<<UML>>'98

Making UML models exchangeable over the Internet with XML:

UXF approach

Junichi Suzuki and Yoshikazu Yamamoto

Department of Computer Science
Faculty of Science and Technology
Keio University
Yokohama, 223-8522, Japan.
+81-45-563-3925 (Phone and FAX)
{suzuki, yama}@yy.cs.keio.ac.jp

Abstract

As Unified Modeling Language (UML) provides most of the concepts and notations that are essential for documenting object-oriented models, it has been widely accepted in the software engineering area. However, UML does not have an explicit format for exchanging its models intentionally. The ability to exchange the models is quite important, because it is likely that a development team resides in separate places on the network environment, and because most current development tools don't provide the interconnectivity of the model information. This paper addresses this problem and proposes UXF (UML eXchange Format), which is an exchange format for UML models, based on XML (Extensible Markup Language). It is a format powerful enough to express, publish, access and exchange UML models and a natural extension from the existing Internet environment. It serves as a communication vehicle for developers, and as a well-structured data format for development tools. UXF shows an important step in sharing and exchanging the model information, and indicates a future direction of the interconnectivity between UML compliant tools.

Keywords

UML, XML, Model exchange

1. Introduction

Since Unified Modeling Language (UML) [1-8] provides most of the concepts and notations that are essential for specifying and documenting object-oriented analysis/design models, it has been widely accepted in the software engineering area. While UML is the union of the previous leading object modeling methodologies; Booch [9], OMT [10] and OOSE [11], it includes additional constructs that these methods did not address, such as Object Constraint Language (OCL) [7] and Object Analysis & Design CORBAfacility Interface Definition [8]. Also, it is the state of the art convergence of practices in the academic and industrial community.

UML defines the following diagrams for the object modeling:

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

- **Semantics**
  A UML metamodel defines the abstract syntax and semantics of object modeling concepts.
- **Notations**
  UML defines graphic notations for the visual representation of the UML semantics.

While UML defines the above coherent constructs and its interchangeable semantics, it does not intentionally provide the explicit format to exchange the model information. The ability to exchange models is quite important, because the network environment such as Internet grows exponentially and it is likely that a development team resides in separate places. In addition, such an ability facilitates the application interconnectivity so that the model information can be exchanged between various tools such as CASE (Computer Aided Software Engineering) tools, diagram editors and reverse engineering tools. As such, the application-neutral format expands the ability to encode, exchange and reuse the model information.

In this paper, we address this problem and describe how our work aims to make UML models exchangeable on the Internet and/or between development tools. To provide a standard-based and application-neutral means, we propose a stream-based exchange format called UXF (UML eXchange Format), which is based on XML (eXtensible Markup Language) and provides a mapping UML to XML. We consider the use of XML as a mechanism for encoding and exchanging the structured data defined with UML. We outline the rationale and fundamentals of UXF, and discuss how we can exploit it to express, publish, share and exchange UML models in the network-based development environments.

The remainder of this paper is organized as follows. Section 2 introduces the current limitations in encoding and exchanging the model information with HTML (HyperText Markup Language), and presents the motivation and merits to employ XML. Section 3 outlines comparisons with related work. Section 4 defines the scope and syntax of UXF and provides a solution to interchange UML models on the Internet or among development tools. We conclude with a note on the current status of project and future work, in Section 5 and 6.

2. Interchangeability of UML Models

2.1 The Internet as an infrastructure of model exchange

As the explosive growth of the Internet, especially the Web, places increasing changes, in which the data exchangeability and application interconnectivity play a central role, it has been effective and economical infrastructure to make information available to the widely separated group of individuals. In the context of software development, it is becoming more important to find ways to exchange the analysis/design models over the Internet/Intranet environment, in which we can represent, encode and ultimately communicate the software modeling insights and understanding with each other, as well as facilitate the automatic processing such as searching, indexing and drawing with development tools.

