AN EXTENSION TO A CORBA TRADER TO SUPPORT XML SERVICE DESCRIPTIONS

Twittie Senivongse and Wuttichai Nanekrangsan
Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand

Abstract Search functionality of a CORBA trader is restricted to search for service offers and assumes clients' knowledge of service types of those offers. It would be more flexible if the clients can also import other information, i.e. service types and interfaces, before trading for service offers, or conduct keyword search. With this requirement, making service descriptions into XML format can be helpful. This paper focuses on the trader extension that can transform service types and service offers within a CORBA trader into XML service descriptions, and vice versa. The transformation is based on our Document Type Definitions for service types and service offers. This transformer module can be used to create XML service descriptions that will enable flexible XML-based service discovery. The transformer also facilitates clients in viewing details of CORBA services from Web browsers and helps with exporting service descriptions to the trader.

Keywords: service discovery, XML, trader, CORBA

1. INTRODUCTION

CORBA Trader [1] (Trading Object Service) is one of the common services in CORBA [2] that serves as a directory, allowing service exporters (servers) to advertise their service type and service offer descriptions, and allowing service importers (clients) to discover service offers they desire. The trader is designed for trading for service offers, and thus its clients are assumed to know details about types and interfaces of those offers. We aim to provide a flexible service discovery service that can discover service information, i.e. service types, interface definitions, and service offers, with no assumption on clients' exact knowledge of the services and the clients can trade for information in a similar way they do with search engines [3]. With this objective, we use Extensible Markup Language (XML) [4] to represent service descriptions because its self-describing characteristic can contribute to more flexible search, and its accepted status as a data interchange medium will open a way for future integration of service information from several directory services. Our

experimental service discovery service obtains XML service descriptions from CORBA traders; these traders are extended with a module that can transform CORBA service descriptions into XML documents. This paper focuses on the architecture of this transformation module although the overview of the service discovery service will also be discussed in Section 5.1.

The transformation module can transform trader's service type descriptions, with interface definitions from the Interface Repository (IR) embedded, as well as service offer descriptions into XML documents, and vice versa. The transformation is based on our Document Type Definitions (DTDs) for service types and service offers. Since we are focusing more on flexible features for discovery of services than on the interchange of service descriptions among several directory services, we describe CORBA service descriptions by using our own simple DTDs rather than the standard XML Metadata Interchange (XMI) [5] in our experimental prototype. It is foreseen that service descriptions can be represented as XMI documents and used by the transformation module to overcome this limitation. Apart from being used in our service discovery service, the transformation module provides a convenient way to access CORBA service descriptions from other architecture like World Wide Web.

There are several efforts to describe component and service descriptions in XML. WebTrader [6] provides an infrastructure to handle Web-based service market where users can export and import services in XML. A DTD is defined for users to describe their services but it does not provide for all characteristics of CORBA service descriptions. Other works include Open Software Description (OSD) [7] that describes general software descriptions, and Deployable Software Description (DSD) [8] that uses XML to describe components for deployment management purpose; nevertheless, they are too general for describing CORBA services.

Section 2 of this paper gives the explanation of service descriptions that can be found in the trader and IR followed by our proposed DTDs that will be used to describe them. Section 3 discusses the extension to the trader to facilitate CORBA/XML transformation. Our prototype implementation is explained in Section 4 and Section 5 discusses possible use of the transformation module. Section 6 summarises the paper with future work.

2. SERVICE DESCRIPTIONS IN CORBA

Service descriptions within a CORBA trader can be divided into two categories – service type and service offer. A service type is an abstract definition of a service and it is, therefore, a template for advertising service offers. A service offer is the information describing a particular instance of a service that conforms to a service type. We define two straightforward DTDs for XML service descriptions, i.e. service type DTD and service offer DTD.

2.1. Service type and service type DTD

A service type description within a trader comprises service type name, service interface, base service types from which the service type inherits, and a set of service properties. This information is trader-owned and stored in the Service Type Repository module of the trader. It is described by the following BNF:

```
Service <ServiceTypeName> [:<BaseServiceTypeName>
[,<BaseServiceTypeName>]*]{
  interface <InterfaceTypeName>;
  [[mandatory][readonly] property <IDLType> <PropertyName>;]*
}
```

InterfaceTypeName above is the link to the interface definition expressed in OMG IDL and stored in the IR. With this interface definition, the service type description is augmented with information such as the interface name, the set of attributes, and the set of operations that this service can respond to. Hence, our service type DTD that constrains XML service type descriptions is composed of two main parts: the DTD for the interface definition from the IR and the DTD for the service type information from the trader.

