ソフトウェア工学 120－9
（1998．7．24）

XMLで文法を与えるプログラミング言語

飯島 正

慶應義塾大学理工学部

アブストラクト

XMLは，拡張可能なマーク付け言語（eXtensible Markup Language）である．本論文は，XMLを用いてプログラミング言語を規定するための概念とツールを提供するプロジェクトの概要紹介と，現状報告である．このプロジェクトでは，DTDで与えたプログラミング言語の文法により，XML文書としてプログラムを記述することに加え，プログラムの実行時情報（変数の値など）もXML文書として扱う．これによって，XML自身の持つオブジェクトモデルと適合性の高いオブジェクト指向プログラミング言語を規定できる．本プロジェクトの目標としては，文芸的プログラミングやコンパイラ・コンパイラなどがある．

Programming Languages which are specified by XML

Tadashi Iijima

Faculty of Science and Technology, Keio University

Abstract

The name of XML is an abbreviation of the eXtensible Markup Language. This paper describes an overview and current status of the project which provides a concept and tools using XML to specify programming languages. In this project, a program is described as a XML document, and runtime-information(bindings of variables; that is just an instance in the case of object-oriented programming languages) of the program (a class for the OOPLs) is also described as a XML document. The goals of this project are the so-called literate programming, a compiler construction tool, and so on.

1. はじめに

拡張可能なマーク付け言語（eXtensible Markup Language）XMLは，SGML（Standard Generalized Markup Language）のサブセットであり，W3Cにより規格化が進められているものである[1][2][3]．XMLは，WWW（World Wide Web）で使われているHTML（TyperText Markup Language）の拡張とも位置づけられることから，広く関心を集めている．[1]によれば，XMLの目標は，現在のHTMLと同じように，SGML文書をWeb上で配布，受信，処理することにある．

HTMLと比較した場合，XMLの最大の特徴は，その名の通り「拡張性」にある．XMLの構文は，文書型定義DTD（Document Type Definition）によって定義することができる．つまり，簡単にいえば，HTMLと異なり，XMLでは新しいタグを定義することが出来る．

本論文は，このXMLを用いてプログラミング言語を記述するプロジェクトの構想と考察（ならびに若干の予備実験）に関して紹介するものである．XML自体の規格化が未だ進行中であることから，本稿の内容におけるXML仕様は，その最終版ではないことには

注意されたい.

本プロジェクトは, 主に, XML 文書で(a1)プログラムそのもの(DTD で文法を記述する), と(a2)実行時情報, を XML で記述することを試みる. その結果として, 文芸的プログラミング[11], コンパイラ作成ツール, シリアリゼーション, モーバイル・オブジェクト, ソフトウェア構成支援, 分散オブジェクト, リフレクションといった諸方面への展開を目標としている.

以下では, 続く第 2 章で, プロジェクト概要として, 現在の XML の使われ方(用途)の分類調査と, 本プロジェクトの目的であるプログラムに関連した情報の XML 文書化に関する位置づけについて述べる. 3 章では, 後章の理解に必要な XML の機能に関してまとめる. 第 4 章は, 実際にプログラムと実行時情報を与える XML タグに関して, その一例を示す. まずは, 一般的な手続き型言語の制御構文のタグ化を行い, 変数名, 関数名などに XML 機構を利用することについての考察と予備的な実験に関して述べる. 次に, XML の持つ文書モデルに対応するようにデザインしたオブジェクト指向言語の一記述例について述べる

2. プロジェクトのコンセプト概要

2.1. XML で何を記述するか

本プロジェクトで, XML でプログラミング言語を規定するとは, どういうことを指すのかを簡潔に例示すると, 現時点では,

(a1) プログラム記述
(a2) 実行時情報記述

の二つを考えている. (a1)では, DTD で定義したタグを使いプログラミングを行う. 言いかえれば, **プログラミング言語の構文を DTD で与える**. したがって, DTD が異なれば, 別のプログラミング言語が与えられることになる. つまり, **本プロジェクトは単一のプログラミング言語を記述することを目的とするのではなく, XML を用いていろいろなプログラミング言語のプログラムを記述するためのメタレベルのサポートを目的としている**. しかし, XML の持つ文書オブジェクト・モデルと対応のよい言語であることは, XML の持つ機能を有効に使うのに役立つと同時にその言語への理解を助けるはずである. そうした((分散)オブジェクト指向)言語に関する考察は, 後章で与える. (a2)に関しては, 本研究以前に既に, オブジェクトのシリアリゼーション(実行状態のストリーム出力と入力)や外部化によるデータ転送やモーバイルコードの実現を目的として開発を進めているプロジェクト[10]が知られている. . それについては 2.2.2 節で触れる.

