Models and Languages for Describing and Discovering E-Services

Fabio Casati and Ming-Chien Shan
Hewlett-Packard

Semantic Web Working Symposium
Tutorial
Stanford, CA, USA July 2001
E-Services

• Applications accessible electronically by humans and/or other applications

Service Provider A

Service Provider B

Service Provider C

Customers

Truck rental
Shipping
Moving
Traffic report
Route planning
Customized news
...

...
E-Services, Web Services

- Similar, but “Web Service” puts emphasis on Web technologies.
- Application accessible using standard Internet protocols.

<table>
<thead>
<tr>
<th>Service Provider A</th>
<th>Traffic report</th>
<th>Service Provider B</th>
<th>Route planning</th>
<th>Service Provider C</th>
<th>Customized news</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Typically XML over HTTP/s

Customers
Discovering and Invoking E-Services

Customer needs an e-publishing service

1: Queries an e-service directory (broker) searching for services of interest.
   Gets (ranked) list of e-services

2: Access service detail, Negotiate service quality, Invoke the e-service

E-service directory or broker

Customer (or apps on their behalf)

E-publishing service A

E-publishing service B

E-publishing service C
Dynamic Composition

Start Node

Check credit

Check passed?

yes

Abort

And Join

Wheel delivery

Credit card

Complete

Restaurant selection

Service Provider A

Credit Check

Service Provider B

Route planning

Service Provider C

Credit card payment
The Evolution of the Internet

• Internet Chapter I
  • Web used to deliver information and perform e-commerce transactions
  • Web applications targeted to human users, manual interaction

• Internet Chapter II
  • A variety of e-services available on the Internet
  • Users, apps can automatically discover the available e-services that best meet their needs at any given time.
  • Service quality negotiation and invocation are also automated.
The Evolution of the Internet (cont.)

Chapter 1  \rightarrow  Chapter 2
At your desk \rightarrow  Living your life
PC only \rightarrow  PC+ devices + anything
Web storefronts \rightarrow  Automated e-services
Do-it-yourself \rightarrow  Do-it-for-me
“Equal Opportunity”

Low cost entry barrier to service advertisement and delivery. Small shops compete with large enterprise.

For customers: more selection, better services, lower prices
How to Get There

• Many issues to be addressed
  • E-service description
  • E-Service advertisement, discovery, and selection
  • E-Service Composition
  • Secure access, delivery
  • E-service development tools
  • E-Service measuring, monitoring, management
  • Middleware infrastructure for e-services
  • …

• In this tutorial we focus on two fundamental problems: e-service description and advertisement/discovery
E-Service Description

• Users dynamically find services on the Web, offered by different providers
• Before accessing the services, they need to know information such as what exact service is offered, at what conditions, how to invoke the service, etc...

→ The different characteristics of an e-service must be described so that users know what the service offers and how to use the service

E-publishing service
Characteristics of an E-Service

- Type of service offered
- Interface
  - Operations
  - Bindings
- Interaction (Conversations)
- Transaction
- Properties, constraints
Service Type

• Detailed description of what service is being offered, at what conditions
  • E.g., sell used SUV cars of brand X and Y, trade-ins welcome
  • But: it has to be machine-readable
• Back to the ontology problem...
  • Standard ontology? Slow, inflexible, rarely fits needs
  • Provider-specific ontology? How can users understand it?
Service Type - Vocabularies

- Individual providers and standard bodies can define vocabularies
- Providers can then describe services using one or more predefined vocabularies
- Vocabularies could themselves be e-services
  - Can be discovered, accessed with same mechanisms

<Description ServiceName="xx" Vocabulary="Printing">  
  <ProviderName> Printy </ProviderName>  
  <ShippingArea> Sweden </ShippingArea>  
  ......  

Interfaces

• Purpose similar to IDL descriptions, but in the context of e-services
  • Typically described by an XML document
  • Interface defined in terms of which XML documents the service needs and which XML document (if any) is sent back to the user

• Address

• Binding (HTTP, MIME, ...)