Table 1 shows our service type DTD. The design principle is to capture as many CORBA service type characteristics as possible. The *Interface* element in the service type DTD describes the computational signature of the interface definition from the IR, while the *TraderServiceType* element shows the service type description from the trader. The *Interface* element consists of the interface identifier, base interfaces represented as structured graph (adapted from [9]), list of constants, list of attributes, and list of operations. The *TraderServiceType* element comprises the interface id that corresponds to an interface definition in the IR, base service types, and a list of property templates. Note that the DTD does not yet support a full description of any user-defined type; only type name is supported.

2.2. Service offer and service offer DTD

Service offers are stored within the Offer List of the trader, each describing the service type name, name-value pairs of service properties, and the object reference (IOR) used to locate the service instance. The service offer DTD is simple, as shown in Table 2, capturing the service type name, zero or more property values and the reference to the offer instance. The *DynamicPropEval* elements describe exported dynamic properties.

Table 1. Service type DTD

```
<!ELEMENT ServiceTypeDescription (Interface?, TraderServiceType)>
<!ELEMENT Interface (BaseInterfaces?, Constant*, Attribute*, Operation*)>
<!ATTLIST Interface Id CDATA #REQUIRED Name CDATA #REQUIRED
                    Version CDATA #REQUIRED>
<!ELEMENT BaseInterfaces (BaseInterface*, Link*)>
<!ELEMENT BaseInterface EMPTY>
<!ATTLIST BaseInterface Id CDATA #REQUIRED Name CDATA #REQUIRED>
<!ELEMENT Link EMPTY>
<!ATTLIST Link Source CDATA #REQUIRED Dest CDATA #REQUIRED>
<!ELEMENT Constant EMPTY>
<!ATTLIST Constant Id CDATA #REQUIRED Name CDATA #REQUIRED
                   Version CDATA #REQUIRED Type CDATA #REQUIRED
                   Value CDATA #REQUIRED Derived (YES | NO) "NO">
<!ELEMENT Attribute EMPTY>
<!ATTLIST Attribute Id CDATA #REQUIRED Name CDATA #REQUIRED
                    Version CDATA #REQUIRED Type CDATA #REQUIRED
                    Mode (NORMAL | READONLY) "NORMAL"
                    Derived (YES | NO) "NO">
<!ELEMENT Operation (Parameter*, Exception*, Context*)>
<!ATTLIST Operation Id CDATA #REQUIRED Name CDATA #REQUIRED
                    Version CDATA #REQUIRED Type CDATA #REQUIRED
                    Mode (NORMAL | ONEWAY) "NORMAL"
                    Derived (YES | NO) "NO">
<!ELEMENT Parameter EMPTY>
<!ATTLIST Parameter Name CDATA #REQUIRED Type CDATA #REQUIRED
                     Mode (IN | OUT | INOUT) "IN">
<!ELEMENT Exception (Member)*>
<!ATTLIST Exception Id CDATA #REQUIRED Name CDATA #REQUIRED
                    Version CDATA #REQUIRED Derived (YES | NO) "NO">
<!ELEMENT Member EMPTY>
<!ATTLIST Member Name CDATA #REQUIRED Type CDATA #REQUIRED>
<!ELEMENT Context (#PCDATA)>
<!ELEMENT TraderServiceType (BaseServiceTypes?, Property*)>
<!ATTLIST TraderServiceType Id CDATA #REQUIRED Name CDATA #REQUIRED
                            Masked (YES | NO) "NO">
<!ELEMENT BaseServiceTypes (BaseServiceType*, Link*)>
<!ELEMENT BaseServiceType EMPTY>
<!ATTLIST BaseServiceType Name CDATA #REQUIRED>
<!ELEMENT Property EMPTY>
<!ATTLIST Property Name CDATA #REQUIRED Type CDATA #REQUIRED
                   Mode (NORMAL|READONLY|MANDATORY|
                       MANDATORY_READONLY) "NORMAL"
                   Derived (YES | NO) "NO">
```

