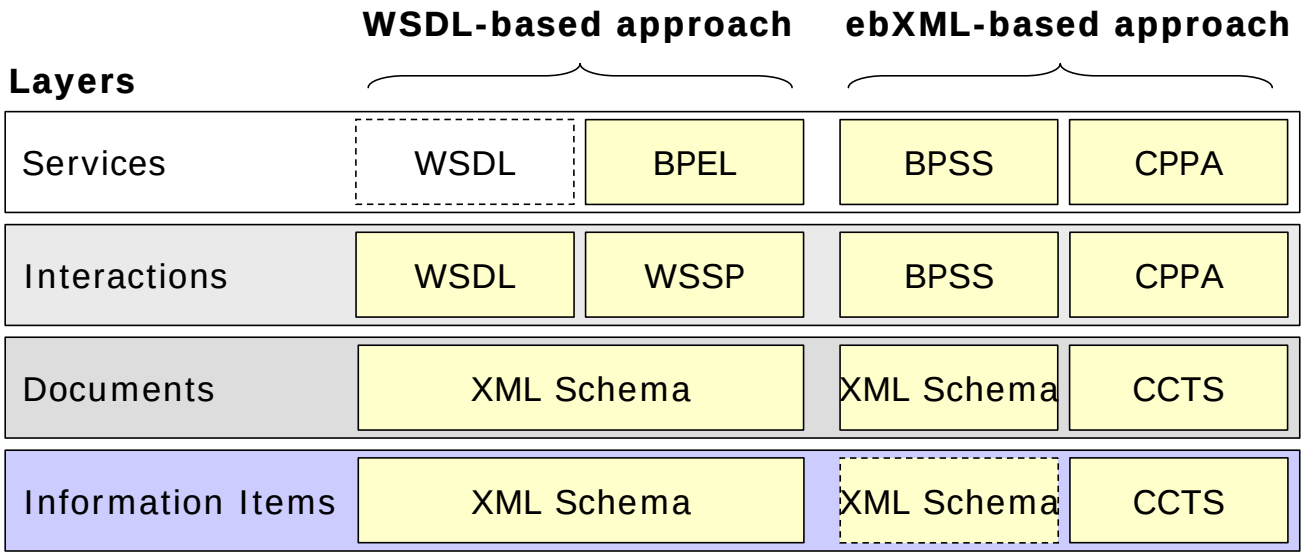
eCo Layers * Workflow Aspects

eCo Layers \ Wf Aspects	Functionality	Organization	Information	Behavior	Security	Transaction	Causality	Operation
Services	X	X	X	X		X		
Interactions	X		X	X	X	X		X
Documents			X				X	X
Information Items			X					

X ... supported combination

Overview of Approaches

Languages Employed per Layer



WSDL – Web Services Description Language
BPEL – Business Process Execution Language for Web Services
WSSP – Web Services Security Policy
BPSS – Business Process Specification Schema
CPPA – Collaboration-Protocol Profile and Agreement Specification
CCTS – Core Components Technical Specification

Information Items (1)

Languages

- WSDL-based approach: **XML Schema**
  - Defines the structure of XML documents
  - Type structuring mechanisms similar to OO languages (types, extension/restriction, etc.)
  - Base language for some standardizations (OAGIS, UBL)
- ebXML-based Approach: **CCTS / Core Components**
  - A conceptual language and methodology for the identification of generic re-usable information items (→ "core components")
  - Independent of implementation languages like XML/Schema, EDI
    - also not specific/limited to ebXML
  - Independent of specific business/application context
    - CCs are abstract – not intended for direct instantiation, but for use in particular contexts/applications

OAGIS – Open Applications Group Integration Specification
UBL – Universal Business Language

Information Items (2)

Concepts Supported

- Informational aspect: syntax, semantics
  - primitive types (predefined)
  - simple value types (incl. metadata)
  - complex value types (incl. metadata)
    - simple value properties
    - complex value properties
    - *referential properties*
    - *associative properties*
  - *specialization (extension, restriction)*
  - value constraints
  - *order constraints*
  - *library of types*

WSDL	ebXML
X	X
X	X
X	X
X	X
X	X
X	–
X	–
X	–
X	X
X	–
–	X

X ... feature supported
– ... feature not supported

Documents (1)