2.2 Limitations of HTML

As such, since the Web is becoming the ubiquitous environment, HTML has been a major document format, and also used for software documentation. Examples of such tools include javadoc included in Java Development Kit (JDK) [12], which is a translator from the comments in source code of Java into the specification documents written in HTML. While such a tool is valuable and helpful for everyday development work, some important information within software models is unfortunately thrown away in the process of producing HTML documents, due to its fixed tag set. Contents in such HTML documents are meaningful only for human, and its semantics cannot be unambiguously recognized by software tools. In other words, HTML documents generated by documentation tools cannot be reused in other applications other than HTML browsers. In this situation, if there is a more semantics-rich markup language and development tools support it, we can interchange UML model information without losing its semantics.

Also, as described above, it is likely that the members in a software development team, including requirement analysts, system architects, designers and programmers, are working in separated places and relying on the electronic communication. Currently, the predominant means to distribute UML diagrams on the Web is the image-based method, in which GIF or JPEG images representing UML diagrams are inlined within a HTML text. However, it is difficult and expensive (i.e. time consuming) to author, read and maintain these images, and inadequate in that the model information is hidden within images and can not be available for other development tools (e.g. for searching and drawing). Another problem with encoding the

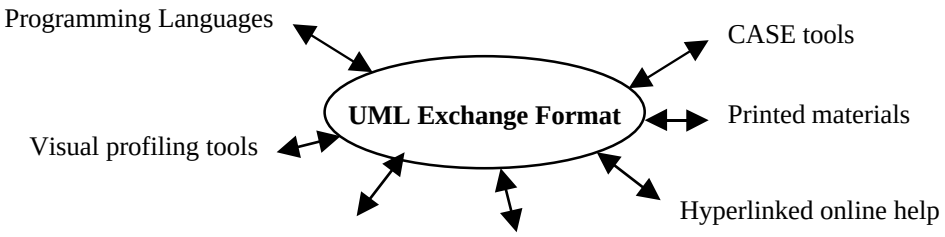

Figure 1: UXF allows the seamless exchange of UML models between development tools.

model information as images is that it requires more network bandwidth. With markup-based encoding, more of the rendering process can be moved to the client machine. Markup describing model information is typically smaller and more compressible than an image of the model.

Coupled with the above problems, the most important factor in exchanging the UML models between software programs is that the semantics within the models should be explicitly described, keeping its semantics. For this purpose, Extensible Markup Language (XML) is a reasonable and practical candidate for a vehicle to interchange UML models.

2.3 Extensible Markup Language (XML)

XML is a data description language standardized by World Wide Web Consortium (W3C) [13]. While HTML is defined by SGML (Standard Generalized Markup Language: ISO 8879), XML is a sophisticated subset of SGML, and designed to describe the data using arbitrary tags. One of the goals of XML is to be suitable for the use on the Web; thus to provide a general mechanism for extending HTML. 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 that both the logical structure and content of semantics-rich information is retained. XML is widely accepted in the Web community now, and the current applications of XML includes MathML (Mathematical Markup Language) [14] to describe mathematical notation, CDF (Channel Data Format) for push technologies, OFX (Open Financial Exchange) [15] to describe financial transactions and OSD (Open Software Distribution) for the software distribution on the Net.

XML focuses on the 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) [16], which is also a W3C 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, XLL (XML Linking Language) is in the process of standardization [17], which is a specification to define anchors and links within XML documents. As such, XML has a great potential as a exchange format for general structured data, and increases the productivity to author and maintain, together with style sheet and linking mechanism, while remaining the feature that HTML has provided.

2.4 UML Exchange Format (UXF)

Faced with the problems described in Section 2.2 and based on an emerging data description language called XML, we propose an exchange format of UML models called UXF (UML eXchange Format). UXF is an application of XML and, designed to be flexible enough to encode and exchange any UML constructs. UXF facilitates:

- **Intercommunications between software developers:**
  UXF is a powerful transfer vehicle for UML models between software developers. It simplifies the circulation of UML models each other, with the human-readable and intuitive format.