Table 2. Service offer DTD

```
1  <!ELEMENT ServiceOfferDescription (OfferType, Property*, ObjectReference)>
2  <!ELEMENT OfferType EMPTY>
3  <!ATTLIST OfferType Name CDATA #REQUIRED>
4  <!ELEMENT Property (DynamicPropEval, ExtraInfo)?>
5  <!ATTLIST Property Name CDATA #REQUIRED Value CDATA #IMPLIED>
6  <!ELEMENT DynamicPropEval (#PCDATA)>
7  <!ATTLIST DynamicPropEval ReturnType CDATA #REQUIRED>
8  <!ELEMENT ExtraInfo EMPTY>
9  <!ATTLIST ExtraInfo Type CDATA #REQUIRED Value CDATA #REQUIRED>
10 <!ELEMENT ObjectReference (#PCDATA)>
```

3. AN EXTENSION TO A CORBA TRADER

An extension to a trader, called the CORBA/XML Transformer (CXT), is responsible for transformation of service descriptions between the CORBA description format and XML (Figure 1). The CXT is designed as a separate module, in addition to existing components of the trader, so that it does not affect the normal operation of the trader. At present, resultant XML documents from the transformation are kept in the underlying file system.

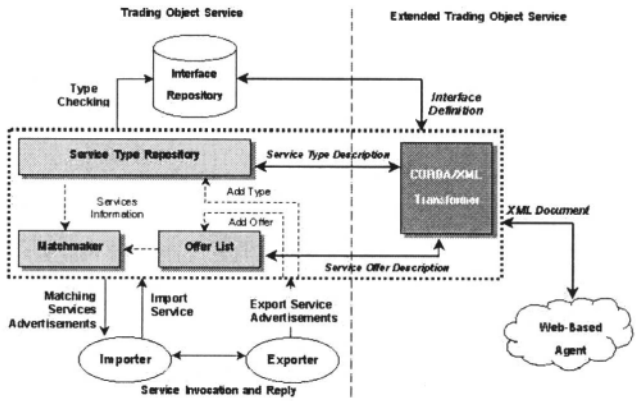

Figure 1. Trader with CORBA/XML transformer

We extend a CORBA trader by adding to CosTrading.idl the following interface definition of the CXT.

```
interface CorbaXmlTransformer {
    typedef Istring Identifier;
    exception InvalidXmlFileLocation { string location; };
```

```
    exception InvalidXmlDocument{};
    exception UnknownInterface { Identifier if-name;};
    exception ServiceTypeExists { ServiceTypeName name;};
    long transform_all_type();
    boolean transform_type(in ServiceTypeName type)
        raises(UnknownServiceType, IllegalServiceType);
    long transform _all_offer(out OfferIdSeq ids);
    long transform _offer(in ServiceTypeName type, out OfferIdSeq ids)
        raises(UnknownServiceType, IllegalServiceType);
    boolean transform_offer_id(in OfferId id)
        raises(UnknownOfferId, IllegalOfferId);
    OfferId transform-xml(in string file)
        raises(InvalidXmlFileLocation, InvalidXmlDocument,
            UnknownInterface, UnknownServiceType,
            ServiceTypeExists);
};
```

The behaviour of the CXT can be described as follows:

transform_all_type( ) : This operation is used to transform all service type descriptions within the trader. Its return type is *long* indicating the number of service types that have been transformed.
transform_type( ) : This operation is used to transform the description of the specified service type. Its return type is *boolean* indicating success or failure of the transformation. Two exceptions may be raised during the operation:

- *CORBA::CosTrading::UnknownServiceType:* The specified service type does not exist in the Service Type Repository of this trader.
- *CORBA::CosTrading::IllegalServiceType* : The specified service type is malformed.

transform_all_offer() : This operation is used to transform all service offers within the trader. The number of offers successfully transformed is returned in a *long* value and their offer ids are returned in the *OfferIdSeq* output parameter.
transform_offer( ) : This operation is used to transform all service offers of the specified service type. This also includes offers of subtypes of the specified type. The number of offers that are successfully transformed and their offer ids are returned. Two exceptions may be raised during the operation:

