second draft submitted to 4th IEEE International Software Engineering Standards Symposium (ISESS '99).

Toward the interoperable software design models: quartet of UML, XML, DOM and CORBA

Junichi Suzuki
Department of Computer Science,
Graduate School of Science and Technology,
Keio University
Yokohama, 223-8522, Japan
+81-45-563-3925
suzuki@yy.cs.keio.ac.jp

Yoshikazu Yamamoto
Department of Computer Science,
Graduate School of Science and Technology,
Keio University
Yokohama, 223-8522, Japan
+81-45-563-3925
yama@cs.keio.ac.jp

ABSTRACT

Unified Modeling Language (UML) has been widely accepted as an object-oriented analysis/design methodology in the software engineering community, and is in the process of revised standardization at OMG (Object Management Group). One of the current major enhancements is SMIF (Stream-based Model Interchange Format) specification, which aims to interchange UML models in a standard based way. It is expected to be based on XML (eXtensible Markup language) standard and used in various development tools such as CASE tools, automatic documentation tools and repositories.

This paper addresses a standard-based UML model interchange and presents our effort to make UML interoperable. We developed a XML-based exchange format called UXF (UML eXchange Format) and a distributed model management system for UML. The system leverages the team development, reuse of design models and tool interoperability by interchanging the model information with XML through the Document Object Model (DOM) interface that is implemented on top of CORBA (Common Object Request Broker Architecture). DOM provides a platform and programming language neutral interface to manipulate the content, structure and style of documents.

Our work shows a practical application of some key standards in terms of the software model interchange. It provides multiple levels of interoperability for UML, thereby UML models can be highly interoperable.

Keywords

UML, XML, DOM, CORBA, Software model interchange, CASE data interchange, UML model interchange

1. Introduction

Unified Modeling Language (UML) [1-8] has been widely accepted as an object oriented software analysis/design methodology in the software engineering community. It provides most of the concepts and notations that are essential for documenting object oriented models. As a publicly available standard, UML is now in the process of revision at Object Management Group (OMG). Enhancements include the improvement of Object Constraint Language (OCL), development process, business modeling, realtime modeling and model interchange.

We are interested in the model interchange. The current UML (version 1.1) does not have an explicit format to interchange its models. The capability is quite important to the future UML compliant tools because it is likely a development team resides in separate places on a network environment, and because UML models are not interoperable between development tools due to the lack of an application-neutral exchange format [9]. To solve this problem, OMG issued an RFP (Request For Proposal) for SMIF (Stream-based Model Interchange Format) specification [10-11], to interchange UML models in a standard based way.

This paper addresses a standard based UML model interchange and presents our effort to develop a distributed model management system allowing UML models to be interoperable. This system allows various tools to publish, access, manipulate and interchange UML models formatted with XML (eXtensible Markup Language) [12] through its published interfaces based on Document Object Model (DOM) [13], a standard by World Wide Web Consortium (W3C). DOM defines a set of interfaces to manipulate the content, structure and style of documents on the Internet. We implemented DOM interfaces on top of CORBA (Common Object Request Broker Architecture) [14], which is a standard for the middleware in the distributed object environment; i.e. ORB (Object Request Broker). By combining promising standards, we achieved an open system allowing to manage the interoperable UML model in a distributed environment.

The remainder of this paper is organized as follows. Section 2 overviews UML, XML, DOM and CORBA. Section 3 describes an UML interchange format called UXF (UML eXchange Format). Section 4 presents the motivation and merits of the combination of technologies described in Section 3, and then describes the architecture and implementation of our system to manage UML models over the network environment. We conclude with a note on the future work, in Section 5 and 6.

2. UML model interchange and its enabler technologies

2.1 Unified Modeling Language (UML)

UML is the union of the previous leading object modeling methodologies; Booch [15], OMT [16] and OOSE [17]. When UML was published in late 1996, it quickly gained momentum and became the de-facto standard for object-oriented modeling. UML has been also submitted to the Object Management Group (OMG) to become a public standard, and included additional constructs that ancestors did not address, such as the extension for business modeling [6], Object Constraint Language (OCL) [7] and Object Analysis & Design CORBAfacility Interface Definition [8]. It is the state of the art convergence of practices in the academic and industrial community. Most developers are expected to eventually choose UML for their modeling work.