Printy
Example

```xml
<Interaction IntType="DocExchange" id="PrintShip">
  <Input>
    <InputDoc id="PrintShipRequest"
      IDSchema="http://acme.org/in-xyz.xsd">
    </InputDoc>
  </Input>
  <Output>
    <OutputDoc id="Invoice"
      ODSchema="http://acme.org/in-xyz.xsd">
    </OutputDoc>
  </Output>
</Interaction>
```
Conversations

• Set of interactions between a user and a service
• A Conversation Definition Language (CDL) specifies rules and constraints about a conversation
  • Such as the allowed order of operation invocation

1. Upload documents
2. Browse Templates
3. Prepare Flyer
4. Print & Ship

Printy
Conversation (cont.)

- Similar to a state machine specification, and to a RosettaNet PIP (NOT a workflow)
- Independent from interface
Transactions

- E-Services may expose transactional properties
  - ACID semantic revisited in e-services context
- "Atomicity", "two phase commit", and even a post-commit "rollback" (compensation)
  - **Semantic, ACID-like properties may be a negotiable quality parameter**
  - May be time-based, cost-based, ...
Web Services Description Language (WSDL)

- An XML language for describing e-service interfaces
- Originally proposed in September 2000 by Ariba, IBM, and Microsoft
- Version 1.1 submitted to W3C
  - As a note for the W3C XML Activity on XML Protocols

- URL: http://www.w3.org/TR/wsdl
WSDL

- WSDL describes the operations provided by e-services in an abstract way, as XML document exchanges
- Allows the description of bindings (e.g., SOAP or HTTP)
- Modular
- No built-in support for classification, conversations, transactions
WSDL Elements

• **Types**
• **Messages**: abstract definition of data being exchanged
• **Operation**: abstract description of a method or function
• **Port Type**: a set of (abstract) operations

```xml
<types>
  <schema [...]> 
    <element name="PrintRequest"> [...]</element> 
    <element name=".."> [...]</element> </schema>
</types>

<message name="PrintInput"> <part name="body" element="nsp1:PrintRequest"/></message>
<message name="PrintOutput"> [...] </message>

<portType name="PrintPortType"> 
  <operation name="Print"> 
    <input message="tns:PrintInput"/> 
    <output message="..."/></operation>
</portType>
```
WSDL Elements (cont.)

- **Binding**: a **concrete** protocol for a port type
- **Port**: an address for a binding
- **Service**: a collection of ports

```xml
<binding name="PrintSoapBinding" [...]>
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="Print"> [...] </operation>
</binding>

<service name="PrintService">
  <documentation>Great print service</documentation>
  <port name="PrintPort" binding="tns:PrintBinding">
    <soap:address location="http://..."/></port>
</service>
```
E-Service Discovery

• The beauty of the e-services vision is the ability to find the currently available service that best fits my need.
  • How can I know what services are available? How can I find which services fit my needs?
  • How can I know where to get detailed information on how to use a service?
Directories

• Would be useful to have **machine-readable service directories**. Issues:
  • What to describe, how to describe it, who describes it?
  • Structure of the directory and classification of services and service providers
  • Interface
Directory Content

• Different levels of details
  • List of companies and services, plus contact information
• Categories
• Detailed information on how to access a service (URL, interface, bindings, etc)
Structure and Classification

- Entries in a directory may be divided in categories
  - Categories and subcategories very useful for searching the directory
- Issues:
  - Who defines and manages the categories? Controlled or open model?
  - Who puts services and businesses into the appropriate categories?
- **Difficult to agree on predefined categories, taxonomies**
Directory API and Access Control

• Query API
  • Find by types, properties, context, interfaces
  • Open vs restricted access

• Publish API
  • Push vs pull
  • Who can publish?
  • What can users publish? How much data?
  • Validate information?
Universal Description, Discovery, and Integration (UDDI)

- Industry consortium, includes major tech companies
- Defines a way to publish and discover information about e-services
  - Format, API, and framework for e-service directories

- Currently 15+ peer members in the committee

- V1: September 2000
- V2: June 2001

- URL: http://uddi.org
Business Registry

- Repository of information about services and service providers
  - UDDI defines the format of the information in the registry and the API to access it
  - Rules for operators implementing a UDDI registry within UDDI net