- *CORBA::CosTrading::UnknownServiceType* : The specified service type does not exist in the Service Type Repository of this trader.
- *CORBA::CosTrading::IllegalServiceType* : The specified service type is malformed.

transform_offer_id() : This operation transforms an offer with the specified offer id. It returns *boolean* to indicate success or failure of the transformation. Two exceptions may be raised during the operation:

- *CORBA::CosTrading::UnknownOfferId* : There is no offer with the specified offer id within this trader.
- *CORBA::CosTrading::IllegalOfferId* : The specified offer id does not comply with the rules for object identifiers defined in [1].

transform-xml() : This operation transforms an XML service description into the CORBA description format according to the root of the document which specifies the type of the information. That is, a service type document will be transformed to a service type description and a service offer document to a service offer description. Several exceptions may be raised during the operation:

- *CORBA :: CosTrading :: CorbaXmlTransformer :: InvalidXmlFileLocation*: The XML document location specified as a URL is malformed, or the specified file location does not exist.
- *CORBA :: CosTrading :: CorbaXmlTransformer :: InvalidXmlDocument*: The XML document is not valid according to the service type DTD and service offer DTD.
- *CORBA::CosTrading::CorbaXmlTransformer::UnknownInterface*: This exception may occur when tranforming from a service type document. If the IR does not store the definition of base interfaces of the service type (i.e. interfaces of the base service types), the transformation fails with no descriptions added to the Service Type Repository and IR.
- *CORBA::CosTrading::UnknownServiceType*: This exception may occur in two cases. One is when transforming from a service type document. If base service types of the service type do not exist in the Service Type Repository of the trader, the transformation fails with no descriptions added to the Service Type Repository and IR. The other is when transforming from a service offer document. If the service type of the offer does not exist in the Service Type Repository, the transformation fails with no descriptions added to the Offer List.
- *CORBA::CosTrading::CorbaXmlTransformer::ServiceTypeExists*: This exception may occur when transforming from a service type document. If the service type already exists in the Service Type Repository of the trader, the transformation fails with no description added to the Service Type Repository and IR. This is to prevent the existing service type from being replaced by a different description with the same name. Note that if the interface definition for this service type already exists in the IR, it may be the case that there is another service type, which exhibits that

interface, stored within the trader. The transformation will be allowed because more than one service type may support the same interface.

4. IMPLEMENTATION OF CORBA/XML TRANSFORMER

The CXT (Figure 2) is implemented to retrieve service descriptions from the Service Type Repository, Offer List, and IR when a client program requests for transformation of a service type or service offer. It then creates an instance of the Document Object Model (DOM) [10] to represent those descriptions before generating an XML document. On the other hand, the CXT can also construct a DOM instance by parsing an XML service description before writing the information as CORBA descriptions into the trader and IR.

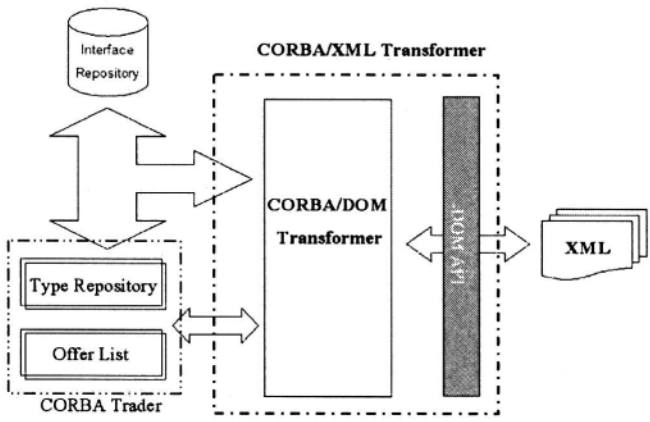

Figure 2. CORBA/XML transformer components

We have developed a prototype of the CXT using Java. This prototype uses the implementation of DOM called Java API for XML Parsing (JAXP) from Sun Microsystems [11]. An extension is added to a JacORB trader [12] that works with the IR from ORBacus [13]. The CXT is implemented in such a way that it uses only the operations specified in the standard IDL interfaces of the trader and IR so that it can be easily integrated with various implementations of the trader and IR. XML service descriptions generated by our CXT are stored as files in a location that is also accessible by HTTP.