UML defines the following diagrams for the object modeling, according to various perspectives to a target problem domain:

- Structural diagrams:
    - Class diagram
    - Object diagram
- Behavioral diagrams:
    - Use case diagram
    - Sequence diagram
    - Collaboration diagram
    - State transition diagram
    - Activity diagram
- Implementation diagrams:
    - Activity diagram
    - Component diagram
    - Deployment diagram

Using these diagrams with the fine level of abstraction, complex systems can be modeled through a small set of nearly independent diagrams. UML provides two aspects for constructs in the above diagrams:

- Semantics: The UML metamodel defines the abstract syntax and semantics of object modeling concepts.
- Notations: UML defines graphical notations for the visual representation of its model elements.

While UML defines the above coherent constructs and their interchangeable semantics, it does not intentionally provide the explicit format to exchange the model information.

2.2 UML model interchange

OMG has issued a RFP for SMIF, as described in Section 1, and some proposals have been received. Responses include CDIF (CASE Data Interchange Format) [18] based, STEP based and XML based proposals. Also, other formats are proposed apart from the SMIF standard [19, 20, 26, 27].

Such application neutral formats facilitates:

- Intercommunications between software developers: The Internet is an emerging infrastructure to distribute and share software model information, because it is effective and economical for making information available to the widely separated group of individuals. Within the Internet/Intranet environment, especially the Web environment, we can represent, encode and ultimately communicate software modeling insights and understandings with each other. An application-neutral interchange format simplifies the circulation of UML models between software developers.
- Interoperability between development tools: Software models are dynamically changed during the analysis/design, revision and maintenance phases, and the software tools used by a development team employ their own proprietary formats to describe the software model information. An application-neutral interchange format allows UML models to be interoperable between development tools throughout the lifecycle of software development. Once encoded with such a format, model information can be reusable for a wide range of usage with different strengths of different tools (Figure 1).

The most important factor in exchanging UML models is the semantics within the models should be described explicitly and transferred precisely. We are interested in a XML based transfer vehicle and have proposed an exchange format called UXF (UML eXchange format) [9, 19].

2.3 XML (eXtensible Markup Language)

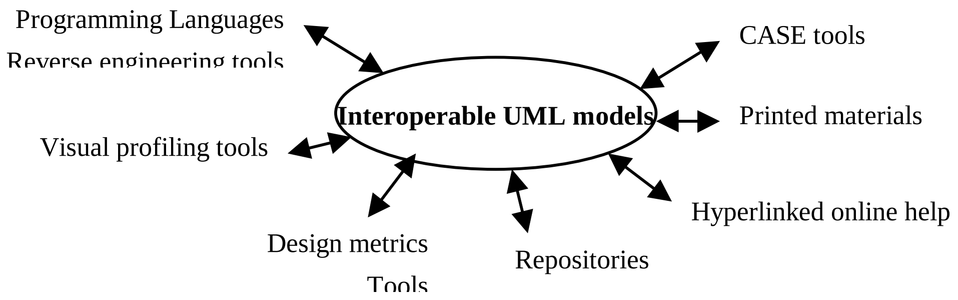


Figure 1: The interoperability between development tools.

XML is a data description language standardized by W3C. XML is a sophisticated subset of SGML (Standard Generalized Markup Language: ISO 8879), and designed to describe document content using arbitrary tags. As its name implies, the extensibility is a key feature of XML; users or applications are free to declare and use their own tags and attributes. Therefore, XML ensures both the logical structure and content of semantics-rich information is retained.

XML emphasizes description of information structure and content as distinct from its presentation. The data structure and its syntax are defined in a DTD (Document Type Definition) specification, which is a derivative from SGML and defines a series of tags and their constraints. In contrast to information structure, the presentation issues are addressed by XSL (XML Style Language) [21], which is also a W3C's emerging standard for expressing how XML-based data should be rendered. XSL is based on DSSSL (Document Style Semantics and Specification Language ISO/IEC 10179) and interoperable with CSS (Cascading Style Sheet), which was originally a style definition language specific to HTML. In addition to XML and XSL, Xpointer [22] and Xlink [23] are in the process of standardization at W3C, a specification to define anchors and links within XML documents. As such, XML has great potential as an exchange format for many kinds of structured data, and increases the productivity to author, maintain and view this data, together with the style sheet and linking mechanisms.

2.4 UXF (UML eXchange Format)

UXF is a XML based format to emcode and exchange UML models [9, 19]. It is carefully designed to be:

- Simple: UXF is compact by including only UML's semantics, while the scope of SMIF includes other semantics (e.g. Meta Object Facility; MOF).
- Intuitive: UXF is designed to be easy-to-understand and readable.
- Natural extension from existing Web environments: UXF is a natural and transparent extension from the existing Web environment. Thus, most of the existing applications around the Web can be used for handling UXF encoded information with relatively minor modifications.