Languages

- WSDL-based approach: **XML Schema**
  - The same concepts used (no two-layered architecture)
- ebXML-based approach: **CCTS / Business Information Entities**
  - Core components are adapted to a specific usage context (→ "Business Information Entity")
  - Predefined kinds of context (business process, product, industry, region, etc.)
  - Constraint language to adapt and aggregate core components: Core Component * context → Business Information Entity
  - Independent of implementation languages like XML/Schema, EDI
    - needs to be translated to an implementation language ("syntax binding")

Documents (2)

Concepts Supported

	WSDL	ebXML
■ Informational aspect: adaptation and aggregation		
● adaptation of information items		
○ add (extension)	X	X
○ *remove (restriction)*	–	X
○ *rename*	–	X
○ value constraints (restriction)	X	X
● aggregation of information items	X	X
■ Causal aspect: design rationale		
● *context of adaptation and aggregation*	–	X
■ Operational aspect: instance syntax		
● *technology independent*	–	X
● *defined representation*	X	–

X ... feature supported
– ... feature not supported

Interactions (1)

Languages

- WSDL-based approach: **WSDL & WS-Policy**
  - WSDL to specify asynchronous messaging and RPC-style interfaces
  - Basically two interaction types (oneway, request/response)
  - WSDL extended by WS-Policy, specifically WS-SecurityPolicy
- ebXML-based approach: **BPSS & CPPA**
  - Specification of business-independent interaction types (BPSS) and business-dependent configuration (CPPA)
  - Supported interaction types as specified in UMM, defining several properties required in typical B2B interactions
  - Predefined patterns exist, e.g., Commercial Transaction, Request/Response, Query/Response, Notification

Interactions (2)

Concepts Supported

	WSDL	ebXML
■ Functionality		
● name	X	X
● *pre- and postcondition*	–	X
■ Information: messages		
● requesting and responding documents	X	X
■ Behavior: predefined patterns		
● oneway	X	X
● request/response	X	X
○ *different successful responses*	–	X
○ different exceptional responses	X	X
● *configurable acknowledgements*	–	X
● *timing constraints*	–	X

X ... feature supported
– ... feature not supported

Interactions (3)

Concepts Supported

	WSDL	ebXML
Security: declarative properties		
message integrity, authenticity, confidentiality	~	X
authorization	–	X
non-repudiation of origin and receipt	–	X
Transaction: declarative properties		
atomicity	–	~
guaranteed delivery	–	X
Operation: configuration options		
transport protocol	X	X
synchronous and asynchronous binding	–	X
reliable messaging	–	X
security algorithms	X	X
message layout	–	X
Overall		
library of interaction types (patterns)	–	X

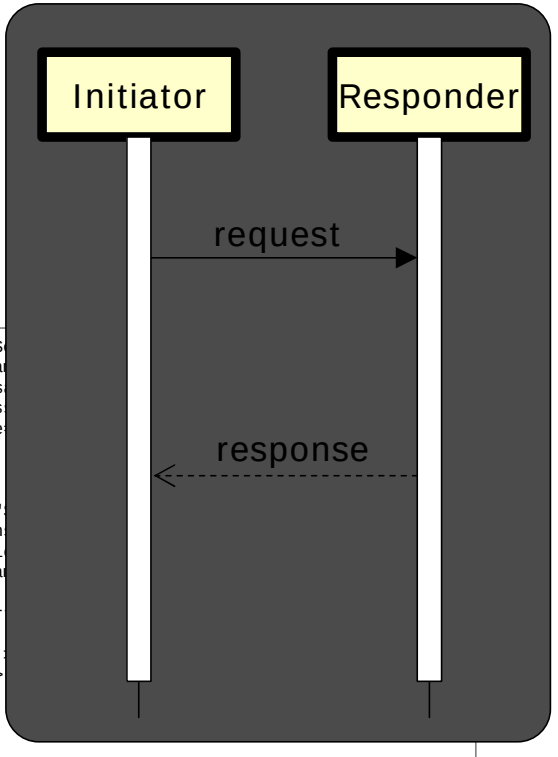
X ... feature supported
~ ... limited support
– ... feature not supported

Interactions (4)

