Cross usage of web services and PLIB ontologies to define a B2B exchange process of product catalogues

Y. Aklouf [1, 2], G. Pierra[1], Y. AIT Ameur[1] and H. Drias[2]

[1] *Industrial and data processing Scientific Laboratory -LISI-*
National Engineering School for Mechanics and Aerotechnics –ENSMA-
86960 FUTUROSCOPE Cedex - FRANCE
Email: {aklouf, pierra, yamine}@ensma.fr

[2] *Electronics and Computer Science Faculty -*
Research Laboratory in Artificial Intelligence, LRIA-USTHB
LP 32, El ALIA, 16111 Algiers, ALGERIA
h_drias@ini.dz

Abstract. The professional electronic commerce (Business-to-Business) relates to the computerization of the exchanges between companies and organizations or, in another term, it provides an inter-organization information system which makes it possible to do any type of automatic information exchange excluding any human intervention. In this context, handling data and knowledge information during the exchange process becomes one key issue in current computer technology and in PLM systems. Therefore, the design of ontologies, which provide a shared and a common understanding of a domain that can be communicated between partners and application systems, becomes fundamental to support exchange and to understand information. This paper proposes an approach that gathers both ontologies and web services technologies in order to build a B2B products exchange model. The proposed model allows companies to share information in different phases of Product Life Cycle Management. When such exchange is planned, it is usually achieved by providing answers to the following three major questions: **Who** are the actors of the exchange: (**WHO ?). How** this exchange is performed (**HOW?) What** kind of information or products are the subject of this exchange (**ON WHAT?).**

Our paper presents a model providing a mechanism for supporting an exchange between companies. It proposes a platform which provides answers to the three questions cited above. To give answer to the **HOW** part, a software architecture is developed based on JAVA/J2EE Web service technologies, and on the PIP2A9 RosettaNet Business Process (BP) model provided as a choreography of exchanges. To answer the **ON WHAT** part, the PLIB ontology is used as a model of contents in order to describe the business elements that are the subject of exchange. It allows reducing semantic ambiguities that may occur when identifying products and theirs characteristics during exchange. Finally, to seek and to find the required Web services, a directory which list these Web services has been developed using the ebXML registry specification. This final part gives answer to the third part of the question: the **WHO** part.

Keywords: B2B, PLIB, Ontology, Business processes, Web services, Java.

1 Introduction

Setting up electronic Business to Business (B2B) relationships is time-consuming and costly. It has been eased to a certain extent by standards such as RosettaNet [17,18] and ebXML[4], which use XML and XML schema [7] technologies to define standardised syntax of messages used in interaction. However, this standardisation activity has necessarily maintained some flexibility to allow companies with different proprietary processes to be in conformity with these standards. Furthermore, the corresponding B2B developed standards and tools describe information from a syntactical point of view and they do not address the semantics of this information: the interpretation of the information is left open and each partner interprets the described information. So, there is a need to provide other standards which define the semantic parts in order to reduce the semantic ambiguities that may lead to inefficient exchange processes. Ontologies are set up for this purpose. Indeed, standard ontologies are developed to cover several available business areas. In parallel, the appearance of the Web services technology makes the B2B aspect of electronic commerce significantly larger. Indeed, the use of Web services to implement Business Processes (BP) reduces significatively the cost and the time of design of B2B relationships. However since the description of these web services is based on XML technologies, the tools provided are primarily syntactic.

In this paper, we describe an approach promoting the semantic aspects which introduce more semantic characteristics in such Web services. More precisely, this approach uses the PLIB [11,12] ontology model to support the description of product data, the RosettaNet Partner Interface Prosess [19] to encode BPs and the ebXML repository to register web services. These three standards are used conjointly to give answer to the questions listed above. The ebXML registry is used to give answer to the **WHO question part**. The RosettaNet PIPs and the web service technology are set up to support Business Processes to answer the **HOW question part.** Finally, for the **On WHAT question part**, dealing with the product representation, the PLIB ontology is put into practice. To structure the different technologies and the corresponding tools, a layered architecture allowing to support, in a modular manner (orthogonal way), the answer to these three questions.

This paper is structured as follows. Next section presents basic concepts related to this work. Then, a layered architecture supporting B2B e-commerce and its various modules is proposed. It ends by detailing the RosettaNet convergence standard. In section 4, we precise our point of view of the B2B system. This section outlines a simplified B2B architecture. The different modules that compose our platform, with the integration of the ontology concepts used in PLIB model(dictionary) and the RosettaNet BP named PIP2A9 (choreography) in the web service platform are described in section 5. This section contains several subsections, each of which gives the details of the different technologies and tools used to develop both Web services and ebXML registries. Sections 6 and 7 present the platform we have developed together with a scenario of use. Finally, section 8 shows how the proposed exchange model can be set up in a PLM setting.

2 Basic Concepts

Several kinds of electronic commerce architectures are available. Among them, we find B2B standards. B2B means the commerce established by two or more businesses (in the most cases between consumer and its providers). Electronic marketplaces for B2B bring together many online suppliers and buyers. They have to integrate many product catalogs provided by partners. Each partner or company can use its own format to represent products in its product catalog. Section 3 depicts an overview of topics related to modern B2B standards in the form of a stack, including content, BP (Business Process) and Registry standards as the most important layers. These elements are gathered in a simplified model proposed in section 4.

The proposed model is represented by three layers defined as follows (Fig1 and Fig 2). The first layer (content layer: Universal business and technical dictionaries), the product, also named payload defines the required information or the unit of exchange between partners. The second one (Business process layer: Universal business process), is related to the BP. It precises the exchange choreography and some information about transmission. Each BP is encoded as an XML document based on a specific DTD or XML schema. So, in order to exchange between partner A and partner B, each partner needs to understand and to have all messages formats and exchange scenarios. The third layer (localization layer: universal registry & repository structure) is the registry layer which shall be set up before doing any business. It acts as a search engine with a specific contents and formats of information. It helps the consumer in finding some potential providers and suppliers of services and products. The relationship between these three layers is detailed in section 4 .

Therefore, if we intend to do business between partners, in a B2B environment, it is necessary to have an architecture that defines and specifies each part or standard used in the system. Since the system is layered and for development purposes, we must ensure that our defined layers have minimal inter-dependencies. Therefore, it is recommended to use a defined standard for each layer promoting the orthogonality principle presented in [30]. Several standards to define B2B architecture in both horizontal and vertical ways shall be used. A vertical standard is a system which is specific to some kind of activities or some particular products e.g: RosettaNet standard specific to semiconductors and electronic information commerce. A horizontal standard is a general system which defines exchange protocols and information formats without referencing any product or service: e.g. ebXML standard.

The approach presented in this paper is based on the application of the orthogonality described above to the RosettaNet convergence model explained in next section.