Companies register information about them and their services

Customers query the registry to find the services they need and learn how to interact with them.
Business Registration

An XML file that describes a business entity and its e-services

- White pages include contact information (e.g., address, phone number, identifiers etc)
- Yellow pages describe businesses and services according to several taxonomies
- Green pages provides detailed technical information about the service
• <businessEntity>
  • Contains information about a company (or division) and the services it offers
    - key
    - authorizedName (publisher of the information)
    - name*
    - Description*
    - Operator
    - Contacts (phone, email, address, person name,..)
    - IdentifierBag (e.g. DUNS)
    - CategoryBag (taxonomy)
    - businessServices: List of services
Business Registration: Services

- **<businessService>**
  - Contains business information about a service.
    - `serviceKey`
    - `businessKey`
    - `name`*
    - `description`*
    - `CategoryBag` (taxonomy)
    - `bindingTemplates`: detailed technical information
<businessService businessKey="..." serviceKey="...">
   <name>ePrintService</name>
   <description xml:lang="en">Compose, print, and ship brochures</description>
   <bindingTemplates>
      <bindingTemplate> […] </bindingTemplate>
      <bindingTemplate> […] </bindingTemplate>
      <bindingTemplate> […] </bindingTemplate>
   </bindingTemplates>
</businessService>
Business Registration: Technical Service Description

<bindingTemplate>

• Technical information about a service.
  - bindingKey
  - serviceKey
  - accessPoint: Attribute-qualified string (Mail, http/s, telephone, fax,..)
  - hostingRedirector (refers to another bindingKey)
  - tModelInstanceDetails
    - How to interact with the service at the specified address
    - Can be empty
bindingTemplate Example

<bindingTemplate bindingKey=".." serviceKey="..">
    <accessPoint urlType="http"> http://... </accessPoint>
    <tModelInstanceDetails>
        <tModelInstanceInfo tModelKey="...">
        </tModelInstanceInfo>
    </tModelInstanceDetails>
</bindingTemplate>
Business Registration: Technical Service Description (details)

```xml
<businessEntity>
  🔄 <businessService>
  🔄 <bindingTemplate>
  🔄 <tModelInstanceInfo>
</businessEntity>

<tModelInstanceInfo>
  • Detailed technical information
    - <tModelKey> defines technical fingerprint (reference)
    - description*
    - instanceDetails
      - E.g., parameter settings, default values
```
Technical Model (tModel)

- Information on how to interact with a service
  - NOT a service description language
  - From the tModelKey, users know the compliance with a specification (Hence: how to interact)
  - E.g., can refer to a Rosettanet PIP, a WSDL service interface

- Independent from specific implementations.

- Very simple structure:
  - Key, operator, authorizedName
  - Name
  - description*
  - overviewDoc
  - indetifierBag, categoryBag identification and taxonomy information
tModel Example

<tModel authorizedName="..." operator="..." tModelKey="...">
   <name>ePrint Service</name>
   <description xml:lang="en">
      WSDL description of a print service interface
   </description>
   <overviewDoc>
      <description xml:lang="en">WSDL service description</description>
      <overviewURL>http://...</overviewURL>
   </overviewDoc>
   <categoryBag>
      <keyedReference tModelKey="..."
         keyName="uddi-org:types"
         keyValue="wsdlSpec"/>
   </categoryBag>
</tModel>
Managing Large Enterprises

- `<publisherAssertion>` defines relationships between two `businessEntity`
  - Can be used to define divisions that belong to a company

- `<fromKey>`
- `<toKey>`
- `<keyedReference>` defines type of relationship
  - A name,value pair within a `tModel`
UDDI Structure (summary)

<businessEntity>
   <businessService>
      <bindingTemplate>
         <tModelInstanceInfo>
            Loc=Dolomiti
         </tModelInstanceInfo>
      </bindingTemplate>
   </businessService>
</businessEntity>
Identifiers

• Can annotate data with identifiers
• Within `<businessEntity>` and `<tModel>`
  • `tModelKey`, `keyName`, `keyValue`
• Can use many types of identifiers (SSNs, DUNS, ..)
• UDDI defines `tModels` for DUNS and Thomas Register
Categories

