An ebXML Infrastructure Implementation through UDDI Registries and RosettaNet PIPs

Asuman Dogac, Yusuf Tambag, Pinar Pembecioglu, Sait Pektas,
Gokce Laleci, Gokhan Kurt, Serkan Toprak, Yildiray Kabak

Software Research and Development Center
Middle East Technical University (METU)
06531 Ankara Turkiye
email: asuman@srdc.metu.edu.tr

Abstract

Today's Internet based businesses need a level of interoperability which will allow trading partners to seamlessly and dynamically come together and do business without ad hoc and proprietary integrations. Such a level of interoperability involves being able to find potential business partners, discovering their services and business processes, and conducting business "on the fly". This process of dynamic interoperation is only possible through standard B2B frameworks. Indeed a number of B2B electronic commerce standard frameworks have emerged recently. Although most of these standards are overlapping and competing, each with its own strenghts and weeknesses, a closer investigation reveals that they can be used in a manner to complement one another. In this paper we describe such an implementation where an ebXML infrastructure is developed by exploiting the Universal Description, Discovery and Integration (UDDI) registries and RosettaNet Partner Interface Processes (PIPs).

ebXML is an ambitious effort and produced detailed specifications of an infrastructure both for B2B and B2C. However a public ebXML compliant registry/repository mechanism is not available yet. On the other hand, UDDI's approach to developing a registry has been a lot simpler and public registries are available. In ebXML, trading parties collaborate by agreeing on the same business process with complementary roles. Therefore there is a need for standardized business processes. In this respect, exploiting the already developed expertise through RosettaNet PIPs becomes indispensible. We show how to create and use ebXML "Binary Collaborations" based on RosettaNet PIPs and provide a GUI tool to allow users to graphically build their ebXML business processes by combining RosettaNet PIPs. In ebXML, trading parties reveal essential information about themselves through Collaboration Protocol Profiles (CPPs). To conduct business, an agreement between parties is necessary and this is expressed through a Collaboration

Protocol Agreement (CPA). To help with this process, a tool is implemented to automate the process of configuring a Collaboration Protocol Agreement (CPA) from given Collaboration Protocol Profiles (CPPs).

Being ebXML compliant mandates the messages exchanged to conform to ebXML messaging architecture. Furthermore since ebXML business processes may include several "Binary Collaborations", there is a need for a workflow enactment service to keep track of the execution. As a part of the infrastructure developed, we provide a server implementation which handles ebXML messages as well as keeping track of the data and control flow in the ebXML business processes.

1 Introduction

As businesses move more and more to the Web, it becomes increasingly important to improve the mechanisms of doing business over the Web. This basically involves developing standard ways of discovering the potential partners and automating the business processes among them.

Electronic business processes include both interactions between trading partners (public processes) and the private processes within a company (private processes). A public business process involves predefined message and document formats exchanged between the partners, the sequence of message exchange as well as some other mechanisms like the transport and security protocol.

A number of B2B standards have emerged [10] supporting the discovery and automation of public business processes over the Web. The basic functionalities of some of these standards are as follows:

- RosettaNet [13] provides a framework and predefined business processes called Partner Interface Processes (PIPs) for IT supply chain partners. This is a successful standard that has become a model for other industries as well. Compliance with RosettaNet framework implies automation of business processes among the partners over the Web. Although RosettaNet provides a Business Dictionary, since this is not a public registry, it is difficult to discover other potential partners. In fact the emphasis of RosettaNet framework is not transacting business "on-the-fly" but rather automating business among existing partners.

- UDDI [14], on the other hand, provides a public registry to discover businesses and their services. However UDDI concentrates on the services and does not address public business processes explicitly. A service is an atomic unit of processing; whereas a business process involves collaboration and hence the exchange of a sequence of messages according to a predefined choreography of interactions with well defined roles for the involved partners.

- Whereby ebXML [3] allows arbitrary public business processes to be defined in XML and complements the framework with a registry where both trading partners and their public

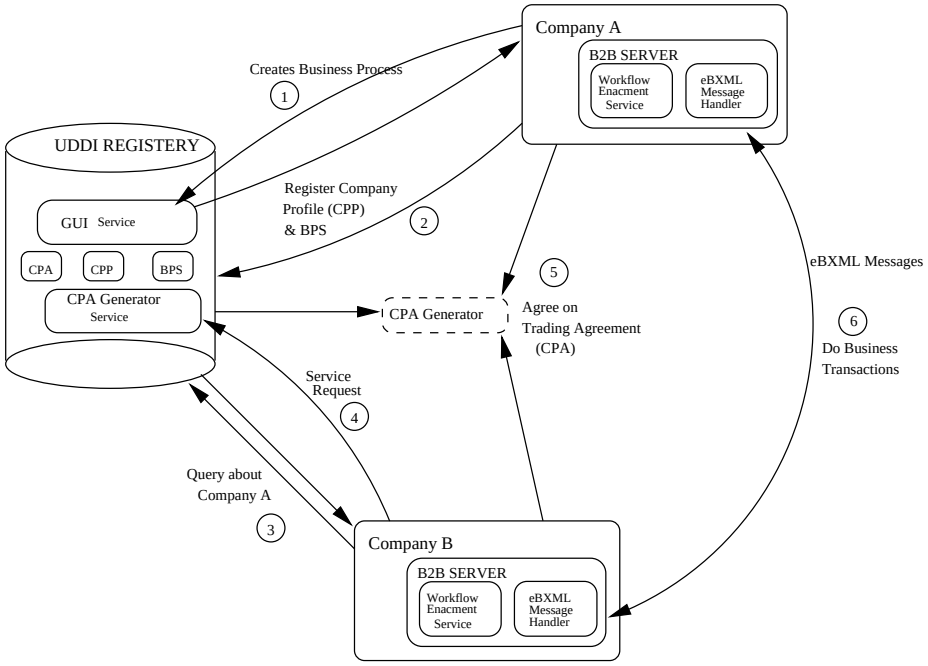

Figure 1: An Overview of the System Architecture

business processes can be discovered. ebXML also has a comprehensive message passing architecture that relies on SOAP with attachments and provides for the security and reliability of the messages. Furthermore ebXML defines a repository and well defined interfaces to query it. Unfortunately a public ebXML registry/repository is not available yet.

These B2B interoperating frameworks are competing and at the same time overlapping standards each with its own strengths and weaknesses. In this paper, we show how these standards can be used to complement one another by using the powerful aspects of each. The infrastructure we have developed exploits RosettaNet's well defined public processes; UDDI's public registry and ebXML's secure, reliable message passing mechanism. The infrastructure also provides a number of add-on services to the combined framework to facilitate conducting business on the Web.

The over all architecture and the contributions of the system can be summarized as follows:

- *The well defined RosettaNet PIPs are used to describe ebXML public business processes.* In ebXML, trading parties collaborate by agreeing on the same business process with complementary roles. If every partner develops its own business process, collaboration becomes very difficult. Therefore there is a need for standardized business processes. In this respect, exploiting the already developed expertise by RosettaNet through PIPs becomes indispensible.

  In order to use RosettaNet PIPs as ebXML business processes, it is necessary to map the PIP definitions into ebXML business process documents conforming to the DTD provided

by ebXML. Since RosettaNet PIPs are defined in Unified Modelling Language (UML), it is possible to automate this conversion. In fact there are tools, such as [11] that generate XML output from UML diagrams through XML Metadata Interchange [19]. On the otherhand mapping RosettaNet PIPs into ebXML processes manually is not a difficult task. We show how to map ebXML public business processes into the RosettaNet Partner Interface Processes (PIPs).