2.1 The Web Services Technology

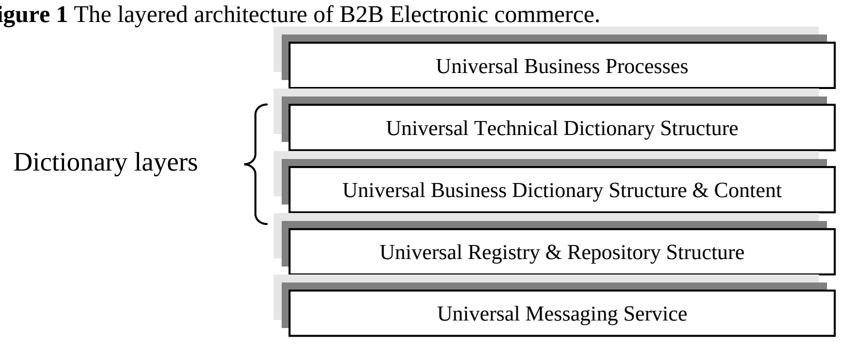
Web services are modular, self-describing applications that can be published and located anywhere on the web or on any local network. Their description is provided by standardized languages on top of XML. The provider and the consumer of the XML web service do not have to worry about the operating system, the language environment, or the component model used to create or to access this XML Web service, as they are based on ubiquitous and open Internet standards, such as XML, HTTP or SMTP. The WSDL (Web Service Description Language [34]) initiated by Microsoft and IBM to describe the messages between client and the web server supports the definition and description of web services. It helps the user to set up a system using a service ranging from connection details to message specification. A WSDL document defines services as a set of network endpoints (ports), that is associated with a specific binding. This binding maps a specific protocol to a port-type composed of one or more operations. In turn, these operations are composed of a set of abstract messages, representing the data. The pieces of data in a message are defined by types.

Most B2B systems are defined with a set of layers. A layered architecture is used for these standards due in the one hand, to the complexity of tasks achieved in electronic commerce exchange, and on the other hand, to the large managed classes of products information. So, the developers of such systems propose a set of layers gathering similar functions and tasks in the same layer in order to obtain a complete layered system for representing B2B architectures. The layers commonly set up in most of the B2B models are presented in next section.

3 The Layered Architecture of B2B Electronic commerce

This part focuses on the definition of a high level architecture for B2B systems. It serve as a framework for both consumers and suppliers. In the framework described herein, the different elements of the defined B2B architecture can be represented as layers, each one built on top of the other i.e. each layer supports the functions provided by the layer below it.

Figure 1 The layered architecture of B2B Electronic commerce.


Universal Business Processes
Dictionary layers
Universal Technical Dictionary Structure
Universal Business Dictionary Structure & Content
Universal Registry & Repository Structure
Universal Messaging Service

3.1 The RosettaNet Convergence standard

As shown previously, a B2B electronic commerce is a layered architecture. RosettaNet defines such a convergence model showing how different organizations and standards can be used efficiently in a common model. The convergence model of RosettaNet [17] is a document defining how the multiple initiatives (SOAP, ebXML, RosettaNet…) are complementary with the aim of setting up a solution for B2B integration in a "Supply Chain" problem. This initiative is an important contribution. Indeed, RosettaNet standard focuses on the articulation of the defined XML based standards.

For RosettaNet developers, a B2B solution contains the layers presented in figure 1 in addition to the specific layers of the trade associated to companies. Below a description of each element of the architecture.

3.1.1Universal Business Processes

This layer specifies BPs which are applicable to a broad range of businesses, regardless of the industry for which the business operates or of the specific characteristics of the business. These processes cover several domains of activity that businesses engage in, such as collaborative product development, request for quote, supply chain execution, purchasing, and manufacturing. A BP is a set of business rules, definitions of the roles of the parties involved, and trigger events that provide the context of information exchange [1,2].

Examples: invoicing process, purchasing process, base level purchase order.
In our work, we recommend the use of RosettaNet PIPs [19] and ebXML BPSS [5] for this layer.

3.1.2 Universal Technical Dictionary Structure (Content oriented ontology)

This component manages the structure for defining form, fit and function of any product or service, regardless of the industry that manufactures it. Specifically, the Universal Technical Dictionary Structure specifies the structure of the specialized technical dictionary or dictionaries used to obtain commonly used domain-specific terminology and accepted values for any product or service. The pre-defined structure is used as the basis for defining supply chain-specific technical dictionary content (i.e. form, fit, and function attributes).
RosettaNat Technical Dictionary (RNTD) is an example of this kind of dictionaries. PLIB ontology model can be used to define technical product information.

3.1.3 Universal Business Dictionary Structure & Content (Business oriented ontology)

The Universal Business Dictionary Structure & Content specifies the structure of payload (business information). It is an aggregation of all content fields, elements, constraints, code lists and objects used within electronic BPs. It describes all of the business content, attributes and relationships between elements that exist in all business documents. For example, needed field descriptors for a query of a price and availability would be: product quantity, product identification, global currency code, and monetary amount.

3.1.4 Universal Registry & Repository Structure

This layer defines both registry and repository.

A. Repository Structure represents the standardized repository services that specify the structure, access protocol and schemas for business content storage and retrieval. It includes terms, constraints, representations, etc. An example of such repository models is the RosettaNet Dictionary Repository or the ebXML Reg/Rep.

B. Registry Services specify the structure and the access protocol of registries and repositories. It is accessed by trading entities to discover each other's capabilities and services. It covers naming, directory, registry, privacy, authorization and identification services. The registry in this layer is used to publish and register BPs and services. An example of such registries are the ebXML registry or the UDDI(Universal Description, Discovery and Integration).

3.1.5 Universal Messaging Service

This layer defines a standardized message and envelope structure and layout definitions, with their specific technical purposes. It addresses the need to record session and communication settings for message transport in order to enable coordination between parties in a business transaction, including parameters that control Reliable Messaging, Secured Messaging, etc. In this layer, the most used specification is RosettaNet RNIF1.1[18], SOAP and ebXML MS[4].

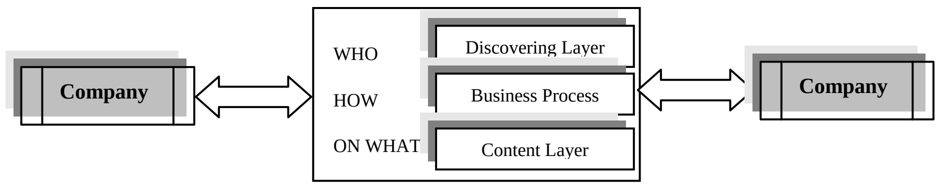
Adopting this architecture eases the use, in any layer, of the most used standard and the best technologies. Proceeding this way preserves our defined orthogonality principle put into practice in the next sections.