Example: WSDL-based Approach

```
<wsdl:definitions ...>
  ...
  <wsdl:import namespace="urn:oasis:names:tc:ubl:Order:1.0:0.70"
               location="UBL_Library_0p70_Order__NS.xsd" />
  ...
  <wsdl:message name="PurchaseOrder"
                wsp:PolicyRefs="tns:IntegrityPolicy">
    <wsdl:part name="order"
               element="ubl:Order" />
  </wsdl:message>
  ...

  <wsp:UsingPolicy wsdl:Required="true"/>

  <wsp:Policy ... Name="IntegrityPolicy">
    <wssp:Integrity wsp:Usage="wsp:Required">
      <wssp:Algorithm Type="wssp:AlgCanonicalization"
          URI="http://www.w3.org/Signature/Drafts/xml-e
      <wssp:Algorithm Type="wssp:AlgSignature"
          URI="http://www.w3.org/2000/09/xmldsig#rsa-sh
      <wssp:SecurityToken>
        <wssp:TokenType>wssp:X509v3</wssp:TokenType>
      </wssp:SecurityToken>
      <MessageParts
          Dialect="http://schemas.xmlsoap.org/2002/12/w
        wsp:Body()</MessageParts>
    </wssp:Integrity>
  </wsp:Policy>
  ...
```

```
<wsdl:portType name="S
    <wsdl:operation na
      <wsdl:input mess
      <wsdl:output mes
      <wsdl:fault name
    </wsdl:operation>
    ...

  <wsdl:binding name="
    <soap:binding tran
                  styl
    <wsdl:operation na
      <wsdl:input>
        <soap:body par
      </wsdl:input>
      <wsdl:output ...
      <wsdl:fault ...>
    </wsdl:operation>
    ...

</wsdl:definitions>
```



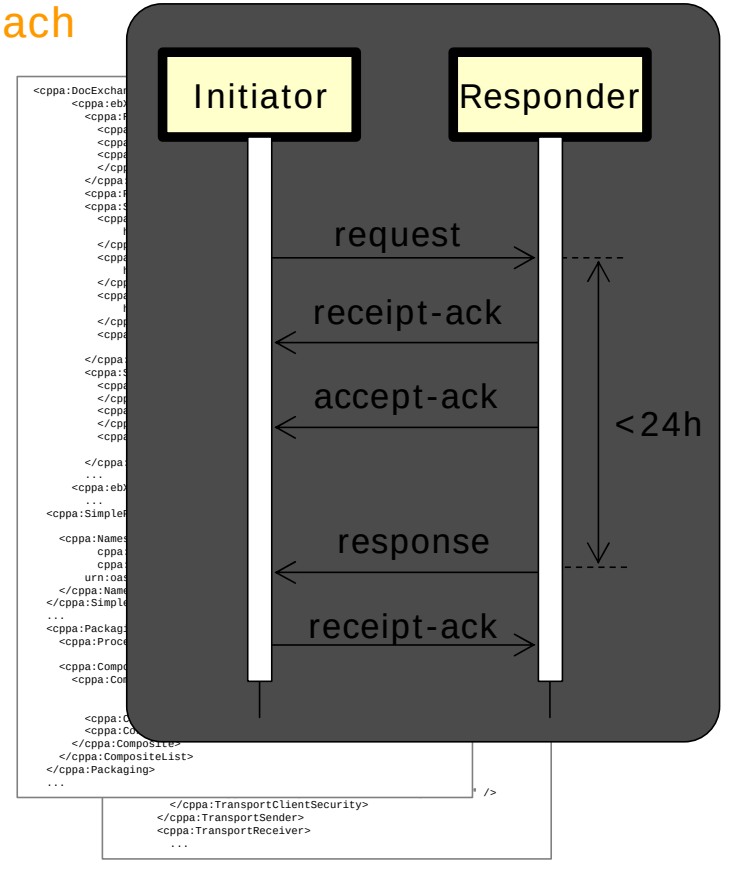
Specification of an *operation* using WSDL and WSSP

Interactions (5)