To author and view UML models encoded with UXF, existing markup languages could be converted to UXF, and most development tools such as CASE tools, documentation tools, visual profiling tools and document repositories, can be modified so that they recognize UXF. In the current situation where many XML-aware applications exist, it is relatively easy to extend existing tools. Also, UML-related technical materials formatted in UXF can be handled by every Web application that manipulates HTML as well as Web browsers/servers, in the near future. As a result, UXF allows the seemless uses of the UML models among development tools (Figure 1). This feature increases our productivity of UML modeling.

At present, UXF is not compliant to SMIF intentionally for the simplicity of the format. Also, SMIF is just proposed and has not been frozen, at the time of this writing. Consequently, we decided to use UXF in this project. Once SMIF is frozen or more mature, we will employ SMIF by developing a translator from UXF to SMIF.

2.5 Document Object Model (DOM)

DOM is a platform and programming language neutral interface that allows to access and manipulate the content, structure and style of Web documents (e.g. HTML and XML documents). It provides a standard set of objects for representing documents, a standard model of how these objects can be combined, and a standard API. It is standardized by W3C.

DOM Level 1 [13] consists of two parts: Core and HTML. The DOM Core specification provides a common set of interfaces that can represent and manipulate XML documents The DOM HTML specification uses the Core specification and provides higher-level interfaces for HTML documents.

At the time of this writing, the DOM level 1 specification (version 1.0) is a W3C proposed recommendation that are capable of the followings:

- Structure navigation, which is a navigation around a document such as accessing and searching elements or attributes.
- Document manipulation, which is a manipulation of document structures such as adding, changing and removing elements or attributes.
- Content manipulation, which allows for navigation and manipulation of document contents.

Additional capabilities including Event model, DTD manipulation and Stylesheet object model are planned for a subsequent version.

The DOM interface is defined with OMG's Interface Definition Language (IDL, ISO standard 14750), a primary component in the CORBA specification, because it is designed to define language neutral interfaces. DOM does not intend to be implemented with CORBA, but implementing DOM interfaces on top of CORBA is reasonable for our goal to make UML models highly interoperable (see also Section 2.7).

2.6 Common Object Request Broker Architecture (CORBA)

CORBA is a standard for heterogeneous object interoperability. It is standardized by OMG and recognized as a Publicly Available Standard (PAS) of ISO. It provides a standard way to share objects and their behaviors on a network.

The CORBA specification defines the interfaces and components that compose an Object Request Broker (ORB). An application that needs to access the services and functionality of a remote object uses an ORB to send a message to the object and receive the results. A series of interfaces in CORBA allows to distribute remote objects on multiple platforms in a way that is seamless and transparent to applications. The architecture itself is isolated from the actual transport (e.g., TCP, IPX, SNA), thereby allowing an open-ended standard.

One of the key components in CORBA is OMG Interface Definition Language (IDL), a language to define interfaces of remote objects. It is programming language neutral by providing a mapping between IDL and each language. Another important component is Internet Inter-ORB Protocol (IIOP), which is a standard protocol based on TCP. IIOP leverages the interoperability between distributed objects. Currently, there are over 60 implementations of CORBA supporting over 15 languages (e.g. C++, Java, COBOL, CLOS and Python) and over 30 platforms from Windows CE to mainframes.

2.7 Three levels of interoperability

Our project intends to prepare three levels of interoperability for UML:

- UXF: UXF allows UML models to be interoperable between UML compliant tools.
- DOM: DOM allows XML formatted data (i.e. UXF data) to be interoperable between XML-aware tools through the uniform interfaces.
- CORBA: CORBA provides the standard interfaces to allows DOM-aware tools to interact with each other on a network, thereby UXF formatted data can be transferred between distributed DOM-aware tools.

Consequently, UML models encoded with UXF can be interoperable between any UML compliant tools, any XML tools, any DOM tools and any ORBs.

3. UML eXchange Format (UXF)

In terms of exchanging model information between development tools, there can be two types of information that should be exchanged [20]:

- Model-related information
- View-related information

While model-related information is a series of building blocks to represent a given problem domain (e.g. classes, attributes and associations), view-related information is composed of the way in which the model is rendered (e.g. the shapes and position of graphical objects). This paper concentrates on exchanging the model-related information. The interchange of the view-related information is future work, but it would be easy to describe the view-related information with XSL.