4 Overview of the proposed B2B model

This part of the paper presents our proposal of a B2B model. An organization of a B2B system is suggested. It represents our point of view through a stack of three layers or levels (see figure 2). The role of each layer is to deliver necessary information needed by the other layers. The first task is about discovering and localization of the partner of exchange. This activity is devoted to the business process layer. Once the partner is localized and different software actions are processed, the next step deals with the exchange performed by the two remaining layers. One layer, the business process layer, defines the scenario of exchange, or the order of operations triggered by partners involved in the exchange. The second layer, the content layer, defines the payload or the content which actually represents the needed information.

The integrated framework for B2B business model with the major structured components and interactive relationships is illustrated in figure 2.

Figure 2 The proposed B2B Model



These three levels or layers represent three concepts and can be viewed as open questions for which answers shall be provided. The answer to each question is given by the proposed architecture. To communicate and to exchange information and services between partners using this simplified B2B model, it is necessary to give answers and solutions to these three questions: WHO (Discovering Layer), HOW (Business Process Layer) and ON WHAT (Content or Product Ontology Layer) with details about their implementation. So, doing commerce using this architecture starts as follows:

- First, the customer tries to use the Discovering layer (Universal Registry & Repository Structure layer in figure 1) to retrieve a partner (supplier);
- Next and once this partner (supplier) is found, a collaboration agreement is established using the BP layer (Universal Business Processes layer in figure 1);
- Finally, we start exchange using a content format accepted by both actors of the agreed exchange (technical and business dictionaries layers in figure 1).

4.1 WHO (Discovering Layer)

The first question asks to find the partners with whom information is exchanged i.e. *who* is the company that provides exactly what is needed by a consumer. A mechanism where business documents, information and relevant metadata can be registered, and retrieved as a result of a query must be available. Businesses can discover, interact, and share information between each partner involved in the exchange. Before making exchanges, it is necessary to find in a more or less specific way the partners involved in the exchange process. It is necessary to use or to build a registry that includes more detailed business process and specific information about services. Using this registry, companies will have a standard method to exchange business messages, conduct trading relationships, communicate data in a common way, define and register BPs. Directories such as yellow pages, UDDI and recently its ebXML registry are provided for this purpose.

4.2 HOW (Business Process Layer)

Once, the exchange partner is located, it is necessary to agree on the various exchange steps or on the exchange scenarios named: business processes. A BP represents an ordered set of interactions and an interaction is a basic exchange of messages between partners. Interaction types are specified in a declarative way by means of predefined constructs supporting common interaction patterns.

As example, a typical action pattern is the request/response pattern (one partner sends a purchase order request, and the other responds with an acknowledgement, meaning that the order is effective). Most of the BPs are represented by UML models using state, collaboration and/or sequence diagrams or by XML schemas to specify binary and multiple collaborations between partners. Currently, the developers aim at creating a generic meta-model for BPs allowing to model any computer sensible BP.

4.3 ON WHAT (Content Layer or Product Ontology)

The content layer specifies structure and semantics of information items, including their possible refinement or composition and various constraints like cardinality. Ontologies and PLM systems characterise this level. Two types of ontology information may be found in this layer: 1) the business oriented ontology identifies business properties that are independent of any particular product and needed to define transactions between trading partners, 2) and the second is the product oriented ontology which represents products by means of a set of classes, properties, domains of values and instances of objects. It provides a shared common meaning of the products that can be used by two or more partners during exchange [26, 23].

As examples, several dictionaries can be used in this layer. RosettaNet uses two separate dictionaries: one to represent business ontology and the other to represent product ontology [13, 14]. PLIB standard ontologies can be used to define both business and product ontologies either in single or in separate dictionaries.

Once the tools and the environment to be used in each layer are defined, the integrated or the complete system defines the operational B2B model in which transactions between partners can be started and performed automatically.

5 The proposed infrastrcture

Our work proposes the implementation and the realisation of a B2B platform based on the three layers shown in the previous section. The next, is to give answers to the three previously outlined questions and give a practical solution implementing these different levels.

1) First of all, in the discovering layer of our system, an ebXML registry is developed for this level;

2) in the business process layer, a web service, based on the PIP2A9 RosettaNet BP for product information query, is implemented,

3) and finally, the content layer uses a PLIB model with its various tools (PLIBEditor and PLIBBrowser) to search and to retrieve the product information content.

The following sections describe in more details the several parts of the architecture.

5.1 ebXML Registry

The ebXML registry is central to the ebXML architecture [24]. The registry manages and maintains the shared information as objects in a repository. Repositories provide trading partners with the shared business semantics, such as BP models, core components, messages, trading partner agreements, schemas, and other objects that enable data interchange between companies. The ebXML registry is an interface for accessing and discovering shared business semantics. In this section, we explain the registry usages, the business semantic model, and the registry functionality including registry classification, registry client/server communications, searching for registry objects, and managing registry objects.

Our registry implementation is based on information in the primary ebXML registry reference documents, including the "ebXML Registry Service Specification [29]" and the "ebXML Registry Information Model [27]." In a marketplace populated by computer companies with proprietary hardware, operating systems, databases, and applications, ebXML gives business users and IT groups control over their lives. The ebXML registry is not bound to a database product or a single hardware vendor. It is designed to interoperate on all kinds of computers. An ebXML registry serves as the index and application gateway for a repository to the outside world. It contains the API that governs how parties interact with the repository. The registry can also be viewed as an API to the database of items that supports e-business with ebXML. Items in the repository are created, updated, or deleted through requests made to the registry.

5.1.1 How the Registry Works

The complete registry system consists of both registry clients and services. All ebXML registry implementations have to support a minimal set of interfaces and corresponding methods as standard interface. The server-side interfaces are registry service, object manager, and object query manager interfaces. The client-side interface is the registry client interface. The registry applications can be written in a variety of programming language, such as Java, Visual Basic, or C++. In our case, the Java language [28] is used with some additional tools: JAXR and JAXM APIs respectively named Java XML APIs for registry and for messaging [25].

There are a few generic objects in the registry design: registry entries, packages, classification nodes, external links, organizations, users, postal addresses, slots (for annotation purposes), and events (for auditing purposes). The registry entry is a basic registry unit for submitting and querying registry objects, containing a crucial reference to the actual document or data. Every repository item is described by a registry entry. For example, a CPP (Collaboration Profile Protocol), a document defining an enterprise profile, is stored in the repository. It has a registry entry that allows finding the actual CPP stored in the repository.

5.2 Designing our PIP2A9 Web service

This section presents the mechanisms and the approach used to design the Web service using the PIP2A9 as BP. First, the PIP2A9 role and tasks are introduced, after that, some details about the matching between PIP2A9 and Web service are outlined.

