ebXML CPP/CPA 문서 편집기 설계 및 구현

정문영*, 오수영*, 조현성*, 조현규*

Design and Implementation of an ebXML CPP/CPA Editor

Moonyoung Chung, Soo-Young Oh, HyeonSung Cho, Hyun-Kyu Cho

Abstract

B2C e-commerce is now sufficiently stable. Judging from its success, we can expect B2B to similarly improve business processes for a better return on investment. To achieve B2B e-commerce, many consortia have been trying to develop a B2B framework. As a result, the ebXML is started as an international standard for B2B e-commerce based on XML.

The exchange of information between two parties requires each party to know the other party's supported business collaborations, the other party's role in the business collaboration, and the technology details about how the other party sends and receive messages. In some cases, it is necessary for the two parties to reach agreement on some of the details. The ebXML CPP and CPA specification defines how to describe these.

In this paper, we'll describe our design and implementation of an ebXML CPP and CPA editor which enables each party to create and edit his profiles and which automatically composites two CPPs. Although it is possible to create and edit CPP and CPA documents with general-purpose XML editor, it is not easy to create and edit CPP and CPA documents with a general-purpose XML editor. Moreover, the detailed procedures for CPA formation from CPPs are currently not provided in the specification. Therefore, we propose two approaches for a CPA formation in addition to easy-to-use user interface.

***Key Word** : ebXML, CPP, CPA, B2B, e-commerce*

*한국전자통신연구원 컴퓨터소프트웨어기술연구소 전자상거래연구부 전자거래연구팀

1. Introduction

Electronic commerce lets people purchase goods and exchange information on business transactions online. E-commerce gives companies improved efficiency and reliability of business processes through transaction automation. There are two major types of e-commerce: business-to-customer (B2C) and business-to-business (B2B). While B2C e-commerce was relatively easy to achieve, that's not the case with B2B.

For over 25 years Electronic Data Interchange (EDI) has given companies the prospect of eliminating paper documents, reducing costs, and improving efficiency by exchanging business information in electronic form. But, only large companies are able to afford to implement it, and much EDI-enabled e-commerce is centered around a dominant enterprise that imposes proprietary integration approaches on its trading partners.

In the last few years, the eXtensible Markup Language (XML) is a simple, portable, and human-readable syntax. XML provides a format particularly convenient to exchange data and is expected to play a central role in the new generation of Web applications, such as electronic commerce and corporate portals. Officially recommended by the World Wide Web Consortium (W3C) in early 1998, XML addresses the issues many earlier proprietary solutions tried to resolve to enhance the Hypertext Markup Language. Web developers use HTML to define a document's presentation format with a predefined set of tags. In contrast, designers use XML, to create customized tags by describing the document's structure.

Since a document in XML format contains a structure, it is easy to transform structured business data into XML and vice versa. Users can search XML data for an item by looking for specific tags in a structured document.

XML documents contain logical units called elements. With an XML parsers, users can read XML documents and access their contents and structures. Three companion entities complement XML: Document Type Definition (DTD), Extensible Stylesheet Language (XSL), and Extensible Link Language (XLL), which specify the document's layout, style sheet, and dynamic links, respectively. An XML DTD defines a document's logical structure.

The ebXML specifications provide a framework in which EDI's substantial investments in business processes can be preserved in an architecture that exploits XML's new technical capabilities.

Electronic Business eXtensible Markup Language (ebXML) is an international initiative established by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the Advancement of Structured Information Standards (OASIS). The purpose of the ebXML initiative is to research and identify the technical basis upon which the global implementation of XML can be standardized. The goal is to provide an XML-based open technical framework to enable XML to be utilized in a consistent and uniform manner for the exchange of electronic business data [1].

According to ebXML specifications, the message-exchange capabilities of a party may be described by a Collaboration Protocol Profile (CPP) and the message-exchange agreement between two

parties may be described by a Collaboration Protocol Agreement (CPA). Included in the CPP and CPA are details of transport, messaging, security constraints, and bindings to a business process specification document that contains the definition of the interactions between the two parties while engaging in a specified electronic business collaboration.