- *A GUI tool is developed which allows users to graphically build their ebXML business processes by combining RosettaNet PIPs.* A RosettaNet PIP corresponds to a "Binary Collaboration" in ebXML. Business processes, however, can include more than one "Binary Collaboration" and there could be more than two party involved (multiparty collaboration). Therefore there is a need for a tool to help with the design of the choreography of multiple "Binary Collaborations" into a business scenario. We provide a GUI which allows users to graphically built their ebXML business processes by combining RosettaNet PIPs. The tool is registered to UDDI as a service to facilitate its discovery.

- *A UDDI registry is used to store ebXML documents like business processes, CPPs and CPAs and mechanisms are provided to facilitate their discovery.* In UDDI, businesses and services can specify the categories to which they belong in their category bags to facilitate their discovery. An item in a category bag contains a tModel key. tModels provide the ability to describe compliance with a specification, a concept, or a shared design. When a particular specification is registered with the UDDI as a tModel, it is assigned a unique key, which is then used in the description of service instances to indicate compliance with the specification.

  There are more than a hundred PIPs and a trading party may include any number of these PIPs into its business process specification (Note however that the order of PIPs can not be arbitrary; for example it does not make sense is a Purchase Order Request (PIP3A4) follows an Invoice Notification (PIP3C3)).

  To facilitate the discovery of a business process, we have defined "tModels" for each of the RosettaNet PIPs. A business process specification includes the tModels of the involved PIPs into its catagory bag. In this way a trading party can discover other parties with compliant business processes by checking their catagory bags to see whether they contain the same PIPs as its own.

- *A tool is provided to automate the process of configuring a Collaboration Protocol Agreement (CPA) from given Collaboration Protocol Profiles (CPPs).* In ebXML a CPA is derived from the intersection of two CPP's. A Collaboration Protocol Profile (CPP) in ebXML

framework contains essential information about the trading partner like contact information, industry classification, supported business processes and messaging service requirements. A CPA describes the *Messaging Service* and the *Business Process* requirements that are agreed upon by both partners. Therefore to configure a CPA, the tool matches *Process Specification, Roles, Transport and Transport Security* and *Document Security* levels of the CPPs. This tool is stored as a service in the UDDI registry.

- *A B2B server is developed to provide mechanisms for properly processing ebXML messages as well as keeping track of the data and control flow in the ebXML business processes.* ebXML business partners need to have software components at their sites to automatically process incoming ebXML messages as well as keeping track of business scenarios they are involved in. We developed a generic B2B Server implementation for this purpose. The server provides an ebXML Messaging Service infrastructure and workflow capabilities to keep track of business processes. The tool itself is available as a service from the UDDI registry.

Figure 1 demonstrates the overall architecture of the system. To be able to use the infrastructure developed, the trading parties need to install the B2B server implementation at their sites. This server has two main functionalities: handling the ebXML messages which basically involves preparing, sending, receiving and routing the messages, and keeping track of data and control flow in ebXML business processes. The next step to become an ebXML compliant business is to create ebXML business processes. We provide a GUI to help with this step which is available as a service from the UDDI registry. The graphical tool helps combine ebXML "Binary Collaborations" to define complex, multiparty ebXML business processes. These "Binary Collaborations" are obtained from RosettaNet PIPs. The trading parties can use this tool to form their business process. To become ebXML compliant, a business also needs to create its CPP and point it to the business process defined. Once all the necessary documents, that is, CPPs, business processes and business documents are created, they are registered to the UDDI registry.

The business parties can use UDDI registry to discover the potential ebXML compliant parties and their CPPs. To facilitate the discovery process we associate a tModel with each PIP and store the related tModel keys in the category bags of business processes.

Once the potential partners are discovered, it is necessary to reach an agreement on the way to carry out the business transactions. It is the Collaboration Protocol Agreement (CPA) that specifies the details of how two trading parties have agreed to conduct business electronically. A CPA is formed by combining the involved CPPs. We also provide a tool to facilitate this step which tries to match corresponding elements like Process Specification, Transport and Roles in CPPs to create a CPA.

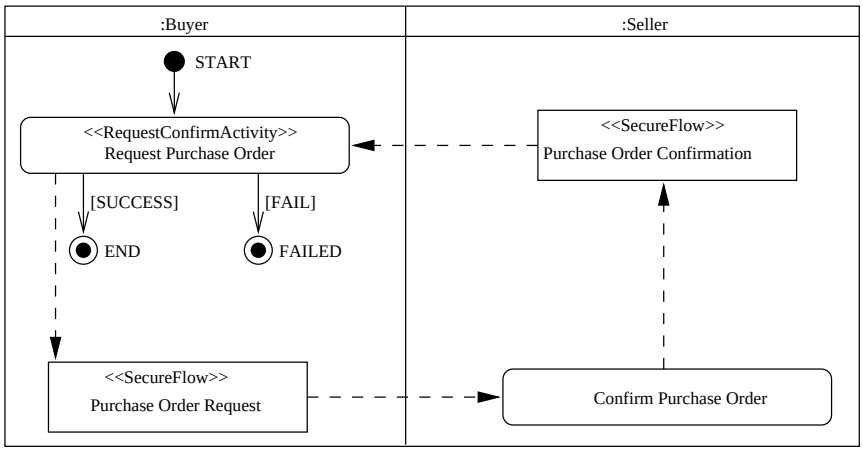

Figure 2: RosettaNet Business Process Flow Diagram for PIP3A4

After each company configures their B2B servers, the system becomes ready for operation and the parties can conduct their business transactions in an ebXML compliant way.

The paper is organized as follows: Section 2 gives an overview of the B2B standards addressed in this work, namely, RosettaNet, UDDI and ebXML. In Section 3 we show how to map RosettaNet PIPs into ebXML business processes. Section 4 presents the GUI Tool developed for flexibly constructing ebXML business processes from RosettaNet PIPs. Registering ebXML documents to UDDI registries is described in Section 5. Section 6 summarizes the tool automating the process of configuring the CPA given two CPPs. The details of the B2B server is presented in Section 7. Finally Section 8 concludes the paper.

2 An Overview of the B2B Frameworks Addressed

A B2B interoperability standard, in general, involves the description of the message formats exchanged (e.g. purchase order), bindings to transport protocols (e.g. HTTP), the sequencing (e.g. after sending a purchase order message, an acknowledgment message must be received), the process (e.g. after a purchase order is accepted, the goods must be delivered to the buyer), and the security to be provided (like encryption, non-repudiation) [1]. Currently there are many Business-to-Business (B2B) electronic commerce standards based on XML [10]. The aim of this section is to briefly summarize the B2B electronic commerce frameworks addressed in this paper, namely, RosettaNet, UDDI and ebXML.

Message Exchange Controls - Request Purchase Order								
#	Name	Time to Ackmowledge Receipt Signal	Time to Acknowledge Acceptance Signal	Time to Respond to Action	Included in Time to Perform	Is Authorization Required?	Is Non-Repudiation Required?	Is Secure Transport Required?
1.	Purchase Order Request Action	2 hrs	N/A	24 hrs	Y	Y	Y	Y
1.1.	Receipt Acknowledgment	N/A	N/A	N/A	Y	Y	Y	Y
1.2.	Purchase Order Confirmation Action	2 hrs	N/A	N/A	Y	Y	Y	Y
1.2.1.	Receipt Acknowledgment	N/A	N/A	N/A	N	Y	Y	Y

Figure 3: RosettaNet Message Exchange Controls for PIP3A4 (Request Purchase Order)

2.1 RosettaNet

Founded in 1998, RosettaNet [13] is an independent, self-funded, non-profit consortium dedicated to the development of XML-based standard electronic commerce interfaces to align the processes between supply chain partners on a global basis. The RosettaNet consortium includes IT companies like IBM, Microsoft, EDS, Netscape, Oracle, SAP, Cisco systems, Compaq and Intel.