3.1 UXF DTDs

As described above, the UXF specification actually consists of a series of XML DTDs. It provides the mapping of UML model information into document tags in the DTDs. UXF captures the constructs (i.e. model elements) in a UML metamodel, and defines each construct as a tag (i.e. a document element) straightforwardly. The attributes of each UML construct are mapped into attributes of the corresponding UXF tag.

We have specified UXF DTDs for the following three UML diagrams. These diagrams are fundamental for the analysis and design of problem domains.

- Class diagram: A class diagram shows the static structure of classes and relationships between them. This diagram also defines the foundation for other diagrams that specify different aspects of the problem domain.
- Collaboration diagram: A collaboration diagram shows an interaction organized around the objects in the interaction and their links to each other.
- Statechart diagrams: A statechart diagram shows the sequences of states that an object goes through during

its life in response to received stimuli, together with its responses and actions.

Table 1 depicts the comparison of UML model elements and UXF tags. Current UXF supports most elements in the UML's Core package, Collaboration package, State Machines package and some elements in other packages (see Table 1).

Using UXF, most essential concepts and constructs in UML can be mapped to the stream-based exchange format seamlessly. Sample markup (encoding) examples can be found in [24]. Note that constructs described with UXF are not shared between different diagrams for the simplicity. Sharing model information between different diagrams consistently is considered the responsibility of UXF aware applications.

4. Distributed model management system

This section describes our effort to distribute UML models over the Internet/Intranet environment.

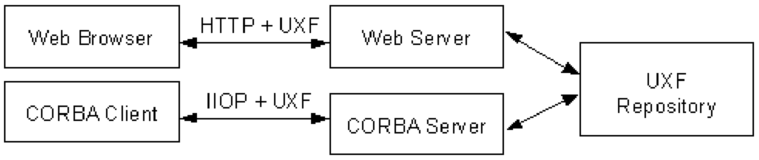
4.1 System's goal

We developed a prototype system to share and manage the UML design information precisely within a distributed environment based on the Internet. The Internet-oriented centralized management of design information leverages:

- Team development: allows analysts, designers and programmers to continue their work concurrently at physically separated places. They can create, retrieve and change the UML model information.
- Reuse of development information: allows developers to communicate with each other, and eases a smooth progress of the development project. This feature increases productivity of the team development. The capability to maintain development information at a single point throughout a software lifecycle is vital to development teams for archival purpose, because every

UML Package	UML Model Element	UXF Representation
Core	Association	<Association>
	AssociationEnd	<AssocRole>
	Attribute	<Attribute>
	Class	<Class>
	Dependency	<Dependency>
	Generalization	<Generalization>
	Interface	<Interface>
	Operation	<Operation>
	Parameter	<Parameter>
Auxiliary Elements	Refinement	<Refinement>
Extension	TaggedValue	<TaggedValue>
Common Behavior	Exception	<Exception>
	Action	<Action>
	ActionSequence	<ActionSequence>
	Instance	<Instance>
Model Management	Model	<Model>
	Package	<Package>
Collaborations	Collaboration	<collaboration>
	Interaction	<Interaction>
	Message	<Message>
StateMachines	CompositeState	<CompositeState>
	Event	<Event>
	Guard	<Guard>
	State	<State>
	Transition	<Transition>
	PseudoState	<PseudoState>

Table 1: Comparision between UML model elements and UXF tags



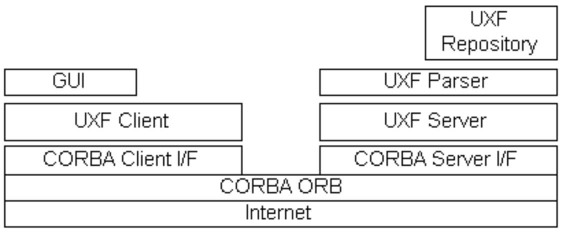


Figure 2: Deployment architecture of our original prototype system that allows to share UML models over the Internet (left), and layered architecture for CORBA based system (right).

```
typedef sequence<string> StrSequence;
interface UxfHandler {
    StrSequence getPackageList();
    StrSequence getClassList( in string pkgName );
    StrSequence getOperationList( in string pkgName, in string className );
    string getPackageDescription();
    string getClassDescription( in string pkgName );
    string getOperationDescription( in string pkgName, in string className );
    boolean findConstruct( in string name );
}
```

Figure 3: Selected IDL interfaces of a CORBA server for accessing UXF formatted information.

team typically has large volumes of materials. Developers can use these materials in their work to refer and revise the current information, or record results of experiments or historical logs.