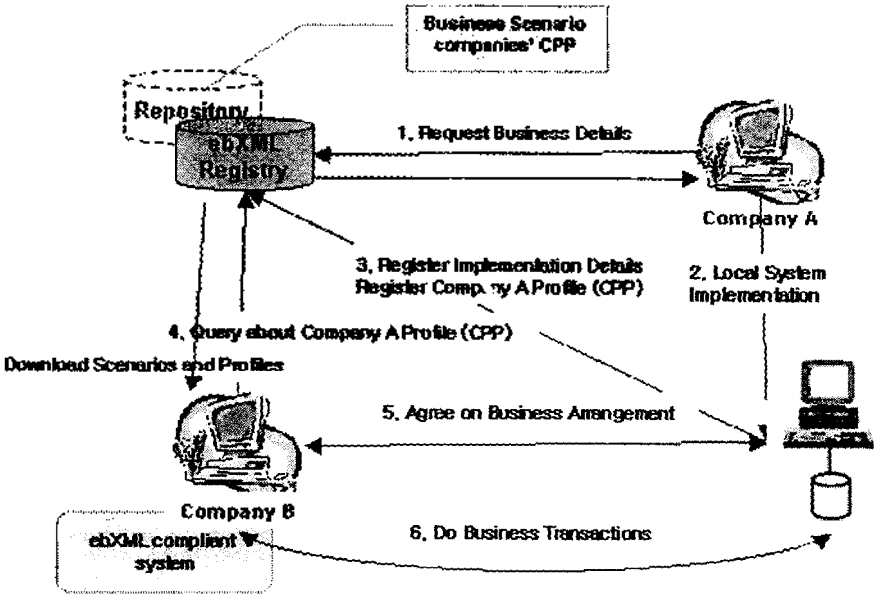
In this paper, we introduce the ebXML CPP and CPA specifications and how CPP and CPA work in the e-commerce framework, and we also describe our design and implementation of an ebXML CPP/CPA editor conforming to the ebXML CPP/CPA specification. In addition, we propose two approaches for CPA formation from two CPPs: one uses XSLT and other uses JAXB.

We begin in Section 2 with introduction of an overview of an ebXML system, ebXML CPP and CPA, XML and XSLT, and JAXB. In Section 3, we describe the design of our CPP/CPA editor in detail and show how to implement our editor in section 4. We present our conclusion and future work in Section 5.

2. Preliminaries

2.1. ebXML System Overview

<Figure 1> shows a high-level use case scenario for two trading partners, configuring and engaging in a simple business transaction and interchange. In the <Figure 1>, company A has become aware of an ebXML registry that is accessible on the Internet (step 1). After reviewing the contents of the ebXML registry, company A builds and deploys its own ebXML compliant application (step 2). Company A then submits its own business profile information (including CPP) to the ebXML registry (step 3). The business profile submitted to the ebXML registry describes the company's ebXML capabilities and constraints, as well as its supported