- **Interconnectivity between development tools:**
  UXF is a well-structured and portable (i.e. application neutral) format between various development tools. Once encoded with UXF, the information of UML models can be reusable for wide range of uses using different strengths of different tools (Figure 1).

- **Natural extension from existing Web environments:**
  UXF is a natural and transparent extension from the existing Web environment. Thus, it allows to edit, publish, access and exchange the UXF data as easily as is currently possible with HTML. In addition, most of existing applications around the Web can be used for handling UXF with relatively minor modifications.

In order to author UML models encoded with UXF, existing markup languages can 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 which manipulates HTML as well as Web browsers and Web servers, in the near future. As a result, UXF allows the borderless uses of UML models among development tools (Figure 1), and provides the tight integration with Web environments. This feature increases our productivity of UML modeling.

The potential use cases of UXF cover a broad spectrum. Developers including analysts, designers and engineers can communicate their insights, understanding or intention on their UML models, by interchanging UXF formatted files. Also, the ability to maintain technical information during software lifecycle is vital to development teams for archival purpose, because every team typically has large volume of materials. Engineers use these materials in their work to refer and revise the current information, record results of experiments or historical logs. For such uses, well-structured UXF provides a standard way of sharing information, in which we can easily read, process and generate UML models using commonly available tools.

In addition, UXF ensures a variety of possibilities of its output representations; how UXF data should be rendered or viewed. Since UXF can apply arbitrary XSL style sheets, it can be converted into materials in a wide range of media like RTF (Rich Text Format), HTML, LaTeX, PDF (Portable Document Format). Moreover, UXF data can embed hypermedia links with the linking mechanisms of XLL. This allows us to link UML constructs each other. As such, developers can involve in technical materials at all level from electronic versions, printed documents to interactive versions.

3. Related Work

The famous and mature format for exchanging the software modeling information is CDIF (CASE Data Interchange Format) [18]. CDIF is a generic mechanism and format to interchange the software models between CASE tools, and a family of standards defined by the Electronic Industries Association (EIA) and International Standard Organization (ISO). CDIF defines a meta-metamodel, a tool interchange format, and a series of subject areas:

- CDIF Framework for Modeling and Extensibility
- CDIF Integrated Metamodel
  - Foundation Subject Area
  - Common Subject Area
  - Data Modeling Subject Area
  - Data Flow Model Subject Area
  - Data Definition Subject Area
  - Physical Relational Database Subject Area
  - State/Event Model Subject Area
  - Project Planning and Scheduling Subject Area
  - Object-Oriented Analysis and Design Subject Area
  - Presentation Location and Connectivity Subject Area
  - Presentation Global Subject Area
- CDIF Transfer Format
  - General Rules for Syntaxes and Encodings
  - SYNTAX.1

- ENCODING.1

CDIF separates the semantics and syntax from the encoding, and thus provides flexibility in the representation and transfer mechanism. SYNTAX.1 and ENCODING.1 defines the means that allows for a tool-independent exchange of models. CDIF has provided the mapping to UML [19], by using the Foundation Subject Area and CDIF Transfer Format, and by defining the UML subject area that provides the definitions of metamodel entities and their relationships in UML. The UML Subject Area is dependent on the CDIF Foundation Subject Area.

UXF is a UML-specific exchange format and an alternative vehicle to transfer UML models. Since it is a straightforward extension from and transparent to the Web-based distributed environment, it can be easy-to-understand and use for the huge amount of people that is familiar with HTML or SGML. We believe UXF is a practical approach for encoding and exchanging UML models over the Internet.

4. UXF Fundamentals

4.1 UXF Design Principle

In terms of exchanging model information between development tools, there can be two types of information that should be exchanged [17]:

- Model-related information
- View-related information

While model-related information is a series of building blocks that are used 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 (see also Section 5).