また, 本プロジェクトで規定するプログラミング言語は, その**言語の機能として, 必ずしも XML 文書をデータとして取り扱う機能を備えている**というわけではないことに注意してほしい. が, それを禁止してもいない. そうした機能を備えることにより, リフレクティブなプログラミングへの展開がありうる.

2.2. XML の利用

次に, 上記のような「XML によるプログラミング言語記述」が XML の一般的な用途として考えられるものとどういう関係にあるかについて触れる. 既に述べたように, XML は, SGML 文書の WWW での利用を意図したものである. そこで, 現時点で着目されている XML の実際の用途(今後, もっといろいろな用途が開発されるかもしれないが)を考えてみると, 主に以下の三つに分類できると筆者は考えている.

(b1) 文書構造指定言語
(b2) 共通データ交換フォーマット記述言語
(b3) アーカイバルデータのための柔軟なスキーマ(メタデータ)記述言語.

以下では, 順次, これら用途(関連する研究開発の現状の調査を含めて)と本プロジェクトとの関係について述べる

2.2.1. 文書構造指定言語としてXML

第一の用途である(b1)「**文書構造指定言語**」は，本来，SGMLの目的であった「文書の意味的構造をマーク付けによって規定し，計算機で取り扱うこと」，から生じている．HTMLが，TeX / LaTeXと同じように，出力となる印刷(出版)形態／表示形態を規定しているのに対し，SGML/XMLが規定するのはあくまで文書の意味的な構造である．DTDは，その意味的な構造の型(文書型)として，文書の構文(マーク付の構文)を定義する．

プログラムは，構造化文書に他ならない．その構文は，BNF(Backus Naur もしくはNormal Form)や構文図(syntax chart)で与えられることが多い．本プロジェクトでは，プログラムの構造をXMLで与えることを試みる．DTDの表現能力から，必ずしも読み易く書き易いとは言い難い面があるが，基本的に木構造の構文木を表現することができ，リンク機構によって，木構造から外れた構造も表現できる．特にオブジェクト指向言語の持つセマンティックスに対し，適合性が高い．

このメリットの一つは，**構文解析に既存のXML用パーサを使うことできる**ことである．現在の予備実験では，Java用のXMLパーサLark[8]を用いて簡単なインタープリタを開発している．

本プロジェクトにおいて，XMLを利用するもう一つのメリットとすべく予備的な実験を試みているポイントとして，**文芸的プログラミング(Literate Programming)**[11]への展開がある．文芸的プログラミングは，D.E.Knuthの提唱するソフトウェア・ドキュメントの在り方であり，ツール(WEBシステム)も開発されている．この基本発想は，プログラムと文書整形マークアップとを混在させたドキュメントを作って出版すると同時に，そのドキュメントからプログラムを抽出して実行させるというものである．この文芸的プログラミングと，本プロジェクトとは完全に一致するというものではないが，その発想を引き継ぐことを目標の一つとしている．

本プロジェクトにおけるプログラミング言語の制御構文の多くは，いわゆる文書内容(contents)ではなく，タグとして与えられる．contentsには，プログラムの一部と，自然言語や数式で与えられたコメントが与えられる(そうしたコメントのマークアップは，プログラミング言語とは別のDTDで規定されるようにしたい．現時点では，そうしたコンセプトを示すだけで，そのための実装実験には至っていない)．しかし，基本的な考え方としては，むしろコメントの中にプログラムをタグ付して埋め込むようなイメージで，プロジェクトのコンセプトを固めつつある．