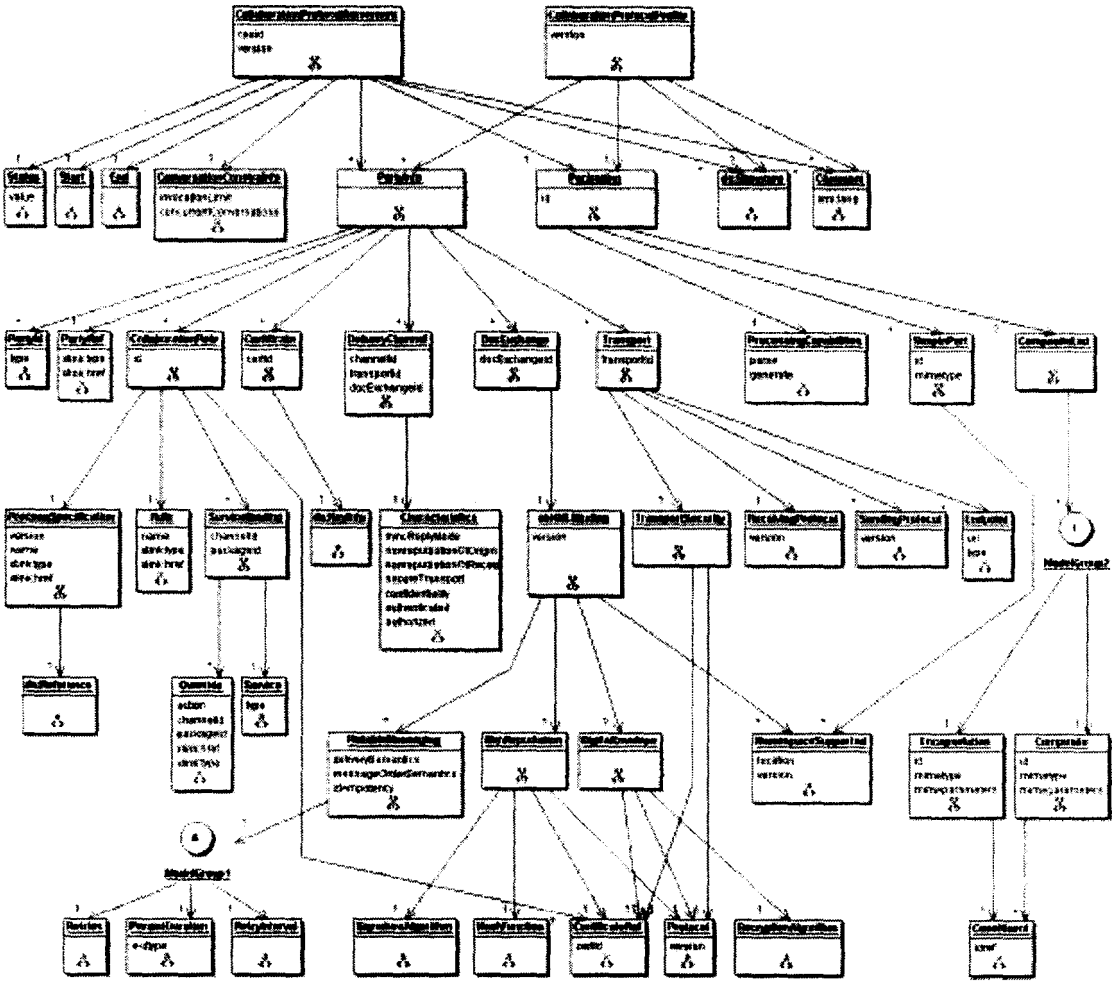
<Figure 1> Overview of the interaction of two companies conducting e-business using ebXML

business scenarios. Company B discovers the business scenarios supported by company A in the ebXML registry (step 4). Company B sends a request to Company A stating that they would like to engage in a business scenario using ebXML (step 5). The proposed business arrangement outlines the mutually agreed upon business scenarios and specific agreements. The business arrangement also contains information pertaining to the messaging requirements for transactions to take place, contingency plans, and security-related requirements. Company A then accepts the business agreement. Company A and B are now ready to engage in e-commerce using ebXML. (step 6)

2.2 CPP and CPA

To facilitate the process of conducting e-commerce, 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. The CPP defines a party's message-exchange capabilities and the business collaborations that it supports. The CPA defines the way two parties will interact in performing the chosen business collaboration. The CPA is derived from the intersection of two or more CPPs.

CPPs and CPAs are XML documents. The



<Figure 2> The set of elements in the CPP and CPA

CPP/CPA specification defines the markup language vocabulary for creating electronic CPPs and CPAs. The <Figure 2> represents the relationship among elements in the CPP and CPA.

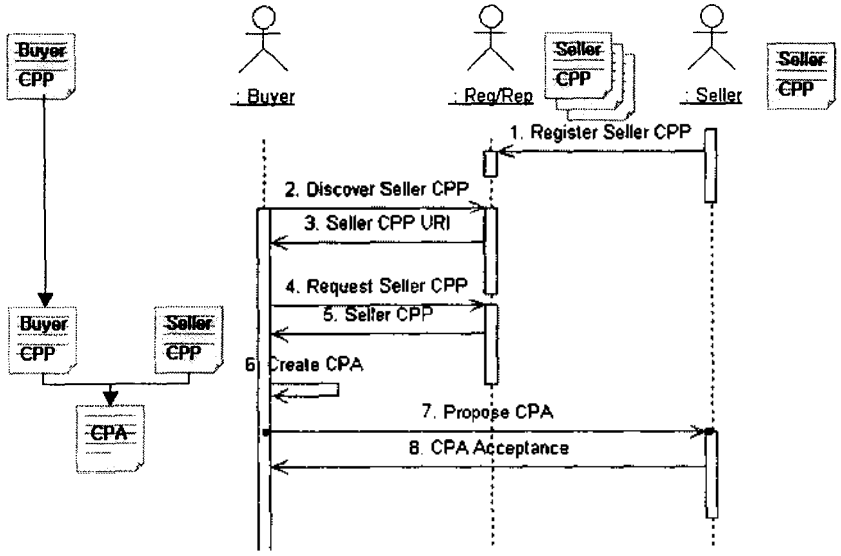
The CPP describes the specific capabilities that a trading partner supports as well as the service interface requirements that need to be met in order to exchange business documents with that trading partner. The CPP contains essential information about the trading partner including, but not limited to: contact information, industry classification, supported business processes, interface requirements and messaging service requirements. CPP may also contain security and other implementation specific details.