RosettaNet Framework [13] consists of Partner Interface Processes (PIPs), a master dictionary and an implementation framework, the relationship among which can be expressed with the following analogy: RosettaNet dictionaries provide the words, the RosettaNet Implementation Framework (RNIF) acts as the grammar, and RosettaNet Partner Interface Processes (PIP) form the dialog.

RosettaNet dictionaries provide a common vocabulary platform for conducting business within the trading network:

- The *RosettaNet Business Dictionary* contains information about the trading partners like Business Properties (e.g. business address), Business Data Entities (like ActionIdentity), and Fundemental Business Data Entities (e.g. BusinessTaxIdentifier, AccountNumber). There is only one business dictionary that encompasses all supply chains like Electronic Components (EC), Information Technology (IT), etc.

Business Document	Description
Purchase Order Request	A request to accept a purchase order for fulfillment
Purchase Order Confirmation	Formally confirms the status of line item(s) in a Purchase Order. A Purchase Order line item may have one of the follwing states: accepted, rejected, or pending.

Figure 4: RosettaNet Business Documents for PIP3A4

PARTNER ROLE DESCRIPTION		
Role Name	Role Description	Role Type
Buyer	An employee or organization that buys products for a partner type in the supply chain	Functional
Seller	An organization that sells products to partners in the supply chain	Organizational

Figure 5: Partner Role Descriptions for PIP3A4

- The *RosettaNet Technical Dictionary (RTD)* provides properties for describing products and services. Note that the RTD integrated the two formerly distinct dictionaries, namely *EC Technical Dictionary* and *IT Technical Dictionary*.

The RosettaNet framework enables supply chain business partners to execute interoperable electronic business (e-business) processes by developing and maintaining PIP implementation guidelines. RosettaNet distributes PIPs to the trading partners, who use these guidelines as a road map to develop their own software applications. PIPs include all business logic, message flow, and message contents to enable alignment of two processes.

In order to do electronic business within the RosettaNet framework, there are a number of steps the partners have to go through. First, the supply chain partners come together and analyze their common inter-company business scenarios (i.e., public processes), that is, how they interact to do business with each other, which documents they exchange and in what sequence. These inter-company processes are in fact, the "as-is" scenarios of their way of doing business with each other. Then they re-engineer these processes to define the electronic processes to be implemented within the scope of the RosettaNet Framework.

An electronic business process includes both the interactions between partner companies, and the private processes within the company. The interactions between supply chain partners are analyzed to create RosettaNet's Partner Interface Processes (PIPs). Note that each partner imple-

Role Name	Activity Name	Acknowledgment of Receipt		Time to Acknowledge Acceptance	Time to Perform	Retry Count	Is Authorization Required?	Non-Repudiation of Origin and Content?
		Is Non-Repudiation Required?	Time to Acknowledge					
Buyer	Purchase Order Request	Y	2 hrs	N/A	24 hrs	3	Y	Y

Figure 6: Business Activity Performance Controls for PIP3A4

ments its own private processes. RosettaNet provides guidelines only for PIPs which are the public part of the inter-company processes.

To have a manageable framework in developing PIPs, RosettaNet has grouped supply chain processes into clusters, which are further grouped into segments. For example, Cluster 3 is "Order Management" and "Segment 3A" in this cluster is about "Quote and Order Entry". As an example of the PIPs in this segment, "PIP3A4: Manage Purchase Order" has the purpose of supporting a process between trading partners that involves issuing a purchase order and acknowledging that purchase order. This PIP also supports the capability to cancel or change the purchase order based on the acknowledgement response.

The choreography, that is, the sequence of steps within a PIP is given in the "blueprint" which is a business process flow diagram. The blueprint for "PIP3A4: Manage Purchase Order" is given in Figure 2. A number of tables provide accompanying information as presented in Figures 3, 4, 5,and 6. The information provided in these figures are used in mapping PIPs to ebXML "Binary Collaborations" as explained in Section 3.

2.2 UDDI

UDDI [14] is jointly proposed by IBM, Microsoft and Ariba. It is a service registry architecture that presents a standard way for businesses to build a registry, discover each other, and describe how to interact over the Internet. Currently IBM and Microsoft are running public registries [15] and Hewlett-Packard is expected to launch a third one. An example screenshot from UDDI registry is shown in Figure 7.

The UDDI information model, defined through an XML schema, identifies five core types of information. These core types are business, service, binding, service specifications information and

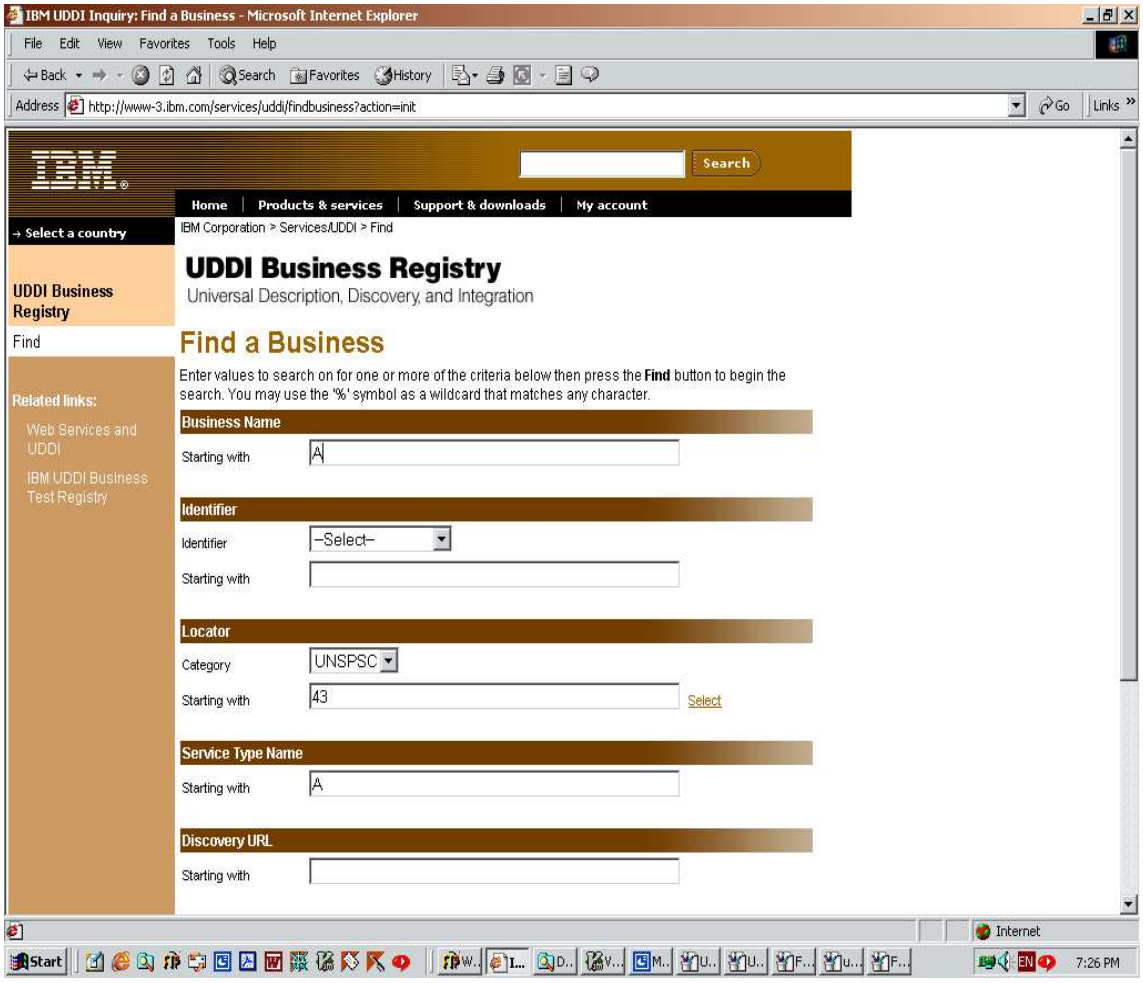

Figure 7: A Screenshot from UDDI Registry

relationship information between two parties. Through these data structures, business entities describe information about businesses like their name, description, services offered and contact data. Business services provide more detail on each service being offered. Services can then have multiple binding templates, each describing a technical entry point for a service (e.g., mailto, http, ftp, phone, etc.). These structures use category bags for categorization purposes. An item in a category bag contains a tModel key and an associated OverviewDoc element.