The class model of our implementation is presented in Figure 3. The implementation is composed of three main classes that can be added to any Java implementations of CORBA-compliant traders: *ServiceTypeExporter* for transformation of CORBA service types to XML, *ServiceOfferExporter* for transformation of CORBA service offers to XML, and *ServiceImporter* for trans-

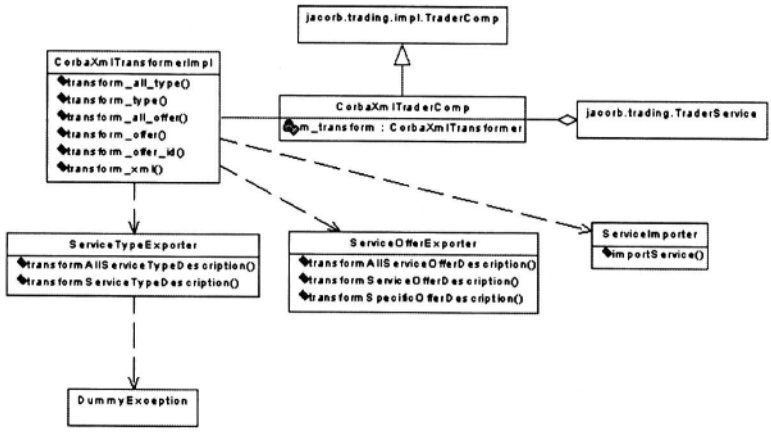

Figure 3. CORBA/XML transformer class model

formation of XML service descriptions to CORBA description format. Other classes are used for integrating with the JacORB trader.

5. USE OF CORBA/XML TRANSFORMER

In this section, we discuss some possible use of the CXT.

5.1. Service discovery service

As mentioned earlier, the CXT has been used to provide our Service Discovery Service (SDS) prototype with service descriptions in XML format [3]. Figure 4 shows the overview of the SDS.

The SDS obtains service descriptions from multiple traders and can also federate with other SDSes to extend search space. A Trader Agent will intercept new service advertisements or updates that are sent to its associated trader and requests the CXT to transform the service descriptions into XML documents before passing onto the Service Provision Centre of the SDS for storage. Search on these XML service descriptions can be conducted via the Search Interface that supports multiple XML query languages and keyword search and is accessible to both CORBA clients and Web users. SDS users can discover service types, service offers, and service interfaces information without having to know exact details of the required services. Details and the comparison between a trader and the SDS can also be found in [14].

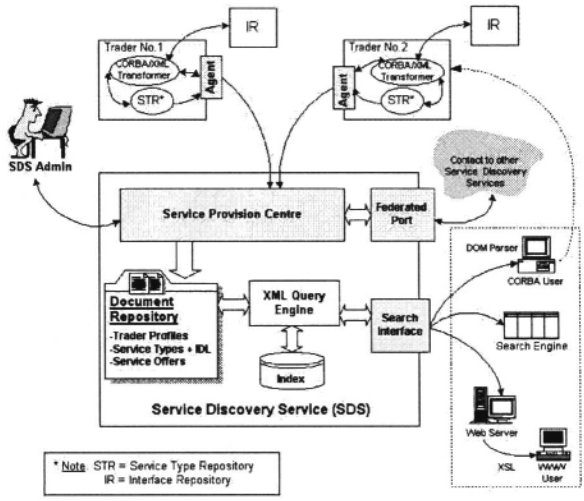

Figure 4. Service discovery service components

5.2. Convenient access to trader

Another advantage of having the trader extended with the CXT is that access to service descriptions within the trader from other environment like World Wide Web is more convenient by using HTTP directly. Normally, a Web user can access the trader's service descriptions (as CORBA objects) using IIOP via a browser with ORB plug-in, but this requires a user-side effort (e.g. by an applet) to represent those CORBA service descriptions in a form that is understandable to the user. Associating the CXT with the trader allows server-side manipulation that makes service descriptions Web-ready as XML documents, and hence the access can be by HTTP. The user does not require a browser with ORB plug-in and if the browser is XML-enabled, service descriptions can be displayed directly; otherwise simple use of the Extensible Stylesheet Language Transformations (XSLT) [15] is required.