5.2.1 PIP2A9

RosettaNet aims to align the BP of supply chain partners. This goal is achieved by the creation of Partner Interface Processes or PIPs. Each PIP defines how two specific processes, running in two different partners' organizations, will be standardised and interfaced across the entire supply chain. PIP includes all business logic, message flow, and message contents to enable alignment of the two processes. RosettaNet defines more than one hundred PIPs. The purpose of each PIP is to provide a common business/data models and documents enabling system developers to implement RosettaNet eBusiness Interfaces.

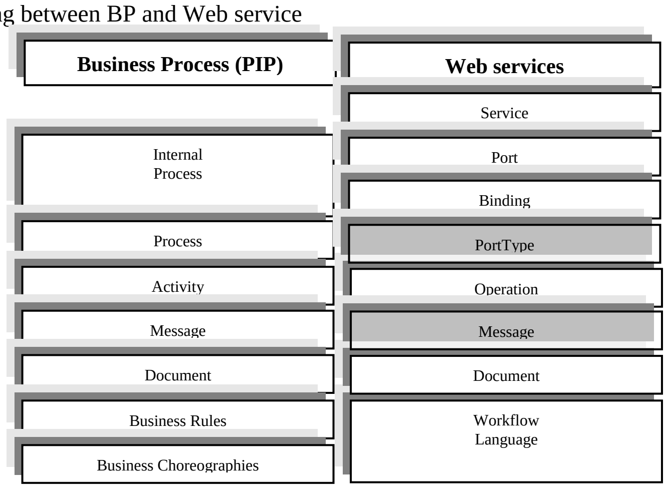
The PIP studied in this paper is the PIP2A9 (Query Technical Product Information) [19]. Technical product information is the category of information that describes the behavioural, electrical, physical, and other characteristics of products. There are numerous classes of customers within the supply chain that need to be able to access product technical information. These include distributors, information providers (such as web portal companies, other commercial information aggregators, and end-customer information system owners), engineering, design engineering, manufacturing and test engineering.

5.2.2 Matching between BP and Web Services

Web services are modular, self-describing applications that can be published, located anywhere on the web or on any local network. The provider and the consumer of the XML web service do not have to worry about the operating system, language environment, or component model used to create or to access the XML Web service. They are based on ubiquitous and open Internet standards, such as XML, HTTP, and SMTP. An initiative from Microsoft and IBM to describe the messages between client and the web server, WSDL (Web Service Description Language), describes and defines web services. It helps the user to set up a system using a service ranging from connection details to message specification. A WSDL document defines services as a set of network endpoint (ports) that is associated with a specific binding. This binding maps a specific protocol to a port-type composed of one or more operations. In turn, these operations are composed of a set of abstract messages, representing the data. The pieces of data in a message are defined by types [22].

Due to the similarity between BPs and Web Services and to the small gap between them, a mapping between these two technologies is defined. The result of this matching is shown in figure 4. Both BPs and web services make use of messages containing documents. They perform the same function in both theories. An atomic activity from a BP workflow corresponds to a single web service operation. An activity is wrapped into a process and therefore, it is more complicated than an atomic activity. With the web services, exchange operations are between entities: processes and the system (e.g connection details). In ebXML, these entities are companies defined by CPPs but in RosettaNet they are covered by the internal processes which communicate with other internal processes defined in another place. In order to match with these technical information, Web service side ports and bindings are used. Finally, the concept of service is defined as a collection of operations (port types) which are logically related by relationships other than a sequence like in a workflow. Here, no corresponding concept in BPs exists.

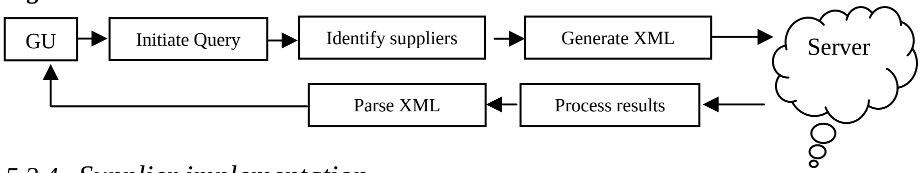
Figure 3 Matching between BP and Web service



5.2.3 Customer implementation

The client must generate with its internal system a local request and then, transform it in order to be in the format accepted by the service. The message transformation task, is realized by the Web service. Figure 4 recapitulate these different operations:

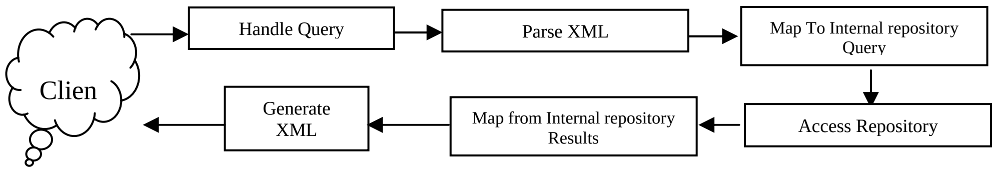
Figure 4 Consumer realised tasks.



5.2.4 Supplier implementation

Once the requested document reaches the supplier (server), the first task extracts the data request and translates it to the internal query format. In our case, the request is translated to OntoQL Format. After the interrogation of the internal repository, the response is mapped from this internal format to the response document format used by the Web service. Finally, the server using the SOAP service, sends the response to the client who analyses the response locally. Figure 5 shows these different steps.

Figure 5 Supplier realised tasks.


Handle Query
Parse XML
Map To Internal repository Query
Clien
Generate XML
Map from Internal repository Results
Access Repository

5.3 The PLIB Ontology in the Content Layer

The third and the last level of our architecture is the content layer. As shown previously, this layer defines data and products information for which the use of PLIB ontology model is proposed. PLIB, the Parts Library standardisation initiative, was launched at the ISO level in 1990. Its goal is to develop a computer-interpretable representation of parts library data to enable a full digital information exchange between component suppliers and users. A PLIB data dictionary [12, 20] is based on the object oriented concepts [3]: components are gathered in parts families that are represented by classes. This set of classes is organised in a simple hierarchy (on which factorisation/inheritance applies) of classes. Such classes are then, precisely described (textually, with technical drawings,...). Finally, each class is associated with a set of technical properties, also precisely described (domain of values, possible measurement unit,...). A basic idea of the definition of a PLIB dictionary is that properties and classes shall be defined simultaneously: applicable properties allow to define precisely a parts family, and conversely, a parts family determines the meaning of the property in its particular context.