The CPA is a document that represents the intersection of two CPPs and is mutually agreed upon by both trading partners who wish to conduct e-commerce using ebXML. A CPA describes: the messaging service and the business process requirements that are agreed upon by two or more trading partners.

The <Figure 2> shows the relationships among elements those are included in the ebXML CPP and CPA. In this Figure, each rectangle represents an element in the CPP or CPA, and a word in the first line of the each element is an element name while words below are attributes.

A CPP shall be capable of referencing one or more business processes supported by the trading partner owning the CPP instance. The CPP shall reference the roles within a business process that the user is capable of assuming. An example of a role could be the notion of a "Seller" and "Buyer" within a "Purchasing" business process. The CPP shall be capable of being stored and retrieved from ebXML registry mechanism.

Similarly, a CPA must reference to a specific business process and the interaction requirements needed to execute that business process, and a CPA may be stored in a registry mechanism, hence an implied ability to be stored and retrieved is present.



<Figure 3> Trading Partner Formation Conversation

2.3 How CPP and CPA works

In the e-commerce transactions, arrangements to conduct business are done during configuration time. This includes binding business processes and business documents to business collaborations by creating and registering CPP. From here, CPA is formed in which business can be conducted. Finally, these agreements provide the basis for configuring the runtime systems.

The <Figure 3> describes this procedure. First, any party may register its CPPs to an ebXML *Registry*. *Seller* registers his own CPP to the *Registry* in the Step 1 of the Figure 3. Then *Buyer* discovers trading partner (*Seller*) by searching in the *Registry* and downloads *Seller*'s CPP to *Buyer*'s server (step 2-5). Next, *Buyer* creates CPA and sends CPA to *Seller*. After *Seller* and *Buyer* negotiate and store identical copies of the completed CPA as a document in both servers. This process is done manually or automatically. Finally, *Seller* and *Buyer* configure their run-time systems with the information in the CPA. Now, *Seller* and *Buyer* do business transaction under the new CPA.

2.4 XML and XSL

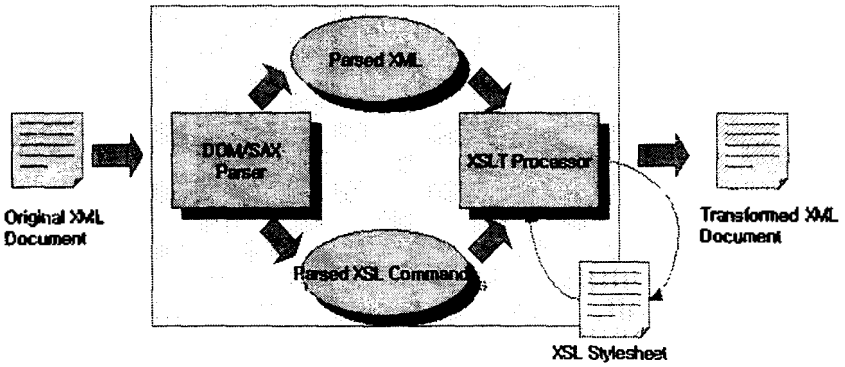
The Extensible Markup Language (XML) is a subset of SGML that is the universal format for structured documents and data on the Web.

XSL is a language for expressing style sheets. It consists of three parts: XSL Transformations (XSLT): a language for transforming XML documents, the XML Path Language (XPath), an expression language used by XSLT to access or refer to parts of an XML document. XSLT originally intended to perform complex styling operations, like the generation of tables of contents and indexes, it is now used as a general purpose XML processing language. XSLT is thus widely used for purposes other than XSL, like generating HTML web pages from XML data. Styling requires a source XML documents, containing the information that the style sheet will display and the style sheet itself which describes how to display a document of a given type.

The <Figure 4> describes how to transform XML document.