• Businesses, services, and tModels can specify the “category” to which they belong
  • Travel agency, Xtreme Travels, ...
• Given by service provider
• Information defined as keyed references
  • tModelKey, keyName, keyValue

• Predefined taxonomies (tModels): NAICS (industry codes), UNSPSC (products and services), ISO 3166 (geography)
UDDI Structure (summary)

- `<businessEntity>`
- `<businessService>`
- `<bindingTemplate>`
- `<tModelInstanceInfo>`
- `<tModel>`

Identifiers, categories

Loc=Dolomiti
Publishing a WSDL Service Interface in UDDI

• I want to publish a service interface in WSDL
• Register WSDL description as tModels
  • Classified as “wsdlSpec” with uddi-org:types taxonomy
  • OverviewDoc will point to WSDL document
  • If WSDL description is scattered across many documents, then register several tModels
Publishing a WSDL service in UDDI

1. Retrieve the tModel of interest
2. Read the overviewDoc
3. Generate the implementation
4. Publish a new businessService

A tool can automate this process
UDDI API

- **Inquiry and Publisher API**
  - **Inquiry**: search/browse information
    - Open, no security, access control
  - **Publisher**: publish and manage information
    - Requires registration with operator, service level agreements
    - Secure and controlled interaction (security and access control features are operator-specific)

---

**Publisher API**: Companies register information about them and their services

**Inquiry API**: Customers query registry to find services they need and learn how to interact.
Inquiry API

- **Find_xx**: overview of registration data.
- **Get_xx**: detailed info about businessEntity, businessService, bindingTemplate.

```xml
<find_binding serviceKey="uuid_key" [maxRows="nn"] generic="2.0" xmlns="urn:uddi-org:api_v2" >
  [<findQualifiers/>]
  <tModelBag/>
</find_binding>
```
Publisher API

- Save_xx: add/update entries in the registry
- Delete_xx: delete entries
- Get_xx: Info about docs registered by a company

```xml
<delete_binding generic="2.0" xmlns="urn:uddi-org:api_v2">
  <authInfo/>
  <bindingKey/> [ <bindingKey/> ... ]
</delete_binding>
```
UDDI Operators

• Implement publish and inquiry API
  • Control access to information. They are “custodian” of information published at their site.
  • Validate documents (also trims spaces, fields; checks taxonomies, UUID references)
  • Assign UUIDs
• Replicate information with other operators
  • Global replication every 12 hours
Tier 1 and Tier 2 Operator Accounts

- Tier 1 may only create a limited number of entities
  - businessEntity: 1
  - businessService per entity: 4
  - BindingTemplates per service: 2
  - tModel: 100
  - Relationship: 10
# E-Service Interoperability Stack

<table>
<thead>
<tr>
<th>Interop Stack</th>
<th>Universal Service Interop Protocols (these layers are not defined yet)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Universal Description, Discovery Integration (UDDI)</td>
</tr>
<tr>
<td></td>
<td>Simple Object Access Protocol (SOAP)</td>
</tr>
<tr>
<td></td>
<td>Extensible Markup Language (XML)</td>
</tr>
<tr>
<td></td>
<td>Common Internet Protocols (HTTP, TCP/IP)</td>
</tr>
</tbody>
</table>

Source: UDDI Technical White Paper - September 2000
Simplicity

- **No WSDL-like service description language**
  - It is complementary to UDDI
- **Open identification and categorization model**
  - Classification defined by service provider, with limited validation
- **UDDI definitions and interfaces are simple**
  - Advanced discovery mechanisms to be provided by portals, marketplaces
ebXML - Introduction

- electronic business XML (ebXML) is an international initiative established (in 11/1999) by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the Advancement of Structured Information Standards (OASIS).

- The ebXML vision is to deliver “A single set of internationally agreed upon technical specifications that consist of common XML semantics and related document structures to facilitate global trade”

- It is targeted at every sector of the business community, from international conglomerate to small sized enterprise engaged in B2B and B2C trade.