DTDで与えられた構文要素に対して，対応する意味を定義する言語はXMLではない．例えば，出力形態を規定するスタイル情報はHTMLやTeXにおいては，その言語構文から分離されていないが，XMLでは，スタイル情報をDTDが定義する(文書の意味構造)の構文要素に対応する，一種の意味情報とみなすことができる．スタイル情報の指定は，XMLとは別の言語で与える．例えば，XSL(eXtensible Style language)[4]，ISOによるDSSSL(Document Style Semantics and Specification Language)[5]や，CSS(Cascading Style Sheets)[6]がその候補として挙げられている[2]．こうした発想に基づいて，**プログラミング言語の実行系ためのセマンティックス(意味)を規定する言語を開発すること**も本プロジェクトの課題の一つである．現在の予備実験では，構文要素に対応する意味記述は，Java言語で書き下ろしているが，**構文要素に対応するアクションルールを与える意味記述言語をあたえることにより，YaccやBison[12]のようなコンパイラ・コンパイラ・ツールを与えることができる**と考えている．現在，その準備作業中である・

また，制御構文をタグで与えることにより，そのブラウザ上での表示手段の選択は上記のスタイル情報に任されることになる．たとえば，プログラムの制御構造の表示方法として，

PADのような構造化フローチャートを用いることも，この枠組みの上で扱うことが出来るようになる．

2.2.2. データ交換フォーマット指定言語としてのXML

第二のXMLの用途である(b2)「**共通データ交換フォーマット記述言語**」というのは，主に，インターネット上でのアプリケーション間のデータ交換を想定している．たとえば，Webエージェントが WWWと情報交換するのに，XMLを用いることが考えられている[9]．

インターネットでやり取りされるデータでも，実際の転送構文はビット列であったとしても，それをやりとりするアプリケーションでは，より抽象度の高い抽象構文で解釈・処理される．そうした，抽象構文を規定する言語として**プロトコル構文規定言語** ASN.1(Abstract Syntax Notation one)があり，現実にアプリケーションのレベルに近い，高位プロトコルで広く使われている[13]．XMLでやり取りするデータは，基本的にテキスト(実際には，XML実体に格納されるデータはテキストかバイナリ[1])であるが，これは，アプリケーション・レベルでは，むしろ取り扱いがし易いといえるかもしれない．

本プロジェクトとの関係という観点から，この共通データ交換フォーマット言語という用途を考えてみる．アプリケーション間で交換されるデータとして，XML文書が使われる．既に述べたように，本プロジェクトでは，XML文書として(a1)「プログラムそのもの」と(b2)「実行時情報」を扱う．

プログラムをXML文書とした場合，プログラムがアプリケーション間で転送されるということになる．具体的には，アプレットのようなプログラムのダウンロードが考えられる．ダウンロードされるプログラムそのものではなく，そのパッケージの記述(実際にはプログラムに限らずソフトウェア)を交換するためのXML文書の規約として，OSD(Open Software Description Format)[14]がある．これは，そのパッケージに関わるモジュールの依存関係も含めて記述することが出来る．

より野心的な試みとしては，C++とjavaといった複数の言語間のトランスレーションの中間言語とすることが考えられるが，一般に，複数のプログラミング言語間の中間言語を開発することは，言語間の意味的なギャップから難しい．分散オブジェクト環境の仕様であるCORBA (Common Object Request Broker Architecture) における IDL(Interface Definition Language) [15]のような限定した範囲であれば，より実現性は高い．

プログラムの実行時情報をXML文書として扱い，それをインターネットで転送する場面としては，モーバイル・オブジェクト(ないしはモーバイル・エージェント)での利用がある．オブジェクトの実行時情報とは，そのオブジェクトが持つ属性値やプログラムカウンタ，内部のメソッド(関数)呼出しスタック，メッセージキュー，オープンしているファイルなどのリソースなどが考えられる．属性値以外の実行時情報がどれだけ取り扱えるかによって，転送前後の実行を継続させることが出来るかどうかが決まる．オブジェクトの実行時情報(属性値)とプログラムを出力するシリアリゼーションに，XMLを用いる先行研究としてJava(JavaBeans)のためのCOINS[10]がある．