2.5 The Java Architecture for XML Binding

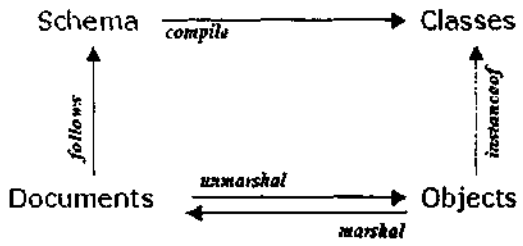
JAXB provides a way to create a two-way mapping between XML documents and Java objects. Given a schema, which specifies the structure of XML data, the JAXB compiler generates a set of Java classes containing all the code to parse XML documents based on the schema. An application that uses the generated classes can build a Java object tree representing an



<Figure 4> XSL Transformation

XML document, manipulate the content of the tree, and regenerate XML documents from the tree, all without requiring the developer to write complex parsing and processing code.

Schemas describe the structure and meaning of an XML document, in much the same way that a class describes an object in a program. To work with an XML document in a program, we would like to map its components directly to a set of objects that reflect the document's meaning according to its schema. This is achieved by compiling the schema into a set of derived classes



<Figure 5> Java Architecture for XML Binding

that handle all the details of marshalling and unmarshalling and also ensure that only valid documents will be produced and consumed.

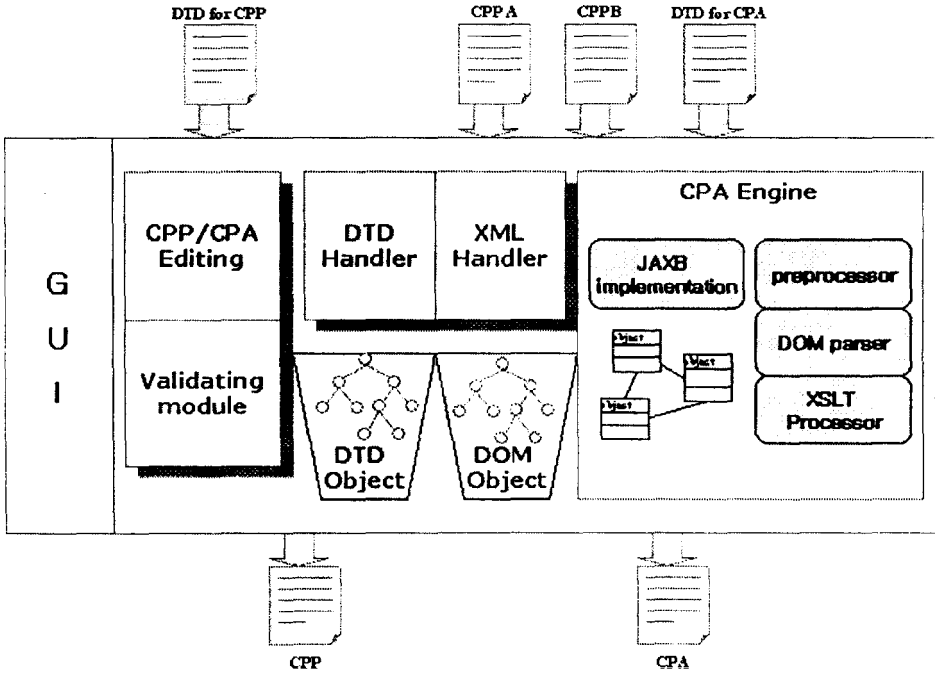
3 Design of an ebXML CPP/CPA Editor

3.1 System Design

Our system design is based on the MVC (Model-View-Controller) design pattern, which separates a software component into three distinct pieces: a model, a view, and a controller.

In our design, *Model* corresponds to the DTD object, DOM object and our own data structure to handle them. And *View* is mapped with the user interface. Finally, we designed several *controller* modules such as XML handler and DTD handler.