- Currently, 7 project teams are chartered by its steering committee:
  - Business process
  - Technical architecture
  - Core components
  - Transport/routing and packaging
  - Registry and repository
  - Technical coordination and support
  - Marketing, awareness and education
ebXML - Business Requirements

• A single, simple, consistent approach to using XML for electronic business processes in both the B2B and B2C environments.

• Support for both vertical and horizontal solutions regardless of the sophistication of the user.

• Support for a range of implementations from basic, low cost solutions for SME deployment, to comprehensive, complex implementation for large enterprises.

• Fully interoperable transport, routing, and packaging solutions.

• Security solutions that meet business confidentiality requirements.

• An open development process with no barriers to entry.
ebXML Operation Overview

1. Request ebXML specification
2. Build System
3. Send ebXML specification
4. Register company profile
5. Send Company X’s Scenario
6. Query about Company X’s Profile
7. Submit CPA
8. CPA Accepted
9. Request Company X’s Scenario
10. Send Company X’s Scenario
11. DO BUSINESS!

Company X

Company Y

ebXML Repository

- Specification
- Profiles
- Scenarios

ebXML BO Library

ebXML BP Model
The ebXML architecture will provide:

- A way to define business processes and their associated message and content.

- A way to register and discover business process sequences with related message exchanges.

- A way to define company profiles.

- A way to define trading partner agreements.

- A uniform message transport layer.
ebXML Operational Environment

Business Process and Information Models

UML to XML conversion

Registration

Repositories

Retrieval of new or updated ebXML Models

Retrieval of new or updated ebXML Models

Retrieval of ebXML Specifications & Models

Internal Business App

Build

Implementers

Build

Shrink-wrap Application

Business Message

BSI Interface

ebXML Transport

CPA

BSI Interface

Business Message
ebXML - Registry and Repository

- A **registry** is a mechanism where business documents and relevant metadata can be registered such that a pointer to their location, and their metadata, can be retrieved as the result of a query.

- A **repository** is a location (or a set of distributed locations) where documents pointed at by the register reside and from which they can be retrieved by conventional means (e.g., http/ftp).

- A registry can be established by an industry group or standards organization.
ebXML – Registry Interface
### Method Summary

- **void addSlotsAccepted(RequestAcceptedResponse resp)**
  - Notifies client that a previously submitted AddSlotsRequest was accepted by the Registry.

- **void addSlotsError(sbXMLError error)**
  - Notifies client that a previously submitted AddSlotsRequest was not accepted by the Registry due to an error.

- **void approveObjectsAccepted(RequestAcceptedResponse resp)**
  - Notifies client that a previously submitted ApproveObjectsRequest was accepted by the Registry.

- **void approveObjectsError(sbXMLError error)**
  - Notifies client that a previously submitted ApproveObjectsRequest was not accepted by the Registry due to an error.

- **void deprecateObjectsAccepted(RequestAcceptedResponse resp)**
  - Notifies client that a previously submitted DeprecateObjectsRequest was accepted by the Registry.

- **void deprecateObjectsError(sbXMLError error)**
  - Notifies client that a previously submitted DeprecateObjectsRequest was not accepted by the Registry due to an error.

- **void removeObjectAccepted(RequestAcceptedResponse resp)**
  - Notifies client that a previously submitted RemoveObjectsRequest was accepted by the Registry.

- **void removeSlotsAccepted(RequestAcceptedResponse resp)**
  - Notifies client that a previously submitted RemoveSlotsRequest was accepted by the Registry.

- **void removeObjectsError(sbXMLError error)**
  - Notifies client that a previously submitted RemoveObjectsRequest was not accepted by the Registry due to an error.

- **void removeSlotsError(sbXMLError error)**
  - Notifies client that a previously submitted RemoveSlotsRequest was not accepted by the Registry due to an error.
Describe the required behavior of the underlying messaging system to:

- Realize reliable secure sending and receiving of messages over any network capable of carrying XML.
- Detail the format and structure of the wrapper, header, and any other data within the message – to include signatures and encryption.
- Query ebXML server for the services they support.
Describe the required support at both a session layer or be applied to a single, stand-alone document instance:

- Confidentiality - Only sender and receiver can interpret document contents.

- Authentication of sender/receiver - Assurance of the sender’s or receiver’s identity.