4.2 Our first prototype system

We developed a system to meet the above requirements, on top of the existing Web environment and a CORBA compliant ORB. It was based on the three-tier deployment architecture, and provides two kinds of accesses to UXF documents: via HTTP and IIOP (the right of Figure 2).

The HTTP access aims to allow client applications including Web browsers to refer the UXF documents stored in Web servers or any backend databases. The left of Figure 4 is a sample screenshot of a Web browser displaying a UXF document with a corresponding XSL stylesheet. If other stylesheets are prepared, different outputs can be displayed as suited to a specific purpose.

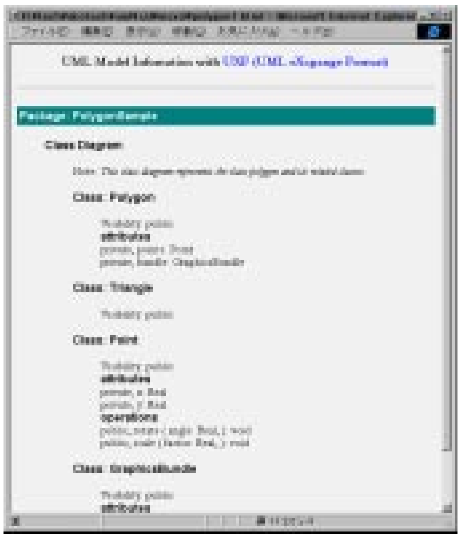
The IIOP access aims to allow developers at separated places to consistently register, refer, process and change UML models. A server application parses UXF formatted documents at a system's start-up time or on the fly, and builds an in-memory structure of these documents; tree structures of parsed UXF elements. Client applications include simple command-line tools, GUI profiling tools, development environments, etc. The right side of Figure 5 is a sample screenshot of a client-side GUI profiling tool similar to the system browser in Smalltalk. In this tools, the left-side pane shows the list of UML packages, the central pane lists classes in the package, the right side is the list of operations (i.e. methods) of the class selected in the central pane, and the bottom pane shows the comments (Note or TaggedValue in the UML term) for each package, class or method. This tool accesses a CORBA server to obtain the necessary data to display, according to the user's mouse manipulation.

The selected IDL interfaces of a CORBA server are shown in Figure 3. The interface `UxfHandler` in Figure 3 defines a series of methods to access the constructs in class diagrams.

In general, APIs that handle XML documents via middleware are categorized into three groups [25]:

- Source document APIs
  manage the XML document instances directly.
- Element APIs
  manage the parsed elements in document instances. The navigation, selection, control and update on of a XML document is achieved through this API.
- Custom APIs
  provide application specific interfaces, and depend on certain applications with related DTDs.

Our original interface listed in Figure 3 is a custom API specific to the UXF description for class diagrams. With custom APIs, it is easy to understand their semantics, but they are not ideal for the interoperability of UML models, because they should be modified when UXF DTDs or UML itself are revised. This is where the element API comes in.



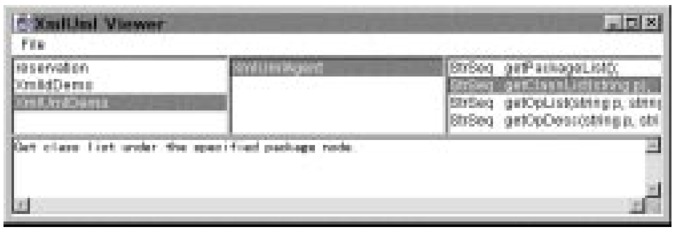


Figure 4: Sample screenshots of a Web browser that displays a UXF document with a XSL stylesheet and a GUI profiling tool that communicates with CORBA servers.