Example: ebXML-based Approach

```
<bpss:ProcessSpecification ...>
 <bpss:BusinessDocument name="PurchaseOrder"
        nameID="PurchaseOrder_BD"
        specificationLocation="UBL_Library_0p70_Order.xsd" />
        specificationElement="Order" >
    <bpss:ConditionExpression expressionLanguage="XPath"
          expression="/Order/OrderPricingCurrencyCode = 'USD'" />
    ...
  </bpss:BusinessDocument>
  ...
  <bpss:BusinessTransaction name="PurchaseOrderTransaction"
          nameID="PurchaseOrderTransaction_BT"
          pattern="http://ebxml.org/patt/CommercialTransaction"
          isGuaranteedDeliveryRequired="true">
    <bpss:RequestingBusinessActivity name="Offer"
            isAuthorizationRequired="true"
            isIntelligibleCheckRequired="true"
            isNonRepudiationReceiptRequired="true"
            isNonRepudiationRequired="true"
            timeToAcknowledgeAcceptance="PT6H"
            retryCount="3"
            timeToAcknowledgeReceipt="PT2H">
      <bpss:DocumentEnvelope businessDocument="PurchaseOrder"
              businessDocumentIDRef="PurchaseOrder_BD"
              isAuthenticated="true"
              isConfidential="false"
              isTamperProof="true" />
    </bpss:RequestingBusinessActivity>
    <bpss:RespondingBusinessActivity name="Accept"
            isAuthorizationRequired="true"
            isIntelligibleCheckRequired="true"
            isNonRepudiationRequired="true"
            timeToAcknowledgeReceipt="PT2H">
      <bpss:DocumentEnvelope
              businessDocument="PurchaseOrderAcceptance"
              isPositiveResponse="true"
              ... />
      <bpss:DocumentEnvelope
              businessDocument="PurchaseOrderDenial"
              isPositiveResponse="false"
              ... />
    </bpss:RespondingBusinessActivity>
  </bpss:BusinessTransaction>
  ...
```

Specification of a *business transaction* using BPSS



Specification of a *business transaction's* operational details using CPPA

Services (1)

Languages

- WSDL-based approach: **BPEL**
  - Specification of reactive/communicating processes (processes associated with one software component)
  - Based on WSDL interaction types
  - Both abstract and executable processes
- ebXML-based approach: **BPSS & CPPA**
  - Specification of business-independent service types (BPSS) and business-dependent configuration (profiles and agreements: CPPA)
  - Focus on binary relationships
  - Multi-party coordination as extension

Services (2)

Concepts Supported

	WSDL	ebXML
Functionality		
name	X	X
interactions	X	X
interaction interdependencies	X	~
nested services	–	X
Organization		
unilateral	X	–
bilateral	–	X
multilateral	–	~
dynamic	X	–
Information		
variables	X	–
data flow	X	–
user-defined message correlation	X	–

X ... feature supported
~ ... limited support
– ... feature not supported

Services (3)

Concepts Supported

	WSDL	ebXML
Behavior		
local state, local activities	X	–
bilateral synchronized state & activities	–	X
multilateral synchronized state & activities	–	–
timing	X	X
events	X	–
exception handling	X	~
compensation handling	X	~
Transaction: properties		
atomicity	–	~

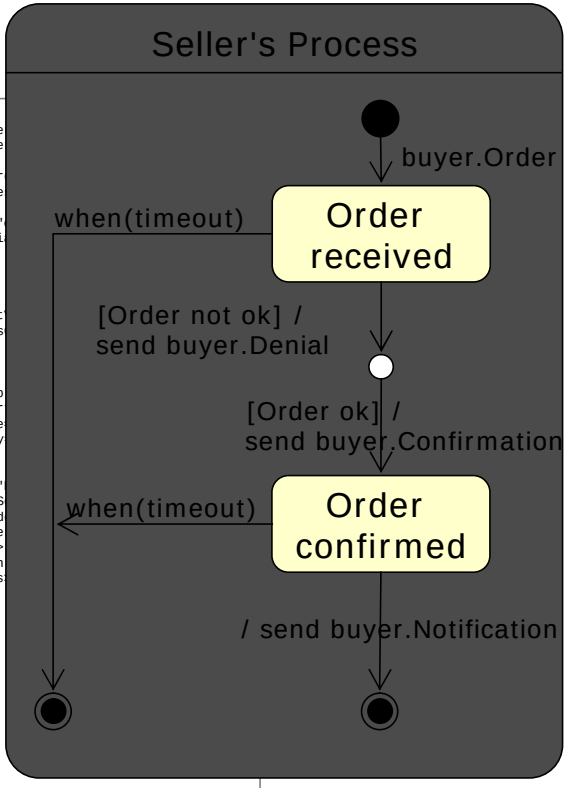
X ... feature supported
~ ... limited support
– ... feature not supported