As described above, the UXF specification actually consists of XML DTD and provides the mapping of UML model information into document tags in the DTD. UXF captures the constructs (i.e. model elements) in a UML metamodel, and defines each construct as a tag (i.e. document element). The attributes of each UML construct are mapped into attributes of the corresponding XML tag. Table 1 depicts the comparison of UML model elements and UXF elements in our initial work, and Appendix A shows the complete UXF DTD, which includes all tags and attributes. At present, UXF defines the tags that are necessary to describe class diagrams; most concrete constructs in the UML's Core package, which is included in the Foundation package in turn, and some constructs in other packages (see Table 1).

UML Package	UML Model Element	UXF Representation
Core	Association	<Association>
	AssociationEnd	<AssocRole>
	Attribute	<Attr>
	Class	<Class>
	Dependency	<Dependency>
	Generalization	<Generalization>
	Interface	<Interface>
	Operation	<Operation>
	Parameter	<Param>
Auxiliary Elements	Refinement	<Refinement>
Extension Mechanisms	TaggedValue	<TaggedValue>
Common Behavior	Exception	<Exception>
Model Management	Model	<Model>
	Package	<Package>

Table 1. Comparision of UML model elements and UXF elements

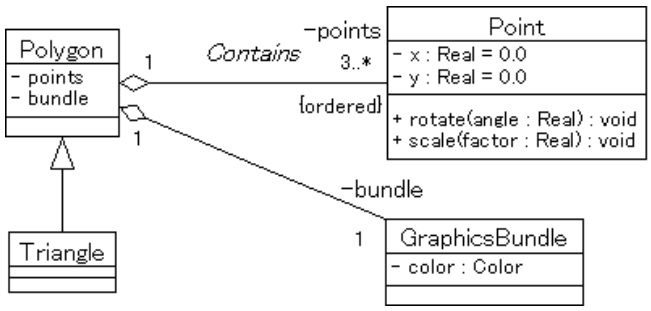

Figure 2: Sample UML class diagram

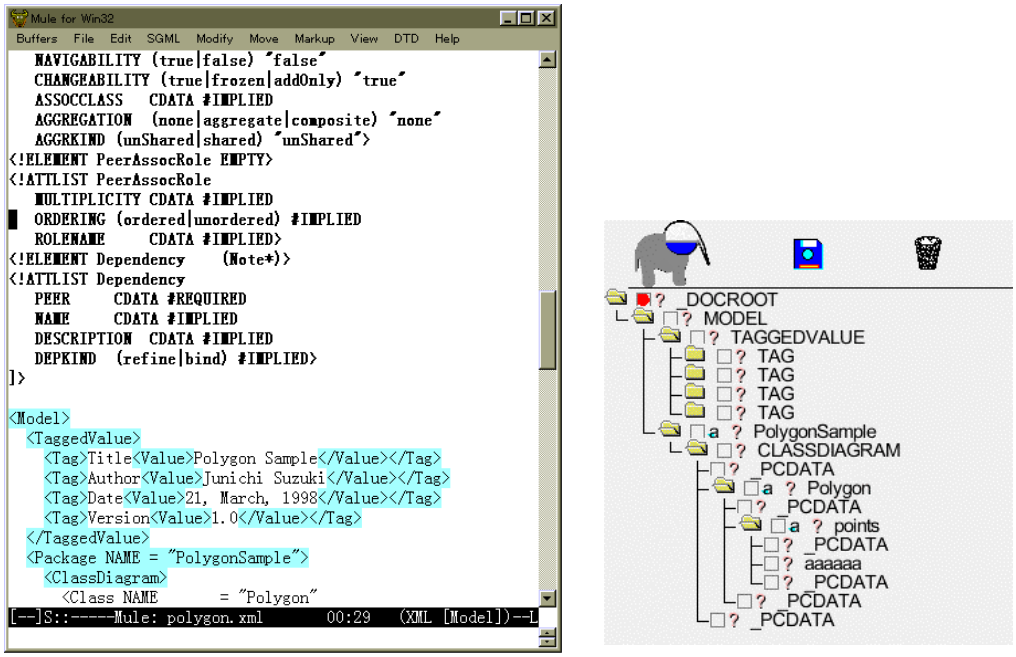

Figure 3: Editing UXF formatted data with an existing XML editor (left), and browsing the hierachical structure of UXF elements with an existing XML browser (right).