tModels provide the ability to describe compliance with a specification, a concept, or a shared design. When a particular specification is registered with the UDDI as a tModel, it is assigned a unique key, which is then used in the description of service instances to indicate compliance with the specification. The specification is not included in the tModel itself. The "OverviewDoc" and "OverviewURL" elements of tModels are used to point at the actual source of a specification. More precisely, the use of tModels in UDDI is two-fold:

- *Defining the technical fingerprint of services:* The primary role that a tModel plays is to represent a technical specification on how to invoke a registered service, providing information on the data being exchanged, the sequence of messages for an operation and the location of the service. Examples of such technical specifications include Web Services Description Language (WSDL) [18] descriptions.

- *Providing abstract namespace references:* In UDDI, businesses, services and tModels can specify the categories to which they belong in their category bags. Categorization facilitates to locate businesses and services by relating them to some well-known industry, product or geographic categorization code set. Currently UDDI uses the North American Industrial Classification Scheme (NAICS) [9] taxonomy for describing what a business does; the Universal Standard Products and Services Classification (UNSPSC) [17] for describing products and services offered; and ISO 3166, a geographical taxonomy for determining where a business is located. It should be noted that any number of categories can be referenced in category bags.

The functionality provided by UDDI can be summarized as follows:

- It is possible to locate businesses and their services by their names published in the UDDI registry.

- The categories referenced in the category bags can be used to find businesses or services of a particular category. For example a user looking for a service for a particular product type can first obtain the product code from one of the defined taxonomies, like NAICS [9] or UNSPSC [17]. Assuming that the user wants to access the services related with optical computer disks, he obtains the UNSPSC code of "Magneto optical disks" which is "43.18.16.07.00" and searches the UDDI registry by using the APIs provided to find the businesses and their services that contain this code in their category bags. However if a business fails to provide this exact code in its category bag, it becomes impossible to locate it in this way.

- UDDI expresses the compliance of businesses and services that reference the same tModel in their descriptions.

A comprehensive semantic framework improving service discovery through UDDI is presented in [2].

2.3 ebXML

ebXML [3] is an initiative from OASIS [10] and United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) [16]. ebXML aims to provide the exchange of electronic

business data in Business-to-Business (B2B) and Business-to-Customer (B2C) environments. The vision of ebXML is to create a single set of internationally agreed upon technical specification that consists of common XML semantics and related document structures to facilitate global trade. It should be noted that ebXML is not about creating standard schemes or DTDs for common business documents such as purchase orders or invoices, but instead is about creating an infrastructure.

The ebXML architecture specifies the following functional components:

- *Trading Partner Information:* The Collaboration Protocol Profile (CPP) provides the definition (DTD and W3C XML Schema) of an XML document that specifies the details of how an organization is able to conduct business electronically. It specifies such items as how to locate contact and other information about the organization, the types of network and file transport protocols it uses, network addresses, security implementations, and how it does business (a reference to a Business Process Specification). The Collaboration Protocol Agreement (or CPA) specifies the details of how two organizations have agreed to conduct business electronically. It is formed by combining the CPPs of the two organizations.

- *Business Process Specification Schema (BPSS):* The Specification Schema [7] provides the definition of an XML document (in the form of an XML DTD) that describes how an organization conducts its business. While the CPA/CPP deals with the technical aspects of how to conduct business electronically, the Specification Schema deals with the actual business process. The process specification document defines, among other things, the request and response messages for each business transaction and the order in which the business transactions should occur.

- *Messaging Service:* ebXML messaging service [6] provides a standard way to exchange messages between organizations reliably and securely. It does not dictate any particular file transport mechanism, such as SMTP, HTTP, or FTP.

- *Core Components:* ebXML provides a core component architecture where a core component is a general building block that basically can be used to form business documents.

- *Registry/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 a result of a query. A registry can be established by an industry group or standards organization. A repository is a location (or a set of distributed locations) where a document pointed at by the registry reside and can be retrieved by conventional means (e.g., http or ftp). An ebXML Registry [4, 8] provides a set of services that manage the repository and

```
1. <BusinessTransaction name="REQUEST_PurchaseOrder">
2.   <RequestingBusinessActivity isNonRepudiationRequired="true" timetoAcknowledgeReceipt="PT2H">
3.      <DocumentEnvelope businessDocument='//BusinessDocument [@name="PurchaseOrderRequest"]'/>
4.   </RequestingBusinessActivity>
5.  <RespondingBusinessActivity isNonRepudiationRequired="true" timeToAcknoewledgeReceipt="PT2H">
6.      <DocumentEnvelope businessDocument='//BusinessDocument [@name="PurchaseOrderConfirmation"]'/>
7.  </RespondingBusinessActivity>
8. </BusinessTransaction>

9.  <BinaryCollaboration name=REQUEST_PurchaseOrder" timeToPerform= "PT24H">
10.  <InitiatingRole name="Buyer"/>
11.  <RespondingRole name="Seller"/>
12.  <BusinessTansactionActivity name= "RequestPurchaseOrder" businessTransaction '
                      // businessTransaction[@name= REQUEST_PurcahseOrder"]'
                      fromAuthorizedRole='../Buyer" toAuthorizedRole="../Seller"/>
13.  <Start toBusinessState='../BusinessTransactionActivity [@name= "RequestPurchaseOrder"]'/>
14.  <Success fromBusinessState='../BusinessTransactionActivity [@name= "RequestPurchaseOrder"]'
               guardCondition="Success" guardExpression='//PurchaseOrderConfirmation/
                                                    GlobalDocumantationCode="Accept"/>
15.  <Failure fromBusinessState='../BusinessTransactionActivity[@name= RequestPurchaseOrder"]'
               guardCondition="BusinessFailure" guardExpression='//PurchaseOrderconfirmation/
               GlobalDocumentCode='Reject"'/>
16. <BinaryCollaboration>
```

Figure 8: An Example ebXML Binary Collaboration Document

enable the sharing of information between trading partners. Note that business processes, CPPs, business document desciptions and core components are published and retrieved via ebXML Registry Services. A trading partner may discover other trading partners by searching for the CPPs in the registry. The ebXML Messaging Service is used as the transport mechanism for all communication into and out of the Registry.

The ebXML infrastructure is modular, and with few exceptions these infrastructure components may be used somewhat independently; they are loosely related. The elements of the infrastructure may interact with each other, but in most cases are not required to. The CPP, CPA and Business Process Specifications may be stored in an ebXML compliant Registry, but this is not required. An ebXML compliant Registry may store any type of object, including non-XML objects. However, all communications with the registry must use the ebXML messaging service [12].

3 Mapping RosettaNet PIPs into ebXML Business Processes

Both the RosettaNet PIPs and the ebXML Binary Collaborations define dialogs between e-Business partners and include the following information:

- the sequence of steps required to execute an atomic business process between two trading partners

- the activities involved
- the roles of the partners
- the specification of structure and content of the business documents exchanged
- the security, authentication, time and performance constraints on the interactions.