We have tried with our Web-based client system in Figure 5. A user can invoke operations on the CXT from the Web environment using Java Servlet and JavaServer Page (JSP). The Servlet mediates between the browser and a Java ORB service, while JSP renders dynamic HTML to be displayed on the browser. Within this architecture, there exist JavaBeans that accept requests from the browser and invoke operations on the extended trader. The results are sent to their associated JSPs and then back to the browser. The functions of the JavaBeans are:

- View service types and service offers within the trader.
- Display a specified service type description.

– Display a specified service offer description.
– Request the CXT to transform a specified service type into XML.
– Request the CXT to transform a specified service offer into XML.
– Request the CXT to transform a specified XML service description for the trader.
– Call XSLT to render an HTML document from a specified XML file.

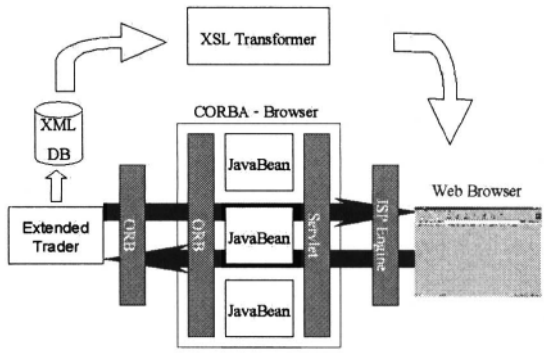

Figure 5. Web-based client prototype

Our model has been tested on Linux 2.2.13 using Apache 1.3.6 as a Web server, Tomcat 3.1 as a Servlet/JSP engine, and JDK version 1.2.2. Test data comprise 7 service types and 12 service offers added to the trader and the CXT is called by the Web-based client system. We have also successfully tested on Windows 98 using Sun's JavaServer Web Development Kit (JSWDK) 1.0 as a Servlet/JSP engine and JDK version 1.1.7b. Figure 6(a) and 6(b) show sample results of the transformation of a service type called PlainConnection and its offer, respectively as a raw XML document and as an HTML document rendered from XML.

Other means of cross-platform communication via HTTP is also possible by an XML protocol such as the Simple Object Access Protocol (SOAP) [16] that allows a message in XML format to be enclosed with some binding protocols including HTTP. Our Web-based client system may use SOAP for the communication between the browser or JSP and the Servlet for request/reply of transformation. However, this is not of our concern since it is only another access protocol; both sides of the communication still have to understand and process the XML message enclosed with the request and reply.

Since the CXT also allows transformation of XML service descriptions into CORBA format, service providers may use this feature for Web-based export

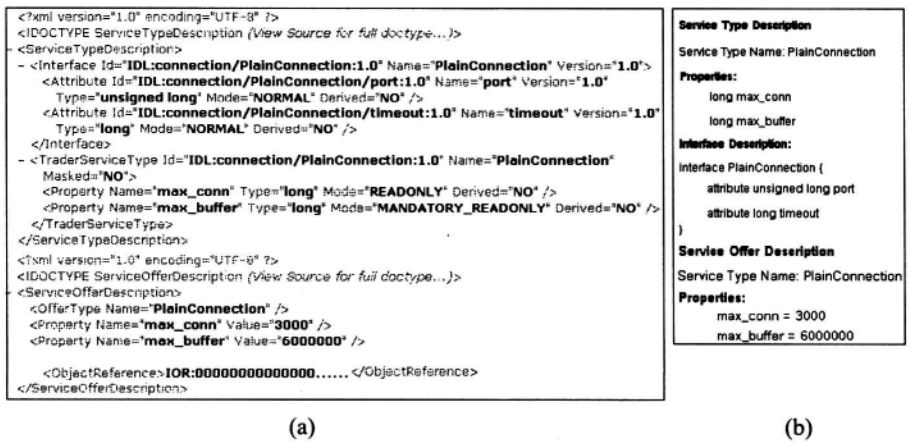

Figure 6. Example of results of transformation in XML and HTML

of service types and service offers without having to write CORBA programs to do so. The CXT can conveniently help with the exchange or import of trader contents, e.g. when exchanging service descriptions between traders or when constructing a replica of a particular trader.