- Integrity - Assurance that the message contents have not been altered.

- Non-repudiation of origin/receipt - The sender/receiver can not deny having sent/receive the message.
A Complete Example
Business Process Analysis

- ebXML business process are defined by the information specified in the ebXML UMM e-Business Process Metamodel.

- The Metamodel specifies all the information that needs to be captured during the analysis of an electronic commerce based business process within the ebXML framework.

* UMM - UN/CEFACT Modeling Methodology N9.0
Business Process Identification/Discovery
(Step 1)

**Business reference Model**
Name: Direct to customer drop ship retail model
**Business Areas:** Direct To Customer Retail
Finance

**Business Area**
Name: Direct To Customer Retail
Boundary: Customer, Retailer, Transport Carrier,
Direct Supply Retail Vendor (DSVendor),
Credit Authority
Process Area: Customer Order Management,
Customer Order Fulfillment,
Vendor Inventory Management,
Product Catalog Exchange

**Business Area**
Name: Finance
Boundary: Retailer, DSVendor
Process Area: Payment
Business Process Identification/Discovery (Step 1)

Process Area
Name: Customer Order Management
Business Processes: Firm Sales Order,
   Customer Credit Inquiry,
   Customer Credit Payment

Process Area
Name: Customer Order Fulfillment
Business Processes: Purchase Order Management
   Ship Goods
Business Process
Name: Firm Sales Order
Actors: Customer, Retailer
Precondition:
Postcondition:
Exceptions:

Business Process
Name: Customer Credit Inquiry
Actors: Retailer, Credit Authority
Precondition:
Postcondition:
Exceptions:
Business Collaboration

Id: bcid.ean.12:CreateOrder$1.0
Partner Types: Customer, Retailer
Authorized Roles:

<table>
<thead>
<tr>
<th>Form Id</th>
<th>BCPT-7.1-Create-Customer-Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>bcid.ean.1234567890128:CreateCustomerOrder$1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>From Business Activity (Transaction)</th>
<th>Initiating Partner Type</th>
<th>Business Activity</th>
<th>Responding/Receiving Partner Type</th>
<th>Transition Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>Customer</td>
<td>Create Order</td>
<td>Retailer</td>
<td>NONE</td>
</tr>
<tr>
<td>Create Order</td>
<td>NOT-APPLICABLE</td>
<td>SUCCESS</td>
<td>Customer</td>
<td>BusinessTransaction(&quot;FirmCustomerSalesOrder&quot;), State=END</td>
</tr>
<tr>
<td>Create Order</td>
<td>NOT-APPLICABLE</td>
<td>FAILURE</td>
<td>Customer</td>
<td>BusinessTransaction(&quot;FirmCustomerSalesOrder&quot;), State=CONTROL-FAILED</td>
</tr>
</tbody>
</table>
Business Transaction & Authorized Roles (Step 4)

Business Transaction

Id:
Business activities & associated roles:
Requesting Partner Type:
Requesting Activity Role:
Requesting Activity Document:
ebXML Business Processes
A business object can be composed of re-usable Core Components.
Business Process Implementation

• The UMM metamodel is a description of business semantics that allows trading partners to capture the details of a specific business scenario using a consistent modeling methodology.

• However, it may contain more information than what is required for configuring ebXML compliant software.

• The ebXML Business Process Specification Schema is a subset of UMM metamodel. Using it, the user may thus create a business process specification that contains only the information required to configure ebXML compliant software.

• The ebXML Business Process Specification Schema is available in two stand-alone representations, a UML version, and an XML version.
A Business transaction with one request and two possible responses
Binary Collaboration