In other words RosettaNet PIPs correspond to ebXML Binary Collaborations. The difference is on how this information is expressed. In RosettaNet, PIPs are defined in Unified Modelling Language (UML) and in ebXML, Binary Collaborations are XML documents conforming to given XML DTDs. Mapping RosettaNet PIPs into ebXML Binary Collaborations is straight forward and can easily be automated by mapping the UML output to an XML document through XMI [19]. However this can also be done manually. In this section we describe this process through an example.

Mapping a RosettaNet PIP into an ebXML "Binary Collaboration" implies filling in the template XML document (with a given ebXML DTD) from RosettaNet Business Process Flow Diagrams and associated Tables for a PIP.

As an example, consider the XML document given in Figure 8. This is an ebXML Binary Collaboration document conforming to ebXML Business Process Specification DTD. This document expresses the semantics of RosettaNet PIP3A4, namely, Request Purchase Order. In this figure, the BusinessTransaction name in line 1 is obtained from the corresponding RosettaNet Business Process Flow Digram given in Figure 2. The values of attributes given in line 2 and line 5, namely, *isNonRepudiationRequired* and *timetoAcknowledge receipt* are available from Table 3 which provides the message exchange controls for "RequestPurchaseOrder" PIP in RosettaNet. The *businessDocument* names in lines 3 and 6 are obtained from Table 4 which describes the Business Documents for the PIP in question. The BinaryCollaboration name given in line 9 is the name of the PIP and the *timeToPerform* attribute are obtained from Table 6 describing Business Activity Performance controls. The roles given in lines 10 and 11 are available from Table 5 which describes "Partner Role Descriptions" for this PIP. Finally, the information in lines 12 through 15 is again available form Business Process Flow Digram given in Figure 2.

It is clear that creating an ebXML "Binary Collaboration" corresponding to RosettaNet PIP involves filling in the ebXML business process template with data obtained from the corresponding RosettaNet PIP which is a straight forward task. As mentioned earlier it is also possible to automate this task by directly mapping RosettaNet PIPs expressed in UML to XML by using tools such as the one given in [11].

4 GUI Tool for Building ebXML Business Process Specifications from Binary Collaborations

A business process in ebXML describes how trading partners take on roles, relationships and responsibilities to facilitate interaction with other trading partners. The interaction between roles takes place as a choreographed set of "Binary Collaborations".

Consider, for example, a scenario where a buyer requests the price and availability of some products from a seller (PIP3A2). After receiving the response, the buyer initiates a Purchase Order Request (PIP3A4). The seller, on the other hand, after acknowledging the Purchase Order Request, sends an invoice notification (PIP3C3) to the buyer. There is a third party in this scenario, which is a shipper. The seller sends a transportation request (PIP3B1) to the shipper. The shipper, after shipment of the goods, sends the status of the shipment (PIP3B3). When buyer receives the shipment, it sends a shipment receipt notification (PIP4B2) to the seller. Finally, the seller prepares a billing statement and notifies the buyer (PIP3C5).

To facilitate the process of building such complex ebXML business scenarios, we have developed a GUI tool which makes it possible to graphically construct ebXML business processes specifications from Binary Collaborations obtained through RosettaNet PIPs. Figure 9 shows how the described scenario is defined through the GUI tool.

The tool allows the user to select PIPs; drag them onto the canvas and draw the transitions among them. After a PIP is placed onto the canvas the user specifies one of the defined roles as the InitiatingRole in the PIP. A condition expression in XPath language may be inserted for the transitions from one PIP to the other. This expression forms the *ConditionExpression* element which is checked to decide whether Business Transaction Activity will be executed. Another useful attribute of the PIP on the canvas is the URLs of the Binary Collaborations. This attribute is also assigned by the user. A partial output of the GUI tool for the example process definition given above is shown in Figure 10.

5 Registering ebXML CPPs to UDDI Registry

To facilitate the process of conducting eBusiness, potential trading partners need a mechanism to publish information about the business processes they support along with specific technology implementation details about their capabilities for exchanging business information. In ebXML, this information is available through the Collaboration Protocol Profile (CPP). The CPP contains essential information about the trading partner like contact information, industry classification, supported Business Processes, Interface requirements, etc.

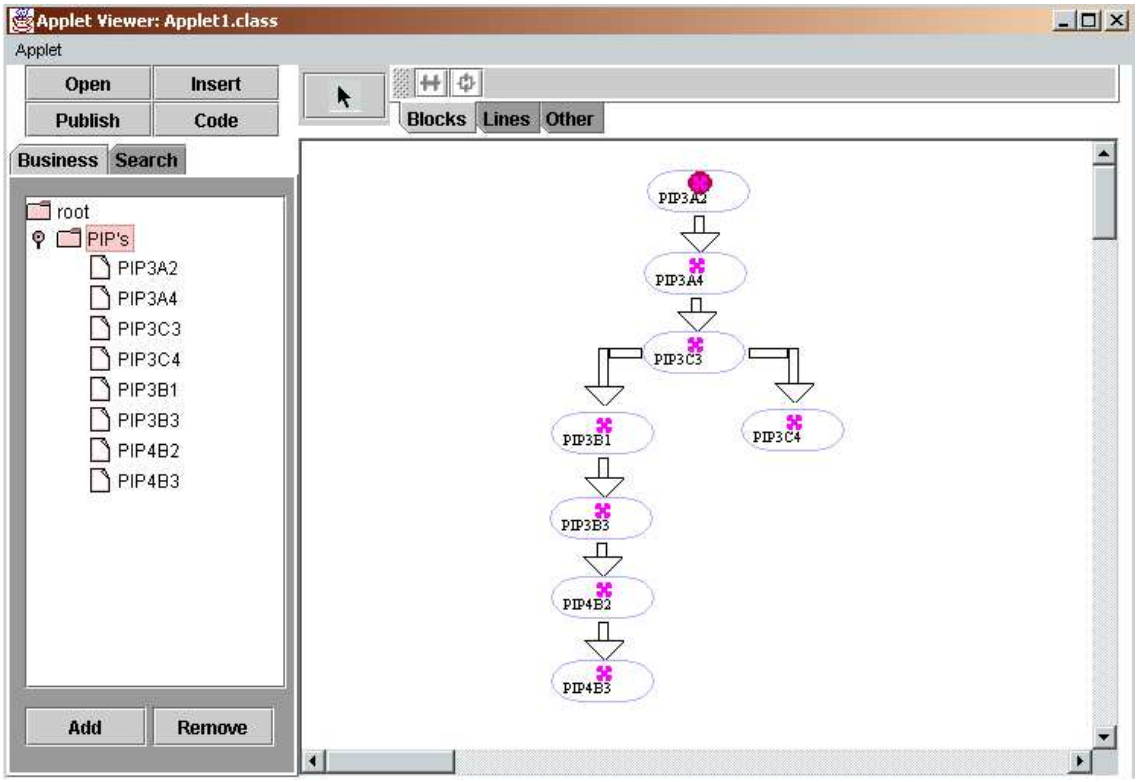

Figure 9: An example business process definition with GUI tool

In the infrastructure developed, ebXML compliant documents are registered to UDDI. While registering CPPs to UDDI registry, a mechanism is necessary to facilitate their discovery. Currently there is "uddi-org:types" taxonomy where, for example, Web Services Description Language (WSDL) is classified as "wsdlSpec". It would help to discover CPPs whose business process specifications are based on RosettaNet PIPs if RosettaNet is classified with "uddi-org:types" taxonomy like WSDL.

In our implementation we have defined tModels for each of the PIPs to help with their discovery. By deveoping a tModel for each PIP and including the keys of these tModels in the category bags of the related CPPs, we facilitate the discovery of CPPs referencing business processes conforming to RosettaNet PIPs.