4.2 Coding Example of UXF

This section presents an example which shows how we can describe UML models with UXF. Figure 2 shows a sample class diagram, which consists of four classes, `Polygon`, `Point`, `GraphicsBundle` and `Triangle`, two aggregations whereby `Polygon` aggregates `Point` and `GraphicsBundle` , one inheritance whereby `Triangle` derives from `Polygon` and some other indicators. The model information described in this diagram can be described with UXF like in Appendix B. As such, most concepts and constructs in UML class diagram can be mapped to UXF seamlessly.

4.3 Processing UXF

This section outlines how a UXF data might be created, processed and displayed. In every phase, we can reuse various existing XML or SGML tools.

Authoring

UXF data can be created with any text editor because it is a text-based format and human-readable format. In practice, however, it is not primarily intended for the manual use by developers. It is likely that the manual generation of UXF is verbose and error-prone. Instead, it is anticipated that they use CASE tools, conversion tools, UXF editors and other specialized software to recognize UXF. It is practical to increase the productivity that we use any editing tool that helps user's input or conversion tools, described below. In

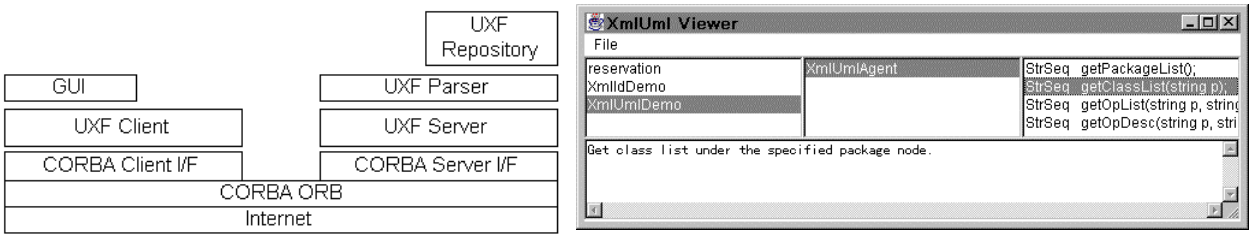

Figure 4: Architecture of our prototype system that allows to share UML models over the Internet (left), and a sample screenshot of the client-side profiling tool (right).

our work, a SGML/XML major mode for Emacs called psgml has been used when UXF data is written from scratch (left of Figure 3).

Conversion

As described above, the authoring process sometimes involves data conversion. It makes the authoring work simple and productive. In the context of UXF, it can be categorized into two schemes; converting legacy documents, program source code or data representation in a development tool (e.g. CASE tool) to UXF, and converting UXF to other formats for printed materials or development tools. UXF allows such conversion programs to be written easily. In our initial work, we have prepared a conversion tool from source code written in Java into UXF, and plan other tools that translate data representation in a CASE tool to/from UXF (see Section 4.4 and 5).

Parsing

Parsing is the process to analyze and validate the syntax of UXF data. XML allows for two kinds of data; valid and well-formed. Validity requires that a data contains a proper DTD and obeys its constraints. Well-formness is a less strict criteria and requires that a data just obeys the syntax of XML. UXF requires a validating parser that confirms to be validity in creating UXF data, and a non-validating parser that confirms just to be well-formness in browsing or delivering the data. We are using a validating parser called nsgmls in creating UXF data, and non-validating parser called Lark in distributing the data.