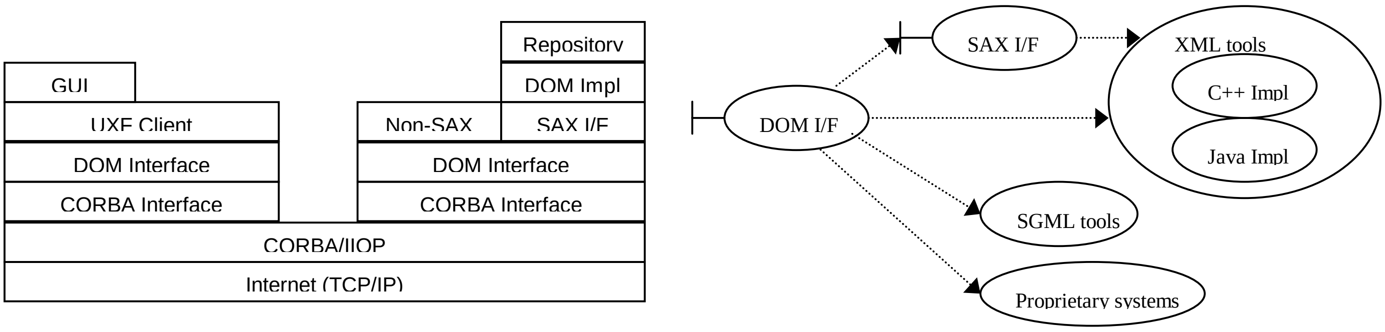


Figure 5: Revised system architecture (left) and connections between the DOM interface and its backends (right).

4.3 Revised system

We revised our system to replace the traditional custom API with the element API to provide more generic interface to UXF encoded data. The DOM interface plays a key role for accessing and manipulating it.

4.3.1 Architecture

As depicted in the left of Figure 5, our layered architecture is revised so that the DOM interface is placed between the CORBA interface and components to process UXF data. The traditional IDL interface, described in Figure 3, is replaced with the DOM interface that is defined by IDL. This means client applications can access UML models through the generic interface, thereby the server-side interface can be kept uniform even if the UXF format or UML is changed.

There can be a wide range of implementations of DOM (right of Figure 5). The difference between implementations is encapsulated with the common DOM interface. It can use backend tools from a variety of selections, including tools to manipulate XML, SGML and HTML or even proprietary systems handling their own format, because it does not specify its implementation issue. We can also easily migrate from existing legacy systems.

Our system employs a defacto standard named "Simple API for XML" (SAX) [28] for a backend of the DOM interface. SAX has been developed collaboratively in the XML community, and defines the common interface between XML parsers and their applications. It is supported by various parsers implemented in Java, C++, Python and so on. SAX allows us to select XML parsers along with a given requirement without effecting other components in the system (right of Figure 5).

The essential components that consist of the DOM Core specification are shown in the left of Figure 6. These defines common parts of documents on the Internet. The DOM extended interface is a set of components to manipulate elements specific to XML (i.e. DTDs). Our system focuses on only DOM Core interface, because it does not have to process any UXF DTDs.