Other documents registered at UDDI include the business documents exchanged by the parties. The business documents we use in our implementation are based on the DTDs provided by RosettaNet. However, it is possible to use a common document library like Universal Business Language (UBL) as announced by OASIS when it becomes available in the future.

```
<ProcessSpecification name="buysell">
   <BusinessDocument name="PriceandAvailabilityRequest"/> <BusinessDocument name="PriceandAvailabilityResponse"/>
   <BusinessDocument name="PurchaseOrderRequest"/> <BusinessDocument name="PurchaseOrderConfirmation"/>
   <BusinessDocument name="InvoiceNotification"/> <BusinessDocument name="InvoiceRejectNotification"/>
   <BusinessDocument name="TransportationProjectionNotification"/> <BusinessDocument name="ShipmentStatusNotification"/>
   <BusinessDocument name="ShipmentReceiptNotification"/> <BusinessDocument name="BillingStatementNotification"/>
   <MultiPartyCollaboration name=abc>
      <BusinessPartnerRole name="Buyer">
         <Performs initiatingRole='//BinaryCollaboration[@name="REQUESTPriceandAvailability"]/InitatingRole
                                   [@name="Customer"]'/>
         <Performs initiatingRole='//BinaryCollaboration[@name="REQUESTPurchaseOrder"]/InitatingRole
                                   [@name="Buyer"]'/>
         <Performs RespondingRole='//BinaryCollaboration[@name="NOTIFYofInvoice"]/RespondingRole
                                   [@name="InvoiceReceiver"]'/>
         <Performs RespondingRole='//BinaryCollaboration[@name="NOTIFYofInvoiceReject"]/RespondingRole
                                   [@name="InvoiceRejectReceiver"]'/>
         <Performs RespondingRole='//BinaryCollaboration[@name="DISTRIBUTEShipmentStatus"]/RespondingRole
                                   [@name="IntransitInformationUser"]'/>
         <Performs initiatingRole='//BinaryCollaboration[@name="NOTIFYShipmentReceipt"]/InitatingRole
                                   [@name="Consignee"]'/>
         <Transition fromBusinessState='//BinaryCollaboration[@name="REQUESTPriceandAvailability"]
                                        /BusinessTransactionActivity[@name="PriceandAvailabilityRequest"]'
                                        toBusinessState='//BinaryCollaboration[@name="REQUESTPurchaseOrder"]
                                        /BusinessTransactionActivity[@name="RequestPurchaseOrder"]'/>
         <Transition fromBusinessState='//BinaryCollaboration[@name="REQUESTPurchaseOrder"]
                                        /BusinessTransactionActivity[@name="RequestPurchaseOrder"]'
                     toBusinessState='//BinaryCollaboration[@name="NOTIFYofInvoice"]
                                     /BusinessTransactionActivity[@name="NotifyOfInvoice"]'>
            <ConditionExpression expressionLanguage="xpath" expressionCondition='//DocumentSpecification/
                                          BusinessDocument[@name=\PurchaseOrderConfirmation"]/PurchaseOrder
                                          /GlobalPurchaseOrderStatusCode="Accept"'/> </Transition>
         <Transition fromBusinessState='//BinaryCollaboration[@name="NOTIFYShipmentReceipt"]
                                        /BusinessTransactionActivity[@name="ShipmentReceiptNotification"]'
                     toBusinessState='//BinaryCollaboration[@name="NOTIFYofBillingStatement"]
                                     /BusinessTransactionActivity[@name="NotifyOfBillingStatement"]'/>
      </BusinessPartnerRole>
      <BusinessPartnerRole name="Seller">...  </BusinessPartnerRole>
      <BusinessPartnerRole name="Shipper">...  </BusinessPartnerRole>
   <MultiPartyCollaboration/>
   <BusinessTransaction name="REQUESTPriceandAvailability">...  </BusinessTransaction>
   <BinaryCollaboration name="REQUESTPriceandAvailability" timeToPerform="PT24H">...  </BinaryCollaboration>
   <BusinessTransaction name="REQUESTPurchaseOrder">...  </BusinessTransaction>
   <BinaryCollaboration name="REQUESTPurchaseOrder" timeToPerform="PT24H">...  </BinaryCollaboration>
   <BusinessTransaction name="NOTIFYofInvoice">... </BusinessTransaction>
   <BinaryCollaboration name="NOTIFYofInvoice">
      <InitiatingRole name="InvoiceProvider"> </InitiatingRole>
      <RespondingRole name="InvoiceReceiver"> </RespondingRole>
      <BusinessTransactionActivity businessTransaction="//BusinessTransaction[@name="NOTIFYofInvoice"]"
                                   fromAuthorizedRole="../InvoiceProvider" name="NotifyOfInvoice"
                                   toAuthorizedRole="../InvoiceReceiver">
      </BusinessTransactionActivity>
      <Start toBusinessState="../BusinessTransactionActivity[@name="NotifyOfInvoice"]"> </Start>
      <Success fromBusinessState="../BusinessTransactionActivity[@name="NotifyOfInvoice"]" guardCondition="Success">
      </Success>
      <Failure fromBusinessState="../BusinessTransactionActivity[@name="NotifyOfInvoice"]"
               guardCondition="BusinessFailure"> </Failure>
   </BinaryCollaboration>
   <BusinessTransaction name="NOTIFYofInvoiceReject">...  </BusinessTransaction>
   <BinaryCollaboration name="NOTIFYofInvoiceReject">...  </BinaryCollaboration>
   <BusinessTransaction name="DISTRIBUTETransportationProjection">...  </BusinessTransaction>
   <BinaryCollaboration name="DISTRIBUTETransportationProjection" timetoPerform="PT2H">...  </BinaryCollaboration>
   <BusinessTransaction name="DISTRIBUTEShipmentStatus">...  </BusinessTransaction>
   <BinaryCollaboration name="DISTRIBUTEShipmentStatus">... </BinaryCollaboration>
   <BusinessTransaction name="NOTIFYofBillingStatement">...  </BusinessTransaction>
   <BinaryCollaboration name="NOTIFYofBillingStatement" timetoPerform="PT2H">... </BinaryCollaboration>
</ProcessSpecification>
```

Figure 10: An ebXML Business Process Specification produced by the GUI tool (partial)

6 An Automated Tool for Configuring the CPA given CPPs

A Collaboration Protocol Agreement (CPA) describes the *Messaging Service* and the *Business Process* requirements that are agreed upon by both partners. The intent of the CPA is to provide a high-level specification that can be easily comprehended by humans and yet is precise enough for enforcement by computers.

The information in the CPA is used to implement Business Service Interfaces (BSI) to enable exchange of messages with trading parties. The ebXML *Message Service Handler* specification is used to implement the exchange of messages.

ebXML provides the guidelines to generate a Collaboration Protocol Agreement (CPA) from given Collaboration Protocol Profiles (CPPs) which involves the following [5]:

- Matching business processes and the roles
- Matching transport and transport security
- Matching document packaging and document security

6.1 Matching Business Processes and the Roles

As an initial requirement to creating a CPA, the *Process Specification* elements of the two parties must match. To facilitate this matching we use the tModels defined for RosettaNet PIPs.

Matching the roles in two CPP implies checking whether the roles of the partners given in the CPPs are complementary. "PartyInfo" element in a CPP contains a subtree of elements called "CollaborationRole". These sets in the CPPs are compared to find out whether the roles are complementary within the specified *BusinessCollaboration*. As an example, the roles "buyer" and "seller" are complementary in a "Request Purchase Order" business process specification.

6.2 Matching Transport and Transport Security

Matching transport means matching the SendingProtocol capabilities of one party with the ReceivingProtocol capabilities of the other party in a complementary role. The CPP DTD (or Schema) has a ServiceBinding element that points to the relevant information through the "channelId" attribute. "channelId" attribute's value defines the DeliveryChannels within each CPP. The DeliveryChannel has a transportId attribute that specifies the relevant Transport subtrees.