2.2.3. アーカイブのための柔軟なスキーマ(メタデータ)記述言語としてのXML

前節のデータ交換フォーマット記述言語は，このスキーマ(メタデータ)記述言語に含めてもよいかもしれない．が，前節の用途がデータ交換を目的としていたのに対し，この節の用途は，保存(アーカイブ)を目的としている．筆者による「文法に基づくデータモデル」[16]は，この用途のカテゴリである．プログラムを対象とした場合にはライブラリ・データベース(CORBA用語ではインプリメンテーリョン・リポジトリ)，実行時情報(属性値)を対象

にした場合には，オブジェクト指向データベースに対応する．

3. XML[1][2][3][7]

3.1. XML とは

XML とは，拡張可能なマークアップ言語(eXtensible Markup Language) である．拡張可能なマーク付け言語 (eXtensible Markup Language) XML は，SGML (Standard Generalized Markup Language) のサブセットであり，W3C により規格化が進められている[1]．XML は，WWW (World Wide Web) で使われている HTML (TyperText Markup Language) の拡張とも位置づけられることから，広く関心を集めている．[1]によれば，XML の目標は，現在の HTML と同じように，SGML 文書を Web 上で配布，受信，処理することにある．

HTML と比較した場合，XML の最大の特徴は，その名の通り「拡張性」にある．XML の構文は，文書型定義 DTD (Document Type Definition) によって定義することができる．

3.2. XML 文書の構造

XML 文書は，論理構造と物理構造を持つ．物理構造は，一つ以上のファイルといったような記憶単位から構成され，その単位を実体(エンティティ)という．論理構造としては，一つ以上の要素を含み，要素は開始タグと終了タグで囲まれることで示される(内容を持たない空要素は開始タグのみでしめされる)．開始タグは，基本的には，名前を<と>で囲んだものであるが，名前の後ろに属性名と属性値を=で結んだ対を複数個置くことができる．

開始タグの例:<名前 属性名=値>

終了タグは，名前を</と>で囲んだものである．

終了タグの例:</名前>

空要素のためのタグは開始タグだけであるが，特に，空要素のためのものであることを示すために<と/>で囲む．

3.3. 文書型定義 DTD

XML 文書の，タグの構文(等)を決めるのが文書型定義 DTD(Document Type Definition)である．これは，XML 文書内部で宣言しても，別のファイルに独立させてもよい．

DTD には，要素型宣言，属性リスト宣言，エンティティ宣言，記法宣言が含まれる．要素型宣言では，その文書にどのような要素がどのような順序でどのような個数，出現しうるかを規定する．属性リスト宣言は，要素がとりうる属性とその型，デフォールト値などを規定する．エンティティ宣言では，取り込みたい外部のファイルやその参照名を指定する．

要素型宣言
<!ELEMENT 要素名 内容モデル>

内容モデルでは，要素名で与えたタグの内側(内容(cotents))の中に置くことが出来る要素の種類，個数，順序などに関して規定する．

例	内容モデルの説明
<!ELEMENT X (a)>	a が出現する
<!ELEMENT X (a?)>	a が 0 回か 1 回出現
<!ELEMENT X (a+)>	1 回以上 a が出現する
<!ELEMENT X (a*)>	0 回以上 a が出現する
<!ELEMENT X (a,b)>	a と b が順に出現
<!ELEMENT X (a|b|c)>	a か b か c が出現

属性リスト宣言は以下のようになる．これにより，要素が持つ属性とその型(属性値候補)ならびにデフォールト値を規定できる．

属性リスト宣言
<!ATTLIST 要素名 属性名 属性値候補 “デフォールト値” … 属性名 属性値候補 “デフォールト値”>

エンティティ宣言では，ファイル等の外部エンティティ等を宣言する．ここで，URL を使うことが出来る点に注意して欲しい．

エンティティ宣言

```
<!ENTITY エンティティ名 SYSTEM
        "ファイルの URL">
<!ENTITY エンティティ名 PUBLIC
        "公開識別子" "ファイルの URL">
```