Distributing

UXF has been designed to distribute UML models precisely over the network environment. It can be used in existing SGML systems that retrieve UXF components and assemble them for the output on the fly. Also, it can be used within the existing Web environment so that a Web browser downloads UXF components and displays them using Java applets. In our work, UXF is transferred within a CORBA based distributed system that manages UML models (see Section 4.4).

Rendering and Browsing

Rendering and browsing involves the delivery of stylesheets or any specialized software for display. In our work, UXF is intended to use stylesheets based on XSL or Java classes associated with each UXF component. Also, UML model information can be browsed with existing XML browsers like Jumbo (Right side of Figure 3).

4.4 Applications

We have developed two prototype applications that process UXF; a source code documentation tool and a Internet-oriented distributed system for managing UML models.

The first application is a documentation tool that parses source code written in Java and generates UXF formatted files. This tool is developed by extending the class `DocumentationGenerator` included in JDK. It allows every construct in Java to be translated to the UXF representation, and reusable for other applications including CASE tools and profiling tools.

The another application is a system that manages UML models in the distributed environment. This system is developed on top of an Java-based ORB (Object Request Broker) that is compliant with CORBA (Common Object Request Broker Architecture), and uses CORBA standard interfaces to transfer UXF data over the Internet. The left of Figure 4 depicts the architecture of this system. It allows developers at distributed places to consistently refer, process and change UML models. A server application parses all the UXF data at the system's start-up time or on the fly, and stores a set of tree structures (i.e. glove objects) composed of UXF data elements in a repository. Client applications include various tools such as normal

Web browsers, simple command-line tools and GUI profiling tools. The right side of Figure 4 is a sample screenshot of a 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 belonged to the package, the right one is the list of operations (i.e. methods) of the class selected in the central pane, and the bottom pane shows the comments (Note and TaggedValue, exactly in the UML term) for each method. This tool accesses a server-side repository and obtains the necessary data to display, according to the user's mouse manipulation.

Based on this mechanism, we can maintain large document collections stored in a server-side repository, which ensures the consistency of information and provides the capabilities like searching, indexing or sorting. This system shows UXF is valuable as a well-structured data format.

These applications given above show straightforward and reasonable ways to share and exchange the UML models between various tools or over the distributed heterogeneous environment, though they are quite simple. We are working on the refinement of these applications and the additional ones, to demonstrate the value of UXF and improve it (see also Section5).

5. Current Project Status and Future Work

UXF initially has eighteen tags and forty two attributes for UML class diagrams. We are refining this DTD, and in the process of defining new ones for collaboration diagrams and statechart diagrams. Also, we are investigating the use of XSL to provide the style sheets for UXF data, and to achieve the multiple media publishing, for example for plain text, HTML, RTF, online help or any other specialized format.

As for UXF applications, we are developing a translator from UXF into importable formats of CASE tools such as Rational Rose™ and into drawing tools. Also, a translator from Python programming language into UXF is planned. With these tools, UML models implemented in multiple programming languages would be fully interoperable with multiple development tools, through a single exchange format. We are also extending our distributed model management system described in Section 4.4 for a fully GUI-based visual profiling tool, and now evaluating it for our internal development projects.

6.Conclusion

This paper addresses how UML models can be interchanged and reused for a wide range of development tools, proposes a solution that provides an application-neutral format called UXF. With UXF, UML models can be distributed universally. We believe UXF shows an important step in exchanging and sharing analysis/design models and indicates a future direction of the interconnectivity between UML compliant tools. Information on the project status can be obtained at http://www.yy.cs.keio.ac.jp/~suzuki/project/uxf.

We would like to thank Peter van Eijk for his help for improving this paper and Kenji Shirane and Yu Tanaka for their help in developing UXF DTD.

Reference

[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] Grady Booch, *Object-Oriented Analysis and Design 2nd Edition*, The Benjamin/Cummings Publishing, 1994

[10] James Rumbaugh et.al., *Object-Oriented Modeling and Design*, Prentice Hall, 1991.

[11] Ivar Jacobson, *Object-OrientedSoftware Engineering: A Use Case Driven Approach*, Addison-Wesley, 1995.