As an example, suppose that a buyer's CPP has the following Transport entry:

```
<Transport transportId = "buyerid001">
   <SendingProtocol>HTTP</SendingProtocol>
   <ReceivingProtocol> FTP </ReceivingProtocol>
   <Endpoint uri = "https://www.buyername.com/
```

```
            po-response" type = "allPurpose"/>
   <TransportSecurity>
      <Protocol version = "1.0">TLS</Protocol>
      <CertificateRef certId =  certid001">BuyerName</CertificateRef>
   </TransportSecurity>
</Transport>
```

Assume further that a seller's CPP has the following Transport entry:

```
<Transport transportId = "sellid001">
   <SendingProtocol>FTP</SendingProtocol>
   <ReceivingProtocol> HTTP </ReceivingProtocol>
   <Endpoint uri = "https://www.sellername.com/
                os_here" type = "allPurpose"/>
   <TransportSecurity>
      <Protocol version = "3.0">SSL</Protocol>
      <CertificateRef certId ="certid002">Sellername</CertificateRef>
   </TransportSecurity>
</Transport>
```

It is clear from the example that the transport elements of the two CPPs match since the "SendingProtocol" and the "ReceivingProtocol" in the complementary roles match; one is HTTP and the other is FTP. If such a match can not be found, then the available exception handling mechanisms will be activated.

Matching transport security involves finding an agreement between the versions and the values of the security protocols specified in the "Protocol" element. To facilitate this process, we provide a Transport Security lookup table that keeps track of compatible protocols and versions. For example if SSL-3 and TLS-1 are defined to be compatible protocols, the examples given above will result in a match in transport security.

6.3 Matching Document Packaging and Document-Level Security

In ebXML messaging structure, message payloads are packaged using the MIME multipart/related content type. MIME is used as a packaging solution because of the diverse nature of information exchanged between partners in eBusiness environments. For example, a complex business transaction between two or more trading partners might require a payload that contains an array of business documents (XML or other document formats), binary images, or other related business information. And due to this possible complexity of the document packaging, checking matches is not a straight forward process. The need to check the document-level security of the involved documents where each might have a radically different security mechanism adds to the complexity of the problem.

"ServiceBindings" elements in CPPs contain the the packaging structure. An example ebXML payload package is given in Figure 11. The CPA tool provided first tries to match the simple

```
<Packaging id="I1001">
   <ProcessingCapabilities parse = "true" generate = "true"/>
   <SimplePart id = "P1" mimetype = "text/xml"/>
   <NamespaceSupported location
       = "http://schemas.xmlsoap.org/soap/envelope/" version = "1.1">
           http://schemas.xmlsoap.org/soap/envelope
   </NamespaceSupported>
   <NamespaceSupported location =
              "http://www.ebxml.org/namespaces/messageHeader"
               version = "1.0">
              http://www.ebxml.org/namespaces/messageHeader
   </NamespaceSupported>
   <NamespaceSupported location =
          "http://www.w3.org/2000/09/xmldsig#"
                version = "1.0">
         http://www.w3.org/2000/09/xmldsig#
   </NamespaceSupported>
   <SimplePart id = "P2" mimetype = "application/xml"/>
   <CompositeList>
      <Composite mimetype = "multipart/related" id = "P3"
                  mimeparameters = "type=text/xml">
           <Constituent idref = "P1"/>
           <Constituent idref = "P2"/>
      </Composite>
   </CompositeList>
</Packaging>

<Packaging id="I2001">
   <ProcessingCapabilities parse = "true" generate = "true"/>
   <SimplePart id = "P11" mimetype = "text/xml"/>
   <SimplePart id = "P12" mimetype = "application/xml"/>
   <CompositeList>
      <Composite mimetype = "multipart/related" id = "P13"
                  mimeparameters = "type=text/xml">
         <Constituent idref = "P11"/>
         <Constituent idref = "P12"/>
      </Composite>
   </CompositeList>
</Packaging>
```

Figure 11: An Example ebXML Payload Package

parts in the packaging and then proceeds with "CompositeList". For matching the document level security, the tool seeks for human assistance due to the complexity involved in this process.

7 A B2B Server Implementation for ebXML Business Process Support

For businesses to become ebXML compliant, they should be able to process incoming ebXML messages. This necessitates a software component at the site of the ebXML compliant partner for handling ebXML messages.

Furthermore, UDDI provides a registry mechanism mainly for describing and invoking Web

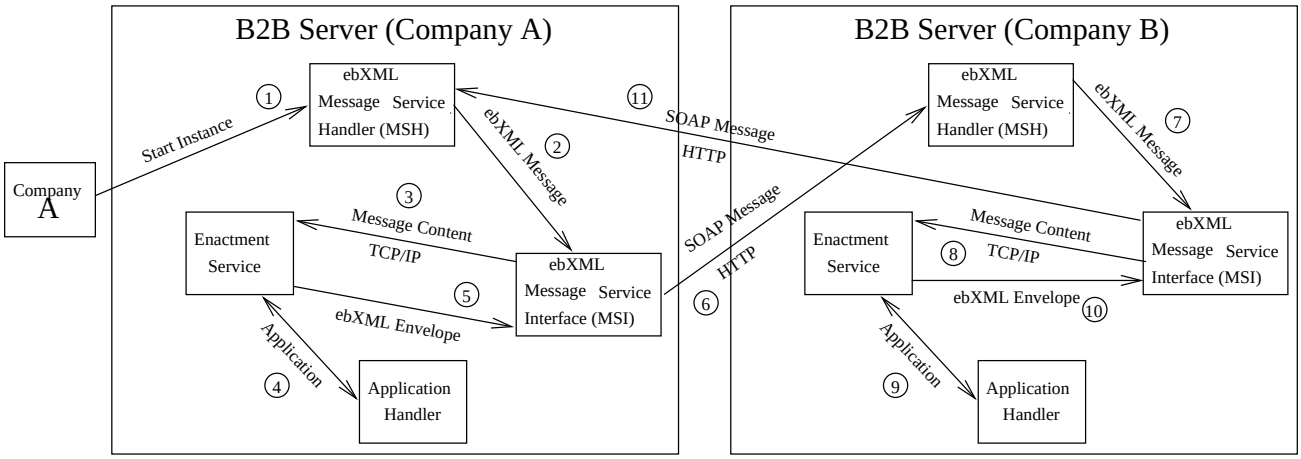

Figure 12: Architecture of the B2B Server

services. In contrast, ebXML, through CPPs, point to the public business processes. The main difference between a service and a business process is as follows: A service is an atomic unit of processing; it is invoked either by passing a message or invoking one or more Remote Procedure Calls (RPCs). Whereas a business process involves collaboration and hence exchange of a sequence of messages according to a predefined choreography of interactions with well defined roles for the involved partners. In ebXML, the atomic unit of interaction between the trading partners is a "Binary Collaboration", and an ebXML public business process may involve more than one "Binary Collaboration" and more than two parties. That is, multi party collaboration is possible.

Therefore, although UDDI registry can be used to discover ebXML CPPs, further mechanisms are necessary at ebXML compliant partners' sites to keep track of the message traffic.

Hence, providing an ebXML infrastructure implies providing mechanisms to ebXML trading partners for properly processing ebXML messages as well as keeping track of the data and control flow in the ebXML business processes. We developed a generic B2B Server implementation for this purpose. The main functionalities of the server are to implement an ebXML Messaging Service infrastructure and to provide workflow capabilities to keep track of business processes.

7.1 ebXML Messaging Service Implementation

An ebXML Message Service mechanism [6] is implemented as a component of B2B server. An ebXML message consists of an optional transport protocol specific outer envelope which contains a protocol independent ebXML Message Envelope as its payload. The ebXML Message Envelope is packaged using the MIME multipart/related content type.