6. CONCLUDING REMARKS

The CXT extension to a CORBA trader provides a way to describe and generate XML version of CORBA service descriptions and also to transform XML descriptions back to the CORBA format. The transformation is straightforward based on our simple service type and service offer DTDs and this extension is now used within our prototype of the service discovery service. We have also integrated it with our service change notification system [17] that notifies subscribing clients about change of advertised services with details of change in XML documents. Another benefit is that the CXT makes trader's descriptions Web-ready for HTTP access by Web users. We hope that the CXT can be applied further to other applications.

As stated earlier, the transformation of service descriptions can be more standardised by adopting XMI to describe the trader's service description model. This will enable further exchange of service information between several kinds of traders or directory services. We may also used XML Schema [18] to constrain service descriptions instead of DTDs.

Acknowledgments

This work is supported by the Thailand-Japan Technology Transfer Project (TJTTP-OECF) and the Development Grants for New Faculty/Researchers of Chulalongkorn University.

References

[1] Object Management Group, "Trading Object Service Specification", Revised Ed., March 1997.

[2] Object Management Group, "The Common Object Request Broker: Architecture and Specification", Revision 2.2, February 1998.

[3] W. Suphasanthitikul and T.Senivongse, "An Architecture for a Service Discovery Service in CORBA", Proceedings of the National Computer Science and Engineering Conference (NCSEC 2000), Bangkok, Thailand, 16-17 November 2000, pp. 49-54.

[4] T. Bray, J. Paoli, C.M. Sperberg-McQueen, and E. Maler, "Extensible Markup Language (XML) 1.0 Specification (2nd Edition)", W3C Recommendation, 6 October 2000, http://www.w3c.org/xml.

[5] Object Management Group, "XML Metadata Interchange (XMI)", 16 July 1998.

[6] V. Vasudevan and T. Bannon, "WebTrader: Discovery and Programmed Access to Web-Based Services (Draft)", OBJS Techinical Report, 1999, http://www.objs.com/agility/tech-reports/9812-web-trader-paper/WebTraderPaper.html.

[7] A. van Hoff, H. Partovi, and T. Thai, "The Open Software Description Format (OSD)", Submitted to W3C, 13 August 1997, http://www.w3.org/TR/NOTE-OSD.html.

[8] R.S. Hall, D. Heimbigner and A.L. Wolf, "Specifying the Deployable Software Description Format in XML", SERL Technical Report CU-SERL-207-99, Software Engineering Research Laboratory, Department of Computer Science, University of Colorado, March 1999.

[9] O. Liechti, M.J. Sifer, T. Ichikawa, "Structured Graph Format: XML Metadata for Describing Web Site Structure", Computer Networks and ISDN Systems Vol. 30 No. 1-7, 1 April 1998, pp. 11-21.

[10] L. Wood et al., "Document Object Model (DOM) Level 1 Specification Version 1.0", W3C Recommendation", 1 October 1998, http://www.w3.org/TR/REC-DOM-Level-1.

[11] Java API for XML Parsing (JAXP), http://java.sun.com/xml.

[12] JacORB – a free Java ORB, http://jacorb.inf.fu-berlin.de.

[13] ORBacus, http://www.ooc.com/.

[14] T. Senivongse and W. Suphasantithikul, "An XML-Based Architecture for Service Discovery", Submitted to the 5th International Enterprise Distributed Object Computing Conference (EDOC'2001), Seattle, Washington, USA, 4-7 September 2001,

http://www.cp.eng.chula.ac.th/faculty/tsv/research/publications/home.html.

[15] J. Clark, "XSL Transformations (XSLT) Version 1.0", W3C Recommendation, 16 November 1999, http://www.w3c.org/TR/xslt.

[16] D. Box et al., "Simple Object Access Protocol (SOAP) 1.1", W3C Note, 8 May 2000, http://www.w3c.org/TR/SOAP.

[17] P. Suriyentrakorn and T. Senivongse, "An Approach for Service Change Notification in Distributed Systems", Proceedings of the National Computer Science and Engineering Conference (NCSEC 2000), Bangkok, Thailand, 16-17 November 2000, pp. 43-48.

[18] D.C. Fallside, "XML Schema Part 0: Primer", W3C Recommendation, 2 May 2001, http://www.w3c.org/TR/xmlschema-0.