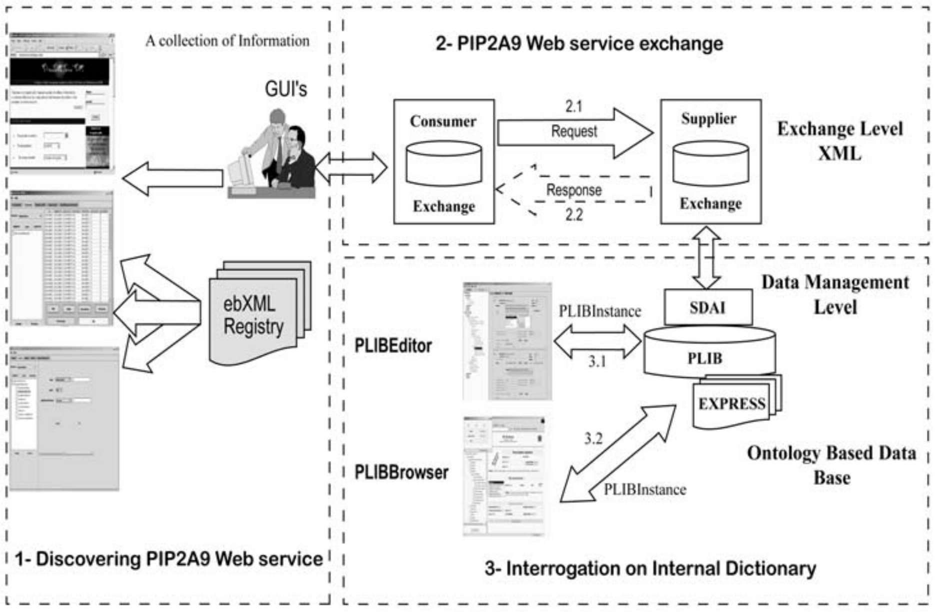
The modelling formalism used in PLIB is the EXPRESS language. The ontology model of PLIB is formally defined and a number of tools have been developed to create, validate, manage or exchange ontologies. They are available on the PLIB server (http://www.plib.ensma.fr) at LISI/ENSMA. The basis of all these tools is a representation of ontology in a processable exchange format. It can be exchanged in various automatically generated formats: XML document (SimPLIB DTD), associated possibly with XSL page (SimPLIBVIEWER), EXPRESS physical file or DHTML document (PLIBBrowser). Finally PLIBEditor makes it possible to create and publish an ontology [20, 21].

PLIBEditor is a tool which allows users to define and to represent graphically and simultaneously a defined ontology and its instances. It is an application containing two parts on its user interface, the left part (frame) allows the definition of classes, properties and relations between them, and the right part defines the ontology population (instances). In another way, PLIBEditor defines in the same application data and metadata of an ontology. PLIBBrowser has the same role as PLIBEditor but it has a Web oriented presentation.

6 Scenario of use

To use our system, the first task is the connection to the ebXML registry to find the involved partners which supports the PIP2A9 exchange. This task is achieved in step 1 of figure 6. Next step, is the implementation of different technical details about this PIP2A9 using a Web service technology. Once, this task is performed, the actual exchange can start using step 2 of figure 6. The Web service represents a set of documents exchanged between partners using a standard agreed format for both the request and the response. It is required that, the client sends a request using step 2.1 and receives the response using step 2.2 of figure 6. The exchange concerns the PLIB defined product. This exchange is realized using PLIBEditor and PLIBBrowser tools to navigate and to visualise in a conformed format our results using steps 3.1 and 3.2 of figure 6. As results, the general architecture of our system is defined in figure 6.

Figure 6 The different Components of the Web service Platform

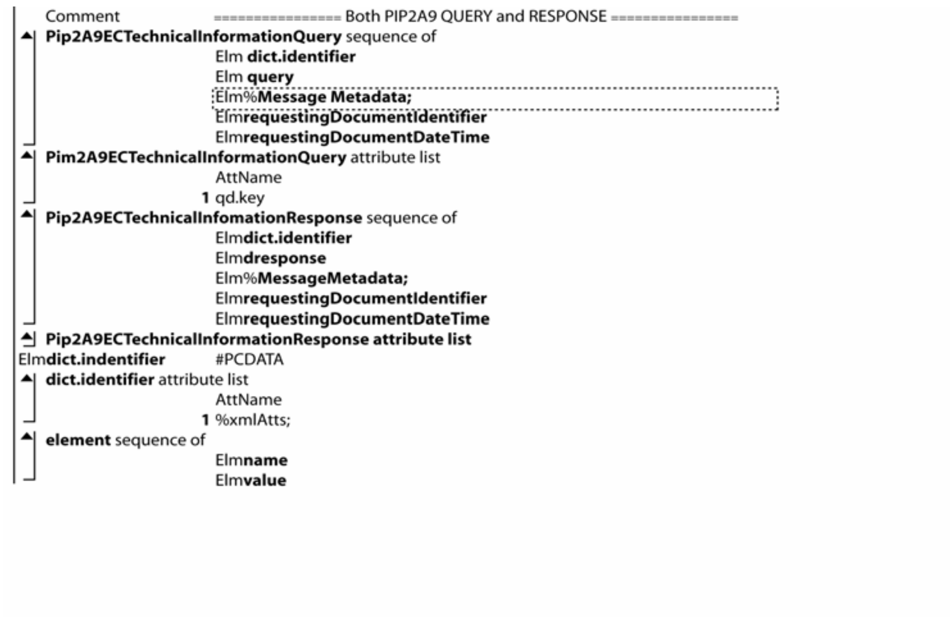


7 Technical information about our platform

This section gives more details about each part of the developed platform. First, we give some details about how the translation from the PIP2A9 BP to the Web services technology has been implemented. Next, both client and server sides are detailed with their technical parts and how they interact. Finally, the general architecture of the ebXML registry implementation is presented with its two associated access methods.

The use of the PIP2A9 allows us to obtain an homogenous environment when the mapping is set up. Indeed, this BP uses XML documents format during exchange: one document for the request and another one for the response. The general DTD request document is described in figure 7 as a grid representation extracted from XMLSpy 2004.