The ebXML Message Service may be conceptually broken down into following three parts: (1) an abstract Service Interface, (2) functions provided by the Message Service Handler (MSH), and (3) the mapping to underlying transport service(s).

The ebXML message processing involves the following:

- Header Processing - the creation of the SOAP Header elements for the ebXML message.
- Header Parsing - extracting or transforming information from a received SOAP Header or Body element into a form that is suitable for processing by the MSH implementation.
- Security Services - digital signature creation and verification, authentication and authorization.
- Reliable Messaging Services - handles the delivery and acknowledgment of ebXML messages sent with delivery semantics of "Once And Only Once".
- Message Packaging - the final enveloping of an ebXML Message (SOAP Header or Body elements and payload) into its SOAP Messages with Attachments container.
- Error Handling - this component handles the reporting of errors encountered during MSH or Application processing of a message.
- Message Service Interface - an abstract service interface that applications use to interact with the MSH to send and receive messages and which the MSH uses to interface with applications that handle received messages.

The Message Service Handler is implemented in Java using Apache SOAP library. Apache SOAP implementation provides for adding MIME attachments to the SOAP messages. Messaging Service Implementation provides an API to facilitate preparing ebXML messages. The API includes interfaces to form the fields of an ebXML message, to construct the envelope of the message and to prepare a SOAP message. The software developed also provides a transport mechanism to send prepared message to the destination according to the related protocol (e.g. HTTP, SMTP, FTP). It uses the corresponding protocols of SOAP messaging service for this purpose. Note that the transport protocol to be used is specified in the CPA.

The Message Service Interface implemented interacts with the Message Service Handler to send, receive and route the messages. The routing information that is, locating the software to process the message is obtained from the "eb:Service" element in the ebXML message header. Note that the "eb:Service" element is originally contained in the CPA. Since the B2B server handles all the message traffic and locates the applications to be invoked, the type attribute of "eb:Service" element in the CPAs using this infrastructure should be "b2bserver".

7.2 Workflow Enactment Service

Enactment service component of the B2B server handles the CPA and the ebXML business processes running at trading partners' sites. Note that the CPA and the process specification document that it references define a conversation between two parties. The conversation represents a single unit of business as defined by the "Binary Collaboration" component of the Process-Specification document. The conversation consists of one or more "Business Transactions", each of which is a request message from one party and zero or one response message from the other party.

The initiating party in each business process starts an instance of the business process by sending a special message to the B2B server. This message includes the business process id, type (e.g. binary or multiparty) and the name of the collaboration to be executed. Upon receiving this message, B2B server activates the enactment service, and the process instance is started. Enactment service assigns a unique id and a conversation id to each business process instance created. ConversationId together with the CPAId, uniquely identifies the instance at all the sites involed.

As described in Section 7.1, B2B server receives messages from other parties through ebXML messaging service and passes it to the enactment service. Enactment service in turn, parses the ebXML message, and locates the related business process instance and its state using the information provided in the message.

7.3 How the System Works

The architecture of B2BServer is shown in Figure 12. Assuming that Company A has the InitiatingRole in the business process, an instance of a business process is started by sending a special message to the B2B Server of Company A. The ebXML Message Service Handler (MSH) component of the B2B Server receives the message and forwards it to the ebXML Message Service Interface (MSI), which is the component responsible for sending and receiving ebXML Messages on behalf of the Enactment Service component. Upon receiving the ebXML message, MSI decomposes the message into its parts (e.g. message header, message body, attachments, etc.) and sends these parts to the Enactment Service via TCP/IP. Enactment Service then locates the related business process and starts a new instance.

It should be noted that an electronic business process includes both the interactions between partner companies, and the private processes within the company. Each Party executes its own internal processes and interfaces them with the Business Collaboration as described by the CPA and Process Specification documents. The execution of a private business process may involve activating some internal processes (Business Transaction Activity, BTA, in ebXML terminology) at the trading partner's site. To automate the invocation of Business Transaction Activities, the

B2B server provides a mechanism through its configuration file to assign applications to the BTAs specified in business processes.

After the completion of internal private processes, the Enactment Service sends a message to the B2B Server of the other party to inform the result. To send a message, Enactment Service contacts Message Service Interface (MSI) by sending an ebXML envelope prepared using the messaging API provided. MSI then sends this message to the specified party using SOAP protocol. Finally the Enactment Service checks the transitions of the related business process and proceeds to the next state (if any). Company B, on the other hand, has the responding role and processes the messages it receives according to the process definition exactly in the same way.

8 Conclusions

In this paper, a *hybrid* B2B infrastructure is described that exploits the strengths of some of the existing standards, namely ebXML, UDDI and RosettaNet. An UDDI registry is used to store ebXML documents including ebXML business processes that are obtained through RosettaNet PIPs. A number of add-on services to facilitate B2B e-commerce are also provided by the system developed, like a GUI to construct ebXML compliant business processes; an enactment service to keep track of data and control flow in the execution of these processes; a tool to assist the derivation of CPAs given two CPPs; and an ebXML messaging infrastructure.

A prototype of the system developed within the scope of the ebXML project is available from "http://www.srdc.metu.edu.tr/ebXML" upon request. The prototype is implemented in Java JDK 1.2. Xerces XML parser is used to parse necessary XML documents. Jakarta Tom Cat servlet Engine is used for Servlet related operations. Apache SOAP, javamail-1.2 and jaf-1.0.1 are used together to implement the ebXML Messaging Service. Visual Cafe is used to implement the GUI tool for Building ebXML business processes.

UDDI provides a publicly available registry. However, with the current specification of UDDI, the data pointed by the registry are scattered all over the Web, which makes it difficult and sometimes even impossible to access them. The hosting site may go down or data may be deleted. To address this problem, there is a need for a "repository" associated with the UDDI registry. ebXML's repository specification, when implemented can fill this void.

References

[1] Bussler, C., "B2B Protocol Standards and their Role in Semantic B2B Integration Engines", IEEE Bulletin of the TC on Data Engineering, http://www.research.microsoft.com/research/db/debull/issues-list.htm, Vol. 24, No. 1, March 2001.

[2] Dogac, A., Cingil, I., Tambag, Y., Laleci, G.B., Kabak, Y., "Describing the Semantics of Web Services and UDDI", submitted for publication.

[3] ebXML, http://www.ebxml.org/

[4] ebXML Registry Services Specification v1.0, http://www.ebxml.org/specs/ebiRS.pdf, May 2001.

[5] ebXML Collaboration Protocol Profile and Agreement Specification, http://www.ebxml.org/specs/ebCCP.pdf.

[6] ebXML Message Service Specification v1.0, http://www.ebxml.org/specs/ebMS.pdf, May 2001.

[7] ebXML Business Process Specification Schema v1.0, http://www.ebxml.org/specs/ebBPSS.pdf, May 2001.

[8] ebXML Registry Information Model v1.0, http://www.ebxml.org/specs/ebRIM.pdf, May 2001.

[9] North American Industrial Classification Scheme (NAICS) codes http://www.naics. com.

[10] OASIS, http://www.oasis-open.org/cover/siteIndex.html

[11] Objecteering, http://www.objecteering.com/us/produits_pe.htm

[12] Rawlins, M., "Overview of the ebXML architectures", http://rawlinsecconsulting.com/

[13] RosettaNet, http://www.rosettanet.org/

[14] UDDI: Universal Description, Discovery and Integration, http://www.uddi.org, 2001.

[15] UDDI Registry, http://www-3.ibm.com/services/uddi/

[16] UN/CEFACT, http://www.diffuse.org/fora.html#CEFACT

[17] Universal Standard Products and Services Classification (UNSPSC), http://eccma.org/unspsc

[18] Web Service Description Language (WSDL), http://www.w3.org/TR/wsdl

[19] XML Metadata Interchange (XMI), http://www-4.ibm.com/software/ad/library/standards/xmi.html