Since CPP and CPA are XML documents, our editor must also handle general XML document. In addition, CPP and CPA editor must provide special



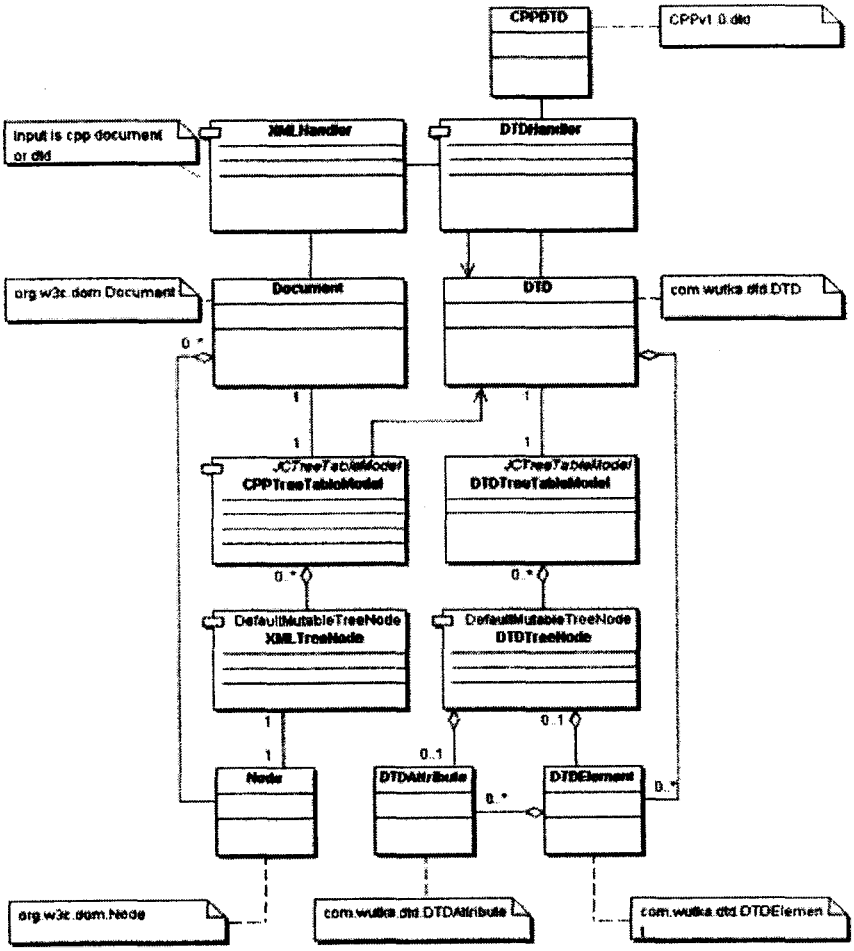
<Figure 6> Overview of the ebXML CPP/CPA Editor

functionalities to create and edit CPP and CPA document more easily and more efficiently. Therefore, we designed CPP and CPA handling modules as well as the common XML document handling modules and XML document validating modules. The <Figure 6> shows the overall system structure of the CPP/CPA editor.

Our system has three core modules: an XML handling module, CPP handling module, and a CPA generation module. CPA generation module is either using XSLT or JAXB, this is explained in detail in the next section.

An XML handling module is responsible to handle XML documents with the related DTD schemas those are provided by the ebXML CPP and CPA specification. Template CPP documents are also created by an XML handling module. This module has document validating functionality. The common data structure corresponding to XML documents is depicted in the class diagrams of the <Figure 7>. This data structure consists of two major parts: DOM object and DTD object. DTD object is generated from the CPP schema by *DTDParser* and DOM object is created from DTD.

Since CPP templates generated by our CPP editor don't reflect parties' profiles, each party need to edit several element according to their own business profiles. So, our editor provides editing



<Figure 7> Data Structure for CPP and CPA

functionalities. Actually, it is possible to create and edit CPP/CPA documents with general-purpose XML editor. It's important that data structure through the whole system is consistent. In this editor, consistency is also maintained by a CPP handling module.

Finally, a CPA generation module is responsible to form a CPA from two or more CPPs. These formation procedures are explained in the next section in detail.

3.2 Forming a CPA from Two CPPs

In this section, we discuss how to form CPA from CPPs. The detailed procedures for CPA formation are currently left for implementers. No normative specification is provided for algorithms for CPA formation in the specification. Therefore, algorithms for CPA formation can be different for different implementations.

In this paper, we propose two approaches to compose two CPPs: using XSLT and using JAXB.