<BinaryCollaboration name="Product Fulfillment" timeToPerform="P5D">
  <Documentation>
    timeToPerform = Period: 5 days from start of transaction
  </Documentation>
  <AuthorizedRole name="buyer"/>
  <AuthorizedRole name="seller"/>
  <BusinessTransactionActivity name="Create Order"
    businessTransaction="Create Order"
    fromAuthorizedRole="buyer"
    toAuthorizedRole="seller"
    isLegallyBinding="true"/>
  <BusinessTransactionActivity name="Notify shipment"
    businessTransaction="Notify of advance shipment"
    fromAuthorizedRole="buyer"
    toAuthorizedRole="seller"/>
</BinaryCollaboration>
<BinaryCollaboration name="Product Fulfillment" timeToPerform="P5D">  
    <Documentation>
        timeToPerform = Period: 5 days from start of transaction
    </Documentation>
    <AuthorizedRole name="buyer"/>
    <AuthorizedRole name="seller"/>
    <BusinessTransactionActivity name="Create Order">
        businessTransaction="Create Order"
        fromAuthorizedRole="buyer"
        toAuthorizedRole="seller"/
    </BusinessTransactionActivity>
    <BusinessTransactionActivity name="Notify shipment">
        businessTransaction="Notify of advance shipment"
        fromAuthorizedRole="buyer"
        toAuthorizedRole="seller"/
    </BusinessTransactionActivity>

    <Start toBusinessState="Create Order"/>

    <Transition
        fromBusinessState="Create Order"
        toBusinessState="Notify shipment"/>

    <Success fromBusinessState="Notify shipment"
        guardCondition="Success"/>

    <Failure fromBusinessState="Notify shipment"
        guardCondition="BusinessFailure"/>
</BinaryCollaboration>
ebXML Business Process Specification Schema

Overall Structure

```plaintext
ProcessSpecification (Documentation*, (Include* | DocumentSpecification* |

Documentation ()
Include ( Documentation* )
DocumentSpecification( Documentation*, BusinessDocument* )

BusinessDocument( Documentation* )
  Package( Documentation*, (Package | BinaryCollaboration | BusinessTransaction | MultiPartyCollaboration)* )

BinaryCollaboration( Documentation*, AuthorizedRole, AuthorizedRole, (Documentation* | Start | Transition | Success | Failure | BusinessTransactionActivity | CollaborationActivity | Fork | Join)* )

AuthorizedRole( Documentation* )
Start( Documentation* )
Transition( Documentation* )
Success( Documentation* )
Failure( Documentation* )
Fork( Documentation* )
Join( Documentation* )

BusinessTransactionActivity( Documentation* )
CollaborationActivity( Documentation* )

BusinessTransaction( Documentation*, RequestingBusinessActivity, RespondingBusinessActivity)
  RequestingBusinessActivity( Documentation*, DocumentEnvelope )
  RespondingBusinessActivity( Documentation*, DocumentEnvelope* )

MultiPartyCollaboration( Documentation*, BusinessPartnerRole* )
  BusinessPartnerRole( Documentation*, Performs*, Transition* )
    Performs( Documentation* )
    Transition( Documentation* )
```
**Collaboration Protocol Profile (CPP)** describes party’s IT capabilities:

- Communication protocols
- Security requirements
- Business process it supports

**Collaboration Protocol Agreement (CPA)** describes:

- Agreed IT capabilities
- Business process to be performed

CPA is intersection of two parties’ CPPs plus results of negotiating variable parameters.

**ebXML CPP & CPA**
ebXML CPP & CPA

What Parties WILL do

CPA

intersection, negotiation

Party 1
CPP

Party 2
CPP

What Parties CAN do
CPP Structure

<CollaborationProtocolProfile id = "id"
   various namespace attributes...>
   <Party partyId = "N01"/>
   ...(Refer next slide)
</Party>
<!--CollaborationProtocol: one or more-->
<CollaborationProtocol version = "1.0" id = "N07"
   xlink:type = "locator"
   xlink:href = "http://www.ebxml.org/services/purchasing.xml">
   Buy and Sell
</CollaborationProtocol>
<ds:Signature>any combination of text and elements
</ds:Signature>
</CollaborationProtocolProfile>
Some Open Research Issues

- Flexible workflows/Service Composition
- B2B operation data mining and analysis
- QoS description and negotiation
- Monitoring and management
- Personalization (profile, context, catalog management)
- XML and heterogeneous databases
- Contracts, non-repudiation

E-Services
Contacts

• Fabio Casati
  Casati@hpl.hp.com
  www.hpl.hp.com/personal/Fabio_Casati

• Ming-Chien Shan
  Shan@hpl.hp.com