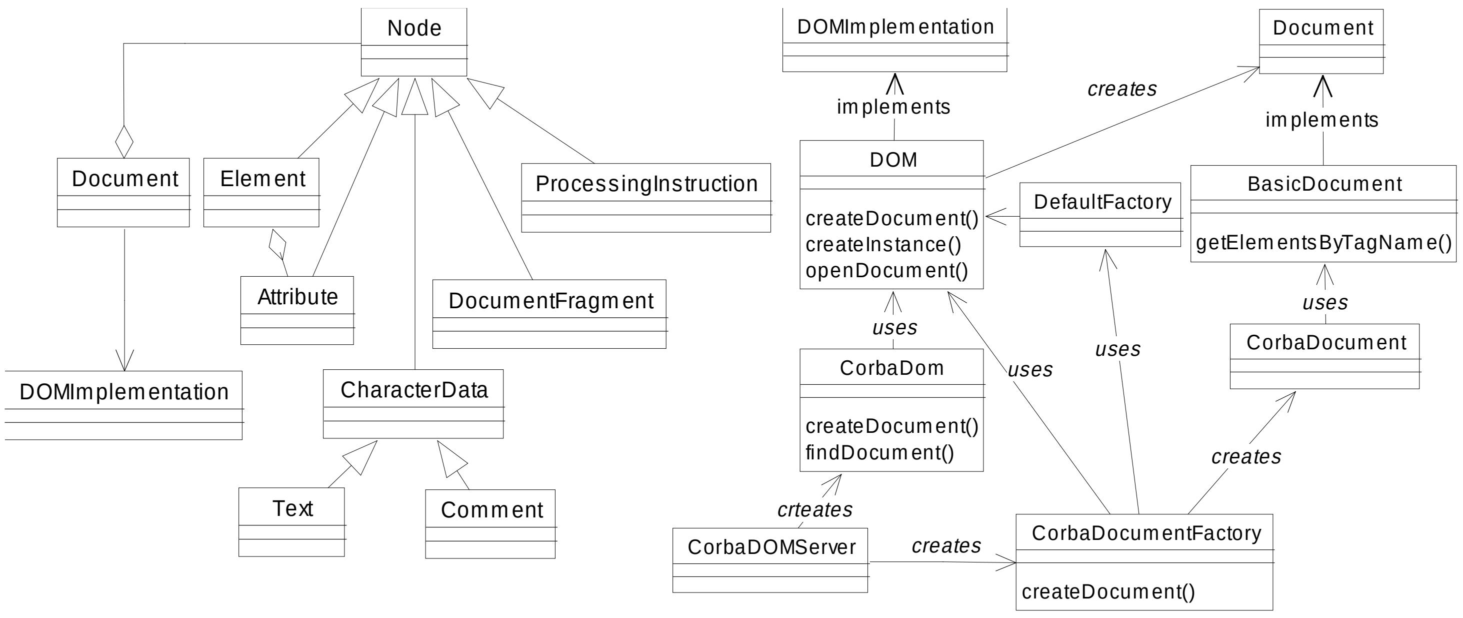


Figure 6: The essential interfaces of the DOM Core specification (left) and their implementations which are used in CORBA environment.

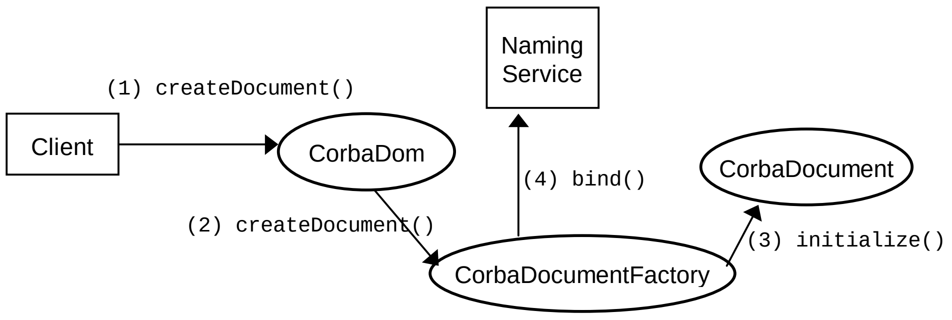


Figure 7: A typical behavior of server-side components.

4.3.2 Implementation

As shown in the left of Figure 5, our system architecture intends to use a SAX-compliant XML parser as an implementation of the DOM interface. Our system currently uses an implementation of the DOM Core specification called Docuverse DOM SDK (DOMSDK) [29], and connects it with a XML parser called Lark [30] using its SAX driver.

DOMSDK provides a series of classes and interfaces implemented with Java. By wrapping them with additional objects, we extended it so that it can run on the CORBA environment, because it assumes a standalone use. The right of Figure 6 depicts the relationship between components in DOMSDK and CORBA objects. DOMSDK provides Java interface classes compliant to the DOM Core specification (`DOMImplementation` and `Document` in this figure), and implementation classes corresponding to interface classes (`DOM`, `BasicDocument` and `DefaultFactory`). To allow these classes to be used on CORBA, we defined a set of classes that are glue to connect DOM and CORBA. The name of every glue class has a prefix "`Corba`". For example, `CorbaDocument` provides the functionality of the interface `Document` in the DOM specification by using `BasicDocument` internally (see the right of Figure 6). The classes with the suffix "`Factory`" are factory classes responsible for creating CORBA objects on demand basis. For example, `CorbaDocumentFactory` creates an instance of `CorbaDocument` with its method `createDocument()`(see the right of Figure 6). All other components are structured in this manner, though the right of Figure 6 includes just two sets of interface class-implementation class-glue class pairs.

Figure 7 shows a typical behavior of server-side components. When `CorbaDom` accepts a request to create a XML document (i.e. a UXF formatted data) from a client application (1), it calls the method `createDocument()` of `CorbaDocumentFactory`, a factory class for `CorbaDocument` (2). This method instanciates `CorbaDocument` by invoking the method `initialize()` (3), and them binds the instance into a CORBA naming service (4). The CORBA naming service is a standard registry for CORBA objects, which allows client applications to locate them. The newly created instance is returned to the client application through `CorbaDom`. After that, the client application can directly communicate the instance to access its content. Our system uses a CORBA implementation called JavaIDL, included in Java Development Kit (JDK) 1.2 or later.