3.2.1 Forming a CPA using XSLT

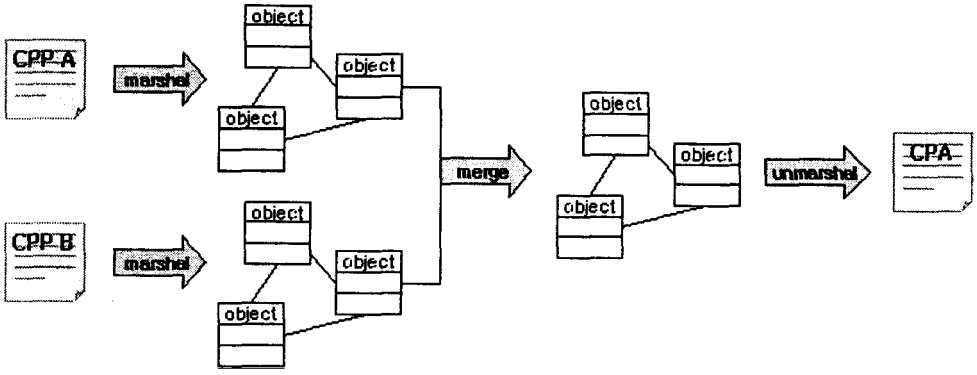
XSLT is an XML application that specifies rules by which one XML document is transformed into another. An XSLT processor compares the elements in an input XML document to the template's contents into an output tree.

We use an XML document composed of two CPP as input XML document and transform it with an XSLT style sheet - This step is required because XSLT processor is able to input just one XML document. If just joining two CPP, output XML document has two root elements, those are two <CollaborationProtocalDocument> elements. However XML document must have just one root element. Therefore, after assigned an arbitrary root element, output XML document is an input to the XSLT processor. Then output tree produced is a CPA template.

3.2.2 Forming a CPA using JAXB

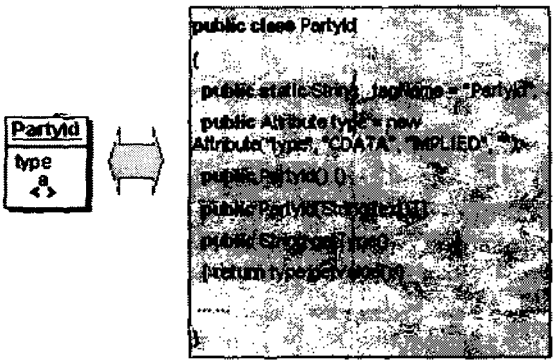
Now we show how to forming a CPA using JAXB. As shown in Section 2, JAXB provides a way to create two-way mapping between XML documents and Java objects.

First, we transform two CPP documents into Java objects. Then we compose Java objects of CPP in the element level according to the specification, and obtain Java objects of CPA. Finally, Java objects of CPA are transformed to a CPA document. <Figure 8> shows our CPA formation algorithm graphically.



<Figure 8> CPA Formation using JAXB

<Figure 9> shows the 1:1 mapping between PartyId element in the CPP and PartyId java object.



<Figure 9> An Example of Mapping between XML Element and Java Object

4 Implementation of an ebXML CPP/CPA Editor

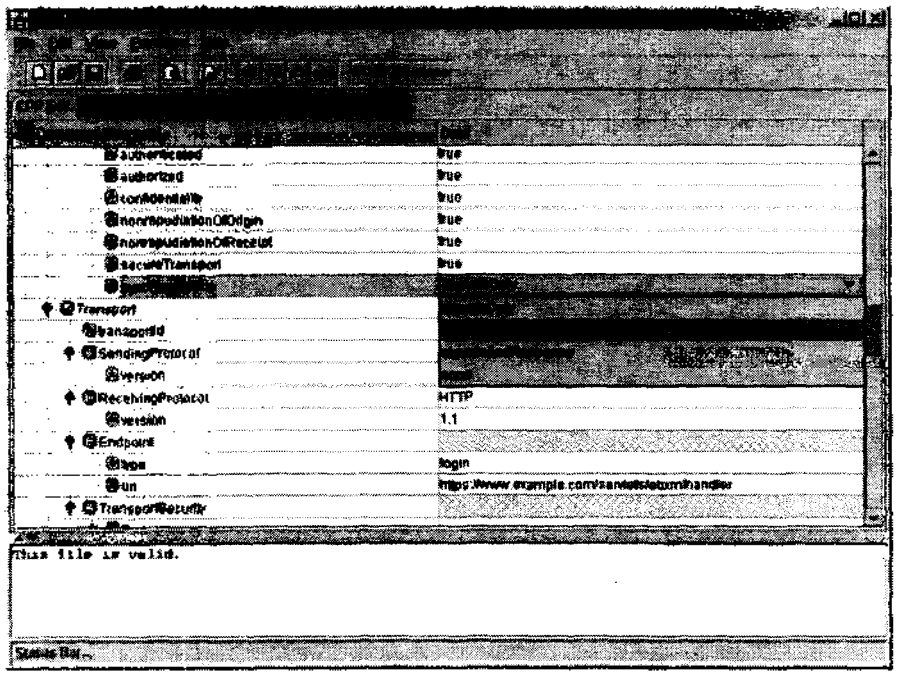
We implemented our ebXML CPP/CPA Editor with Apache XML parser and Wutka's DTD parser using Java programming language in the window platform.