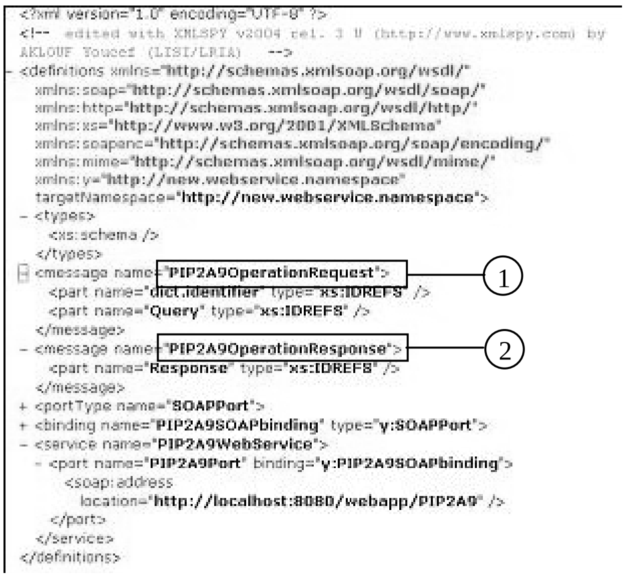
Figure 7 PIP2A9 DTD.



```
Comment                    =============== Both PIP2A9 QUERY and RESPONSE ===============
Pip2A9ECTechnicalInformationQuery sequence of
                           Elm dict.identifier
                           Elm query
                           Elm%Message Metadata;
                           ElmrequestingDocumentIdentifier
                           ElmrequestingDocumentDateTime
Pim2A9ECTechnicalInformationQuery attribute list
                           AttName
                         1 qd.key
Pip2A9ECTechnicalInfomationResponse sequence of
                           Elmdict.identifier
                           Elmdresponse
                           Elm%MessageMetadata;
                           ElmrequestingDocumentIdentifier
                           ElmrequestingDocumentDateTime
Pip2A9ECTechnicalInformationResponse attribute list
Elmdict.indentifier        #PCDATA
dict.identifier attribute list
                           AttName
                         1 %xmlAtts;
element sequence of
                           Elmname
                           Elmvalue
```

As stated previously, our Web service [9,16] is developed in Java J2EE platform. There are two possible ways to implement a Web service. The first one is by programming directly the service classes and interactions, and the second one is by creating a WSDL document and then, by generating the skeleton classes. In our case, the second method has been put into practice and the implementation of the classes issued from the skeleton with our specific code has been performed. Figure 8 shows a part of the WSDL document that defines our PIP2A9 Web service.

Figure 8 PIP2A9 WSDL Document



```
<?xml version="1.0" encoding="UTF-8" ?>
<!--  edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by
AKLOUF Youcef (LISI/LRIA)   -->
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:y="http://new.webservice.namespace"
    targetNamespace="http://new.webservice.namespace">
  - <types>
      <xs:schema />
    </types>
  - <message name="PIP2A9OperationRequest">
      <part name="dict.identifier" type="xs:IDREFS" />
      <part name="Query" type="xs:IDREFS" />
    </message>
  - <message name="PIP2A9OperationResponse">
      <part name="Response" type="xs:IDREFS" />
    </message>
  + <portType name="SOAPPort">
  + <binding name="PIP2A9SOAPbinding" type="y:SOAPPort">
  - <service name="PIP2A9WebService">
    - <port name="PIP2A9Port" binding="y:PIP2A9SOAPbinding">
        <soap:address
          location="http://localhost:8080/webapp/PIP2A9" />
      </port>
    </service>
</definitions>
```

The WSDL document is created by XMLSpy 2004. It represents the PIP2A9 which contains several methods with their parameters. Among these methods, the two basic ones are: a request (1 on fig 8) and response (2 on fig 8) methods. Among their important parameters, we find, the parameter representing the link to the document containing the payload. As a consequence, our service can be used with any dictionary and for any payload. There is just a need to add the correct link to our document representing this payload. This possibility ensures orthogonality between the BP layer and the content layer already addressed in [30].

Each service is composed of both client and server sides, the server contain the implementation classes of the service. Next, an overview of the behaviour of our system from both server and client sides is given.

7.1 Server Side

One of the important parts of our architecture is the server part. This part stores data related to components and product catalogues. It allows the possibility to retrieve these data when queried. The representation of such data in a classical database is not well adapted to our approach and to the exchange environment required in B2B systems. Indeed, it is necessary to store not only the data but also the description (ontology) of these data in order to allow a user to discover which data match the description he provides while querying. Therefore, a new database model, namely OBDB (Ontology Based Data Base), has been created in our laboratory [31]. This new database model allows the representation of the ontology of an application domain (exchange). Among the provided functions of this model, one allows referencing any kind of other databases via an universal identification mechanism. It uses the query language presented in next section. The encoded conceptual model and the logical Database model together with its instances stored in the same database. This representation is useful for our proposal since it increases interoperability and reduces conflict between several information sources and catalogues providers. However, this interoperability is made possible when the ontology model is shared between the partners involved in the exchange. Finally, notice that several tools have been developed in order to manage and visualise catalogues Section 5.3 reviews some of these tools.

The differents tasks that the server executes are as follow:
Once the server side (supplier) receives the request as an XML document sent by the customer, the different following functions are triggered:

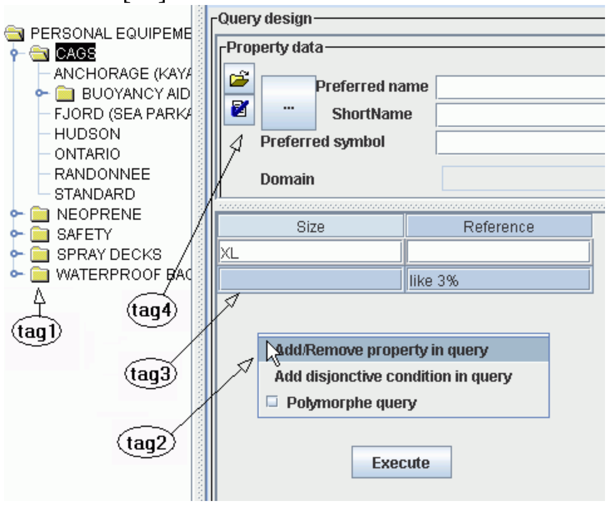
- extraction of information from the request in order to get homogeneous environment and a compatibility with the local system;
- translation of the content of the query to the internal (backend) representation. The query is processed on a local system with the local databases or dictionaries. Then, the result is presented in a local format as a response;
- translation of the result into an XML format and creation of the response document. This document is sent via the corresponding Web services;
- when this document is sent, the customer performs the symmetric task accompanied by analyses and interpretations of the results.

7.2 Client side

Since we use in our platform, the new database model (OBDB) cited above, and in order to manage data, a new query language is required. In the same team at LISI laboratory, a new query language is under development. It is named Ontology Query Language (OntoQL) [32] and a first preliminary version actually run on several OBDB implementations. Among, these applications of OntoQL, we find the usage we propose in our platform. Moreover, in our architecture, when the customer tries to select products, he must create a query to retrieve the required product. This query is received and transmitted by our web service. The Web service we developed supports this kind of query. To simplify the query creation for the consumer, PLIBEditor has been extended in order to support a user freindly querying user interface. This extension allows a user to interactively create the request. Figure 9 gives an overview of this interface where:

- Tag1 represents the ontology hierarchy for choice (classes);
- Tag2 represents a menu for adding and removing properties for the selected class;
- Tag3 is a table created based on the selected properties;
- Tag4 indicates if the query is a polymorphic one.