5. Future work

To achieve the goal described in Section 4.1, our system has many enhancements. For example, we are investigating the use of an object-oriented database for a persistent storage of UXF data. It enhances the current transient CORBA objects to be a persistent one, which can maintain the tree structures of parsed UXF elements even after the shutdown of a server. Another enhancement is to enable the concurrent access in the server-side components and make the response time in client applications to be minimum using a callback capability. Also, we are introducing a mechanism of the revision control for UXF data using two tags of XML; <![INCLUDE[…]]> and <![IGNORE[…]]>. In addition, UXF-aware and DOM compliant tools are planned such as diagram editing/drawing tools, documentation tools and visual profiling tools. We are particularly interested in developing different tools with different languages in order to prove the language neutrality in our system.

As for the UXF format, we are refining the existing DTDs and developing ones for all the UML diagrams.

6. Conclusion

This paper addresses how we can provide an open environment for highly interoperable UML models with some emerging standards. We proposed the co-use of XML, DOM and CORBA as enabler technologies, and developed a distributed system to share and manage UML models. We believe our work shows an important step in interchanging UML analysis/design models, and provides a blue print indicating how emerging standards can be used for practical applications in near future. Information on our project can be obtained at [24].

At last, we would like to thank Yuu Tanaka for his help to design UXF DTDs, Uche Ogbuji for our discussion to use DOM and Kenji Shirane for his initial input for UXF applications.

7. References

[1] Rational Software et.al. UML Proposal Summary, OMG document number: ad/97-08-02.

[2] Rational Software et.al. UML Summary, OMG document number: ad/97-08-03.

[3] Rational Software et.al. UML Semantics, OMG document number: ad/97-08-04.

[4] Rational Software et.al. UML Notation Guide, OMG document number: ad/97-08-05.

[5] Rational Software et.al. UML Extension for Objectory Process for Software Engineering, OMG document number: ad/97-08-06.

[6] Rational Software et.al. UML Extension for Business Modeling, OMG document number: ad/97-08-07.

[7] Rational Software et.al. Object Constraint Language Specification, OMG document number: ad/97-08-08.

[8] Rational Software et.al. OA&D CORBAfacility, OMG document number: ad/97-08-09.

[9] Suzuki, J. and Yamamoto, Y. Managing the software design documents with XML. In Proceedings of ACM SIGDOC '98.

[10] DSTC, IBM, Oracle, Platinum Technology and Unisys, Joint Initial Submission to the SMIF RFP. OMG document number ad/98-07-01, 1998.

[11] DSTC, IBM, Oracle, Platinum Technology and Unisys, XMI specification: Appendices. OMG document number ad/98-07-03, 1998.

[12] Bray, T et.al. (ed.). Extensible Markup Language (XML) 1.0. W3C Recommendation, February 1998.

[13] Apparao, V et al (ed.). Document Object Model (DOM) Level 1 Specification version 1.0. W3C proposed recommendation, 18 August 1998.

[14] Object Management Group, Common Object Request Broker Architecture. 1998.

[15] Booch, G. Object-Oriented Analysis and Design 2nd Edition. The Benjamin/Cummings Publishing, 1994.

[16] Rumbaugh, J et.al. Object-Oriented Modeling and Design. Prentice Hall, 1991.

[17] Jacobson, I. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1995.

[18] A series of CDIF specifications are available at http://www.cdif.org/

[19] Suzuki, J. and Yamamoto, Y. Making UML models exchangeable over the Internet with XML. In Proceedings of UML '98, 1998.

[20] Rational Software. UML-Compliant Interchange Format, OMG document number: ad/97-01-13, 1997.

[21] Clark, J. et.al. (ed.). Extensible Stylesheet Language (XSL) version 1.0. W3C Working Draft 18-August-1998.

[22] A Proposal for XSL. W3C, 1998.

[23] Eve Maler et.al. (ed.), XML Pointer Language (XPointer), W3C Working Draft 3,.

[24] Maler, E et.al. (ed.), XML Linking Language (XLink), W3C Working Draft 3.

[25] http://www.yy.cs.keio.ac.jp/~suzuki/project/uxf/

[26] Ohno, K. and Bayer, M. Development of SGML/XML Middleware Component. In *Proceedings of SGML/XML'98 Europe*, 1998.

[27] UML Xchange. at http://WWW.CAM.ORG/~nrivard/uml/umlxchng.html.

[28] UML to Text. at http://www.ccs.neu.edu/home/nickman/com1205/uml-text.html.

[29] SAX 1.0: The Simple API for XML. at http://www.megginson.com/SAX/index.html.

[30] Docuverse DOM SDK. at http://www.docuverse.com/domsdk/.

[31] Bray, T. An Introduction to XML Processing with Lark and Larval. at http://www.textuality.com/Lark/.