[12] JavaSoft, *JDK 1.1.5 Documentation*, 1997, available at http://www.javasoft.com/products/jdk/1.1/docs/.

[13] Tim Bray et.al. (ed.), *Extensible Markup Language (XML) 1.0*, W3C, available at
http://www.w3.org/TR/1998/REC-xml-19980210

[14] Patrick Ion et.al. (ed.), *Mathematical Markup Language*,W3C, available at http://www.w3.org/TR/PR-math/

[15] CheckFree Corp et.al., *Open Financial Exchange Specification 1.0.2*, available at: http://www.ofx.net/

[16] Sharon Adler et.al., *Initial Proposal for XSL*, available at: http://www.w3.org/TR/NOTE-XSL.html

[17] Tim Bray et.al. (ed.), *Extensible Markup Language (XML): Part 2. Linking, version 1.0*, available at http://www.w3.org/TR/WD-xml-link

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

[19] Rational Software, *UML-Compliant Interchange Format*, OMG document number: ad/97-01-13

Appendix A

```
<!ELEMENT Model              (TaggedValue?, Package*)>
<!ELEMENT TaggedValue (Tag*)>
<!ELEMENT Tag                (#PCDATA, Value*)>
<!ELEMENT Value              (#PCDATA)>
<!ELEMENT Package            (TaggedValue?, Note*, Dependency*, ClassDiagram*)>
<!ATTLIST Package
   NAME CDATA   #REQUIRED>
<!ELEMENT ClassDiagram(TaggedValue?,(Class|Note)*)>
<!ELEMENT Class              (TaggedValue?, (Attr
                                            |Operation
                                            |Generalization
                                            |Association
                                            |Dependency
                                            |Note)* )>
<!ELEMENT Interface          (TaggedValue?, (Attr
                                            |Operation
                                            |Generalization
                                            |Association
                                            |Dependency
                                            |Note)* )>
<!ELEMENT Note        (#PCDATA)>
<!ATTLIST Class
   NAME        CDATA                #REQUIRED
   ABSTRACT    (true|false)         "false"
   VISIBILITY  (public|private)     #REQUIRED
   ACTIVE      (true|false)         #IMPLIED>
<!ELEMENT Attr        (Note*)>
<!ATTLIST Attr
   VISIBILITY  (public|protected|private)   #REQUIRED
   TYPE        CDATA                        #REQUIRED
   NAME        CDATA                        #REQUIRED
   INITVAL     CDATA                        #IMPLIED
   CLASSSCOPE  (true|false)                 "false">
<!ELEMENT Operation          ((Param|Exception|Note)*)>
<!ATTLIST Operation
   VISIBILITY  (public|protected|private)          #REQUIRED
   NAME        CDATA                               #REQUIRED
   RETURN      CDATA                               #REQUIRED
   CLASSSCOPE  (true|false)                        "false"
   CONCURRENCY (sequential|guarded|concurrent)     "sequential"
   EXCEPTION   CDATA                               #IMPLIED>
<!ELEMENT Param              EMPTY>
<!ATTLIST Param
   TYPE        CDATA          #REQUIRED
   NAME        CDATA          #REQUIRED
   DEFAULTVAL  CDATA          #IMPLIED
   DIRECTION   (in|out|inout) #IMPLIED>
<!ELEMENT Exception          EMPTY>
<!ATTLIST Exception
   NAME CDATA          #REQUIRED
   BODY CDATA          #IMPLIED>
<!ELEMENT Generalization     EMPTY>
<!ATTLIST Generalization
   FROM CDATA                       #REQUIRED
   TYPE (public|private|protected)   "public">
<!ELEMENT Association ((AssocRole, PeerAssocRole)| Note*)>
<!ATTLIST Association
   PEER CDATA   #REQUIRED
   NAME CDATA   #IMPLIED>
<!ELEMENT AssocRole          EMPTY>
<!ATTLIST AssocRole
   MULTIPLICITY       CDATA                        #IMPLIED
   ORDERING           (ordered|unordered)          #IMPLIED
   QUALIFIER          CDATA                        #IMPLIED
   ROLENAME           CDATA                        #IMPLIED
   NAVIGABILITY       (true|false)                 "false"
   CHANGEABILITY      (true|frozen|addOnly)        "true"
   ASSOCCLASS         CDATA                        #IMPLIED
   AGGREGATION        (none|aggregate|composite)   "none"
   AGGRKIND           (unShared|shared)            "unShared">
<!ELEMENT PeerAssocRole      EMPTY>
<!ATTLIST PeerAssocRole
```

```
   MULTIPLICITY       CDATA                 #IMPLIED
   ORDERING           (ordered|unordered)   #IMPLIED
   ROLENAME           CDATA                 #IMPLIED>
<!ELEMENT Dependency  (Note*)>
<!ATTLIST Dependency
   PEER        CDATA          #REQUIRED
   NAME        CDATA          #IMPLIED
   DESCRIPTION CDATA          #IMPLIED
   DEPKIND     (refine|bind)  #IMPLIED>
```