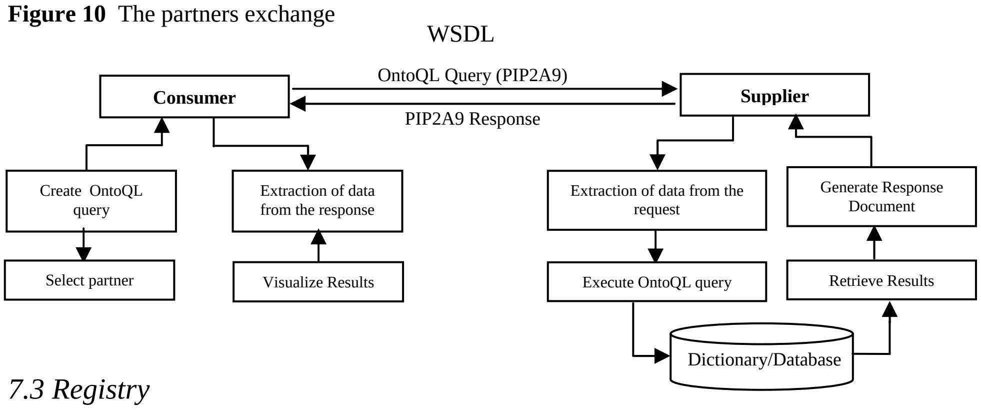
Figure 9: OntoQL interface [32]



For more details and comprehension, the scenario on the client side is described as follows.

An interface creates the skeleton of the service implemented using the J2EE platform. It represents a set of classes implementing the interface of the Web service functions. This interface is responsible of displaying the results in a specific format according to the specific desire of the customer. In our case, PLIBEditor makes it possible to both create the OntoQL query and to display the results when returned from the server side. Notice that the returned content is specific to the OBDB model: a set of files representing the products in a PLIB defined format (EXPRESS format). For other formats, the customer either creates a specific interface or uses its own proprietary existing applications for data visualisation. Figure 10 summarises the different tasks performed between partners (client and server) during exchange using the PIP2A9 Web service.

Figure 10 The partners exchange



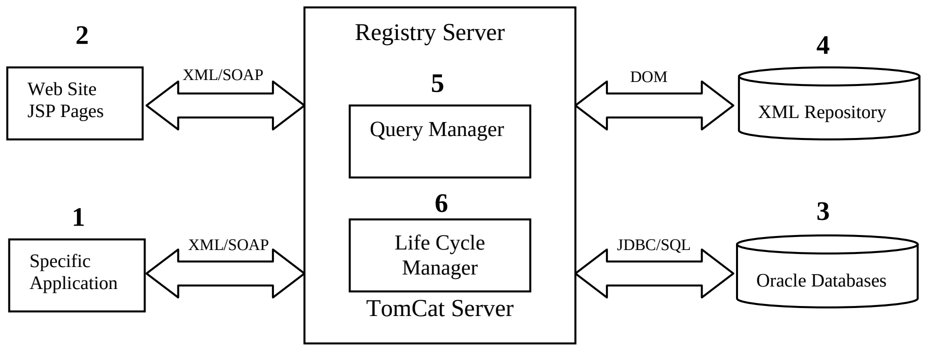
7.3 Registry

Another significant component of our architecture is the register. It enables us to identify and to locate the suppliers and partners, in order to establish connections between them. A registry based on the registry model defined by the ebXML Consortium has been developed. It is composed of two core specifications: the registry information model [27] and the registry specification [29]. Our implementation [33] was realised as follows.

Access to this register is made by two different ways. First, an access by a specific application (1) downloaded and installed by partners who want to use the register. In this case, only the subscriber can access to the registry. Second, the access is achieved on the web JSP pages (2) i.e a free access offered on the Internet. The registry administrator maintains a database which stores different kinds of data and metadata given by partners (3), CPP, CPA, XML documents (4), XML schemas, URLs links to Web services etc. Moreover, this kind of access provides a dynamic and a flexible search mechanism allowing finding and retrieving any type and any format of objects on the registry. The implementation provides these two accesses to the client allowing him to freely use and choose any kind of these two access methods. The registry contains two kinds of modules. The first is the query manager (5) which is responsible of managing the query received from the user, and the second is the Life Cycle Manager (6)which is responsible of managing objects stored in the registry since in the ebxml registry specification all information are represented as objects.

Figure 11 shows the general architecture of the registry:

Figure 11 The Registry architecture



8 Use of the model for PLM

The developed approach is useful during the product life cycle. Let us review three main applications of our proposal.

The first usage of this proposal is related to the choice of components that may be inserted in a given part by some CAD system designer. Indeed, when the first partner uses a CAD system to design a given part and when the second one is/are components supplier(s), this approach can be used to choose and retrieve a desired component to be inserted in a given part or product under design and/or manufacture. The designer uses this B2B exchange process to select, retrieve and send purchase order for a part, while the selected and agreed supplier sends the selected component together with an invoice.

In this case, the selection process is decomposed into a sequence of query/answer that helps the designer in choosing, step by step, the adequate component i.e. the component that better fulfills his requirements, for his design. This sequence defines a useful trace i.e. a workflow that may be stored by the designer for other purpose like maintenance as detailed below.

Before detailing the application for maintenance, recall that PLIB ontologies may be accompanied by libraries of components that store not only component descriptions but also several functional models (e.g. geometry in standard format). These functional models may also be selected, retrieved by the designer and may be also interpreted by the CAD system he uses. Indeed, PLIB provides with connections to the STEP standard for products and allows to make reference to STEP descriptions that may be interpreted by the CAD system in such interpreters are provided.

The second application is related to maintenance. Let us consider the following scenario. A designer searches for a 10 KW power electric engine. He sends the corresponding query through the exchange process defined in this paper, and after a sequence of query/answer, he finds only 10 KW power electric engines. He chooses this engine instead of 10 KW one.

After a while, this engine may be replaced for maintenance reasons. The maintenance manager makes a query to find such an engine and finds 12 KW power electric engines only. Thanks to the fact that the original query/answer trace is stored, the maintenance manager gets the knowledge that the originally asked power is 10KW and therefore, he is able to replace the current engine by a 12 KW power electric engine. This B2B exchange process defines a relevant workflow, carried by the query/answer trace, when stored, provides relevant knowledge to the maintenance manager.

9 Conclusion

e-business applications are adopting standards and initiatives to allow interoperation and interchange of information between information systems. Ontologies aim at providing a shared machine-readable view of domain knowledge, allowing information sharing for heterogeneous systems. The Web service technology allows developers to explicitly define BPs and interact with partners in a Web environment.