General-purpose XML editor is not sufficient to create and edit CPP/CPA documents with general-purpose XML editor because CPP/CPA documents are complicated for users to create or edit CPP and CPA documents. Therefore our easy-to-use user interface gives a visual representation of the hierarchical nature of its structure, and it seems that it is composed of JTree and JTable in the JDK Swing components. It is very useful to represent hierarchical tree structure which nodes contain data. XML document is a good example.

In <Figure 10>, XML tree structure is shown in the left column and editable table is shown in the right column.

5 Conclusion

Business-to-Business transactions are growing in the e-commerce. Businesses conduct e-commerce transactions through standards such as electronic data interchange since the 1970s. EDI benefits large companies by greatly reducing the time and cost of



<Figure 10> User Interface of the CPP/CPA Editor

manual data processing. EDI is complex and difficult to implement, however. For small and medium businesses, the network infrastructure and software required to implement EDI are prohibitively expensive. So, ebXML is proposed by the UN/CEFACT and the OASIS with the goal to provide an XML-based open technical framework to enable XML to be utilized in a consistent and uniform manner for the exchange of electronic business data in application to application, application to human, and human to application environments.

In this paper, we described our design and implementation of a CPP/CPA document editor conforming to the ebXML specifications. CPP/CPA specification, a component of the suite of ebXML specifications, is about interoperability between two parties even though they may procure application software and run-time support software from different vendors.

Although it is possible to create and edit CPP/CPA documents with general-purpose XML editor, CPP/CPA documents are composed of so many elements and users are not able to create or edit CPP and CPA documents easily. Moreover, it's possible to automate most part of the procedures for CPA formation, but no practical algorithm is provided by the specifications. Therefore, we developed two algorithms for CPA formation: one used XSLT and the other used JAXB.

In the future, we will make this system to be used to register CPP/CPA document to the ebXML registry, and eventually to work together with the whole ebXML B2B framework. In addition to this, we will extend our system to be platform-independent.

Reference

[1] ebXML Technical Architecture Project Team, *ebXML Technical Architecture Specification*, v1.0.4, published 16 Feb., 2001

[2] ebXML Trading Partners Team, *Collaboration-Protocol Profile and Agreement Specification*, Version 1.0, published 11 May, 2001

[3] ebXML Transport, Routing, and Packaging Project Team, *Message Service Specification ebXML Transport, Routing & Packing*, Version 1.0, published 11 May, 2001

[4] *ebXML Proof-Of-Concept*, Technical Planning Document for End-to-End Demonstration in Vienna version, Working Draft, April 26, 2001

[5] ebXML Requirements Team, *ebXML Require - ments Specification*, Version 1.06, May 8, 2001

[6] ebXML Business Project Team, *ebXML Business Process Specification Schema*, Version 1.01, May 11, 2001

[7] Simon S.Y. Shim, Vishnu S. Pendyala, Meera Sundaram, and Jerry Z. Gao, *Business-to-Business E-Commerce Frameworks*, IEEE Computer, 2000

[8] World Wide Web Consortium, *Extensible Makrup Language (XML) 1.0 Specificatioln*, W3C Recommendation REC-xml-19980210, Feb. 1998; http://www.w3.org/TR/REC-xml.

[9] World Wide Web Consortium, *The Extensible Stylesheet Language*, July 10, 2001

[10] Mark Reinhold, *The Java Architecture for XML Binding (JAXB)*, May 30, 2001

[11] *The Java Architecture for XML Binding User's Guide*; http://java.sun.com

[12] Elliotte Rusty Harold and W. Scott Means, *XML in a Nutshell*, O'Reilly and Associates, Inc. January 2001.