Appendix B

```
<?xml version="1.0"?>
<!DOCTYPE Model SYSTEM "UML.DTD">
<Model>
  <TaggedValue>
    <Tag>Title  <Value>Polygon Sample</Value> </Tag>
    <Tag>Author <Value>Junichi Suzuki</Value> </Tag>
    <Tag>Date   <Value>21, March, 1998</Value></Tag>
    <Tag>Version<Value>1.0</Value>            </Tag>
  </TaggedValue>
  <Package NAME = "PolygonSample">
    <ClassDiagram>
      <Class NAME            = "Polygon"
             VISIBILITY      = "public"
             ABSTRACT        = "true">
        <Attr VISIBILITY     = "private"
              TYPE           = "Point"
              NAME           = "points"/>
        <Attr VISIBILITY     = "private"
              TYPE           = "GraphicsBundle"
              NAME           = "bundle"/>
        <Association PEER    = "Point"
                     NAME    = "Contains">
          <AssocRole MULTIPLICITY    = "1"
                     NAVIGABILITY    = "true"
                     AGGREGATION     = "aggregate"/>
          <PeerAssocRole MULTIPLICITY       = "3..*"
                         ORDERING           = "ordered"
                         ROLENAME           = "points"/>
        </Association>
        <Association PEER = "GraphicsBundle">
          <AssocRole MULTIPLICITY    = "1"
                     AGGREGATION     = "aggregate"/>
          <PeerAssocRole MULTIPLICITY       = "1"
                         ROLENAME           = "bundle"/>
        </Association>
      </Class>
      <Class NAME            = "Triangle"
             VISIBILITY      = "public">
        <Generalization FROM = "Polygon"/>
      </Class>
      <Class NAME            = "Point"
             VISIBILITY      = "public">
        <Attr VISIBILITY     = "private"
              TYPE           = "Real"
              NAME           = "x"
              INITVAL        = "0.0"/>
        <Attr VISIBILITY     = "private"
              TYPE           = "Real"
              NAME           = "y"
              INITVAL        = "0.0"/>
        <Operation VISIBILITY= "public"
                   NAME      = "rotate"
                   RETURN    = "void">
          <Param TYPE = "Real"
                 NAME = "angle"/>
        </Operation>
        <Operation VISIBILITY= "public"
                   NAME      = "scale"
                   RETURN    = "void">
          <Param TYPE = "Real"
                 NAME = "factor"/>
          <Exception NAME = "SystemException"/>
        </Operation>
      </Class>
      <Class NAME            = "GraphicsBundle"
             VISIBILITY      = "public">
        <Attr VISIBILITY     = "public"
              TYPE           = "Color"
              NAME           = "color"/>
      </Class>
    </ClassDiagram>
  </Package>
</Model>
```