Services (4)

Example: WSDL-based Approach

```
<bpel:process name="SellerOrderProcessing"
      targetNamespace="http://seller.example.com/schemas/order"
      abstractProcess="yes" ... ">

  <bpel:partners>
    <bpel:partner name="buyer"
             serviceLinkType="tns:BuyerSellerLink"
             myRole="seller"
             partnerRole="buyer"/>
  </bpel:partners>

  <bpel:variables>
    <bpel:variable name="BuyersOrder"
                   messageType="pt:PurchaseOrder" />
    <bpel:variable name="OrderAcknowledgement"
                   messageType="pt:PurchaseOrderAcceptance" />
    ...
  </bpel:variables>

  <bpel:correlationSets>
    <bpel:correlationSet name="orderCorrelation"
                         properties="pt:buyerOrderId"/>
  </bpel:correlationSets>

  <!-- default faultHandler and compensationHandler used -->

  <!-- sequence of order and notify shipment -->
  <bpel:sequence>

    <!-- scope of order interaction -->
    <bpel:scope>

      <bpel:eventHandlers>
        <!-- event handler providing timeout -->
        <bpel:onAlarm for="PT24H">
          <bpel:throw faultName="tns:Timeout" />
        </bpel:onAlarm>
        <!-- event handler providing interaction termination -->
        <bpel:onMessage partner="buyer"
                   portType="pt:Seller_PT"
                   operation="NotificationOfFailure"
                   variable="FailureNotification">
          <bpel:correlation set="orderCorrelation">
          <bpel:throw faultName="tns:BuyerNotificationOfFailure"/>
        </bpel:onMessage>
        ...
      </bpel:eventHandlers>
```

```
    <bpel:sequence>
      <bpel:receive partner="buye
                portType="pt:Selle
                operation="Order"
                variable="BuyersOr
                createInstance="ye
        <bpel:correlations>
          <bpel:correlation set="
                              initi
        </bpel:correlations>
      </bpel:receive>
      <bpel:switch>

        <bpel:case condition="get
                                'is
          <bpel:sequence>
            <bpel:assign>
              <bpel:copy>
                <bpel:from variab
                           proper
                <bpel:to variable
                         property
              </bpel:copy>
            </bpel:assign>
            <bpel:reply partner="
                  portType="pt:S
                  operation="Ord
                  variable="Orde
              <bpel:correlations>
                <bpel:correlation
              </bpel:correlations
            </bpel:reply>
          </bpel:sequence>
        </bpel:case>

        <bpel:otherwise>
          ...
        </bpel:otherwise>
      </bpel:switch>
    </bpel:sequence>
  </bpel:scope>

  <bpel:invoke partner="buyer"
          portType="ip:Buyer_PT"
          operation="Shipping"
          outputContainer="ShipmentNotice">
    <bpel:correlations>
      <bpel:correlation set="orderCorrelation" />
    </bpel:correlations>
  </bpel:invoke>
</bpel:sequence>

</bpel:process>
```