In this paper, we have shown how this technology is set up in B2B e-commerce in the particular case of developing RosettaNet PIPs with PLIB dictionary based on the classical layered architecture of e-commerce. As a result, an integrated and a simplified architecture is obtained. The use of PLIB provides us a uniform representation of dictionaries in a common shared ontology. Moreover, we have shown how B2B layered architecture can be abstracted and represented by three main layers. These three layers define the major concepts required for exchange between companies. Furthermore, the paper has shown the need for having a common shared ontology for exchange between partners in a B2B area. We have developed a Web service as support for business processes which was integrated in the proposed layered architecture in order to allow independence and orthogonality between them. Moreover, we overviewed the operational details used to implement our platform. Finally, we presented the application of the developed architecture to the PLM domain . In the future, we plan to scale up the proposed architecture and approach by addressing the study of the other business protocols from both RosettaNet and ebXML and investigate the possibility to define PLIB ontologies for BP's as well.

References

1 Aberdeen Group, (2001). 'The Fast Track to Positive e-Business ROI', *Trading Community Management, June.*

2 BIC. (2001). XML Convergence Workgroup: High-Level Conceptual Model for B2B Integration, *Business Internet Consortium ,Version: 1.0, 5 October.*

3 Coad, P. & Yourdon, E. (1992). Object-Oriented Analysis, *Prentice Hall, Englewood Cliffs, N.J.*

4 ebXML. (2001). ebXML technical architecture specification, *11May, www.ebxml.org*

5 ebXML. (2001). Business Process Team 10 May 2001 Business Process Analysis Worksheets and Guidelines. *Procedures for Developing Business Processes in ebXML v1.0, July.*

6 Gartner Group. (2000). OAGI: Fostering Standards Evolution or Revolution?. *A Gartner Advisory Research Note 14 December.*

7 Peltz, C. and Murray, J., (2004) 'Using XML schemas Effectively in WSDL Design', *Software Development Conference and Expo,* March.

8 Glushko, R. J. Tenenbaum, J. M. & Meltzer, B. (1999): An XML Framework for Agent-based E-commerce, *Communications of the ACM,* 42(3).

9 Heather, K. (2001). Web Service Conceptual Architecture 1.0, May 2001, *IBM Software Group.*

10 ISO 10303-11. (1994). Industrial automation systems: and integration — Product data representation and exchange – *Part 11: Description Methods: The EXPRESS language reference manual.*

11 ISO 10303-22. (1997). Industrial automation systems and integration. – *Product data representation and exchange– Part 22: Implementation methods: Standard Data Acess Interface.*

12 ISO 13584-42. (1998). Industrial automation systems and integration. *Parts library– Methodology for structuring Parts Families, ISO, Genev*

13 Jasper, R. & Uschold, M. (1999). A framework for understanding and classifying ontology applications. *In B. Gaines, R. Cremer, and M. Musen, editors, Proceedings 12th Int. Workshop on Knowledge Acquisition, Modelling, and Management KAW'99 (16-21 October 1999, Banff, Alberta, Canada), volume I, pages 4–9–1— 4–9–20, Calgary. University of Calgary, SRDG Publications*

14 McGuinness, D. L.(1999). Ontologies for Electronic Commerce. In *Proceedings of the AAAI '99 Artificial Intelligence for Electronic Commerce Workshop* , Orlando, Florida, July.

15 Pierra, G. **(**2000). Représentation et échange de données techniques, *Mec. Ind., 1, pp.397-400.*

16 Randy, E. Hall. (2001). "W3C Web Service Position Paper from Intel", *W3C Workshop, April 12.*

17 RosettaNet. (2001). RosettaNet Architecture Conceptual Model, *July.*

18 RosettaNet. (2001), RosettaNet Implementation Framework: Core Specification *Version: Validated 02.00.00 13 July.*

19 RosettaNet. (2001). Specification: PIP Specification Cluster 2: Product Information, Segment A: Preparation for Distribution, PIP2A9: Query Technical Product Information, *Validated 01.01.01, 01 november.*

20 Schenck, D. & Wilson, P. (1994). Information Modelling The EXPRESS Way, *Oxford University Press.*

21 Westarp, F. V. Weitzel, T. Buxmann, P. & König, W.(1999). The Status Quo and the Future of EDI - Results of an Empirical Study. In *Proceedings of the European Conference on Information Systems (ECIS'99).*

22 Spek, A.W.A. (2002) 'Designing Web Services in a Business Context'. *Master Thesis,* University of Van Tilburg. Center of applied research, September.

23 Tellmann, R., and Meadche, A. (2003), 'Analysis of B2B standard and Systems', *SWWS, Semantic Web Enabled Web Services.*

24 Kappel, G. and Kramler, G. (2003), 'Comparing WSDL-Based and ebXML based Approaches for B2B Protocol Specification'. Martin Bernauer, Business Informatics Group. Vienna University of Technology, Austria

25 Brydon, S., Murray, G., Ramachandran, V., Singh, I., Streans, B., and Violleau, T. (2004) 'Designing Web Services with the J2EE™ 1.4 Platform: JAX-RPC, SOAP, and XML Technologies'. *Sun Mictosystems,* January

26 Trastour, D., Preist, C., and Colemann, D.(2003) 'Using Semantic Web Technology To Enhance Current Business-to-Business Integration Approaches'. *Proceeding of the Seventh IEEE International Entreprise Distributed Object Computing Conference,* (EDOC'03).

27 ebXML Registry Information Model Schema (2002),

http://www.ebxml.org/specs/ebRIM.pdf

28 Hunter, J. and Grawford, W. (2002). Servlet java guide du programmeur, Edition

O’REILLY.

29 ebXML Registry Services Specification Version 2.1 (2002), http://www.ebxml.org/specs/ebRS.pdf

30 Aklouf, Y., Pierra, G., Ait Ameur, Y., and Drias, H. (2005) ‘PLIB Ontology: A Mature Solution for Products Charaterization in B2B Electronic Commerce’. Special Issue: E-Business Standards. International Journal Of IT Standards and Standardization Research; Vol 3 No.2. 2005. IRMA& IGP publishing.

31 Pierra G., Dehainsala H., Ait-Ameur Y., Bellatreche L. (2005) ‘Base de Données a Base Ontologique : principes et mise en oeuvre.", *To appear in Ingénierie des Systèmes d'Information (ISI).*

32 Jean S., Pierra G., Ait-Ameur Y. (2005) OntoQL: an exploitation language for OBDBs VLDB PhD Workshop, 29 Août 2005.

33 *Takheroubt M., Boudjemai R. (2004). Etude et Mise en œuvre du registre ebXML dans une plate de commerce B2B. Master Thesis. USTHB Algeria.*

34 S. Brydon, , G. Murray, V. Ramachandran, I. Singh, B. Streans, and T.Violleau, (2004) ‘Designing Web Services with the J2EE™ 1.4 Platform: JAX-RPC, SOAP, and XML Technologies’. *Sun Mictosystems,* January.