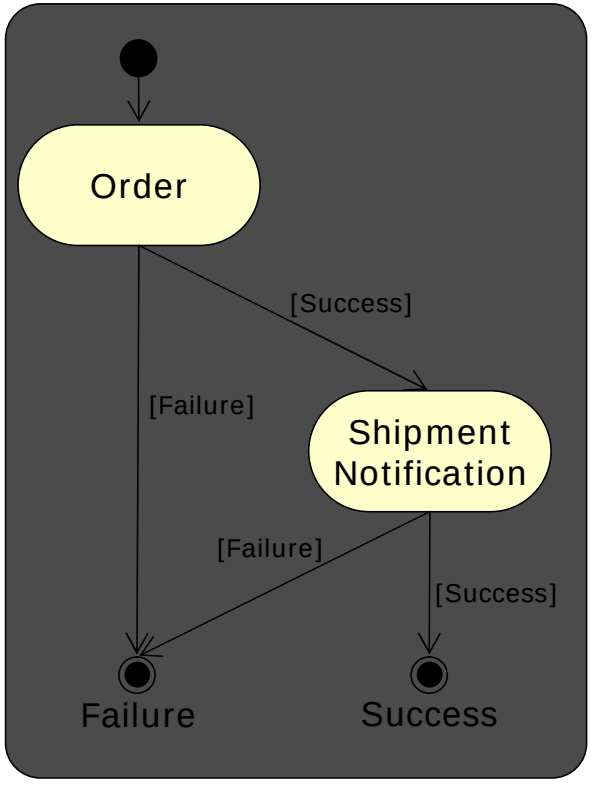


Specification of an *abstract process* using BPEL

Services (5)

Example: ebXML-based Approach

```
<bpss:ProcessSpecification name="test"
      uuid="urn:icann:buyer.com:bpid:test$1.0" version="1.0" ...>
  ...

  <bpss:BinaryCollaboration name="DropOrder"
                           initiatingRole="BuyerId">
    <bpss:Role name="Buyer" nameID="BuyerId"/>
    <bpss:Role name="Seller" nameID="SellerId"/>
    <bpss:Start toBusinessState="Order"
                toBusinessStateIDRef="Order_BTA"/>
    <bpss:BusinessTransactionActivity name="Order"
          fromRole="Buyer"
          toRole="Seller"
          businessTransaction="PurchaseOrderTransaction"
          timeToPerform="PT24H"/>
    <bpss:BusinessTransactionActivity name="ShipmentNotice"
          fromRole="Seller"
          toRole="Buyer"
          businessTransaction="AdvanceShipmentNotice"
          timeToPerform="PT24H"/>
    <bpss:Success fromBusinessState="ShipmentNotice"
                  conditionGuard="Success"/>
    <bpss:Failure fromBusinessState="Order"
                  conditionGuard="Failure"/>
    <bpss:Failure fromBusinessState="ShipmentNotice"
                  conditionGuard="Failure"/>
    <bpss:Transition fromBusinessState="Order"
                     toBusinessState="ShipmentNotice"
                     conditionGuard="Success">
    </bpss:Transition>
  </bpss:BinaryCollaboration>

</bpss:ProcessSpecification>
```



Specification of a *binary collaboration* using BPSS

Summary

- Information items, documents: **interoperable**
  - CCTS and XML Schema can be used in combination (e.g., UBL)
  - CCTS introduces overhead but facilitates interoperability
- Interactions: **general purpose vs. domain specific**
  - WSDL → seamless integration with programming languages
  - ebXML → declarative specification of typical B2B interaction characteristics (receipt acknowledgements, non-repudiation, timing constraints, atomicity)
  - WSDL interactions $\subseteq$ ebXML interactions
- Services: **expressive power vs. ease of use**
  - WSDL → supports complex behavior, message correlation; consistency of interacting processes?
  - ebXML → ease of use for binary relationships (cf. WSCL); based on interaction characteristics (timing, atomicity!)
  - Different levels of abstraction (event-based vs. activity-based)

Conclusion

- Framework proved useful for classifying concepts
  - A more elaborate reference model is needed to derive concrete requirements
- WSDL and ebXML are not interoperable
  - Conceptual and operational differences
  - Difference in level of abstraction: ebXML concepts could be translated to a WSDL-based implementation while loosing operational ebXML compatibility
- If your interactions fit the ebXML interaction types, then the ebXML-based approach is certainly favourable
  - (leaving aside product and market considerations)

Future Work

- Framework
  - Complete with eCo business layer, market layer?
- Model Driven Architecture approach
  - Platform Specific Models for target technologies (Web Services, ebXML)
  - Conceptual, platform-independent modeling of B2B protocols → a UML profile, based on
    - the supported concepts identified (bottom-up approach)
    - existing UML profiles (UN/CEFACT Modeling Methodology, OMG EDOC)
    - UML modeling concepts (UML 2.0)
    - Research on contract specification and verification
  - Transformations, Tools…

Questions?

- kramler@big.tuwien.ac.at
- http://www.big.tuwien.ac.at/