3.4. リンク機構

HTML は，そのリンク機能が極めて特徴的であった．XMLも，規格作業中ではあるが，HTML のリンク機能を包含するより強力なリンク機能が幾つか提案されている(XLink[], XPointer)．もっとも，リンク機能のための構文は XML の構文に関する規定を逸脱するものではない(だろう)．そこで，リンクというセマンティックスに関する規定は，スタイル情報に類するものであり，XML の規格無いというより関連規格として審議されている．

XML のリンク要素は XML:LINK 属性によって認識される．

単純リンク

```
<要素名 xml:link="simple"
        href="ロケータ" />
```

```
<要素名 xml:link="simple"
        href="ロケータ" >
        ....
</要素名>
```

また，多項関係などを表現できる拡張リンクも提案されいたり，役割(role)を規定する属性(content-role 等もある)が提案されているなど，実体関連図(ER-図)やオブジェクト指向分析などで扱われてきた関連の記述能力を有している．

4. プログラミング言語

4.1. 命令型言語の設計と実装

まずは，手始めに簡単な命令型プログラミング言語の記述例からはじめる．最初に，制御構造のタグ化を行い，次に名前参照部分にリンク機構を使った試みについて，紹介する．4.2 節では，更に，XML のモデルに対応の高いオブジェクト指向プログラミング言語の記述例に関して述べる．

4.1.1. 制御構造

まずは，制御構造のマーク付けを試みる．簡単な構文を考えると，例えば，以下のようなDTD ができる．

```
<!DOCTYPE PROG [
  <!ELEMENT PROG ((DECL)*,(STMT)*)>
  <!ELEMENT STMT (ASSIGN|PRINT|IF)>
  <!ELEMENT PRINT (#PCDATA)>
  <!ELEMENT ASSIGN (#PCDATA)>
  <!ELEMENT IF (COND,THEN,ELSE?)>
  <!ELEMENT THEN (STMT)*>
  <!ELEMENT ELSE (STMT)*>
  ]>
```

これを用いると，例えば，以下のようにプログラムを記述することが出来る．

```
<IF><COND> I == 100 </COND>
  <THEN>
   <STMT><PRINT>"Yes"</PRINT></STMT>
  </THEN>
  <ELSE>
   <STMT><PRINT>"No"</PRINT></STMT>
  </ELSE>
  </IF>
```

但し，この場合，制御構造は，タグとしてしか書かれないので，ブラウジングのためには，各タグの意味をしめすような表示を出力するスタイルが必要である．

4.1.2. リンクの利用

前節で与えた DTD の場合，代入文(ASSIGN 要素)や条件式(COND 要素)の中に入る「式」に関しては，まったく，検査していない．これは，記述があまりに煩雑になるのをさけるためである．しかし，式中の変数や関数をマーク付けしたいという要求もありうる．変数宣言や関数定義はそれぞれ一つの要素となる．ここで，変数や関数を参照しているところに，リンクを使うことによって，ブラウザ上で，特定の名前の部分をクリックすると対応する要素の宣言(もしくは定義)部分にジャンプす

るというオペレーションを，XML のセマンティックス(というより HTML のセマンティックスというべきか)で説明できるようになる．
具体例として，以下のような式(演算子は infix 表現)

```
変数名 1 := 変数名 2 + 定数
```

に対し，以下のような XML 表現(演算子は prefix 表記)が考えられる．

```
<WRITE href="#id(変数名 1)">
 <PLUS>
  <READ href="#id(変数名 2)"> コメント
  </READ>
  <CONST id="定数"/>
 </PLUS>
  </WRITE>
```

ここで，WRITE や READ のような変数参照を行うタグにはデフォールト値として，xml:link 属性を与えておく．例えば，DTD の属性リスト宣言で，

```
<!ATTLIST READ
        xml:link CDATA #FIXED "simle"
        href     CDATA #REQUIRED>
```

というように単純リンクであることをデフォールト値で与えておくとよい([7]を参考にした)．変数の場合には，リンク先はその変数の宣言部となる．例えば，以下のように記述される

```
<VAR id="変数名 1" type="...">
            コメント </VAR>
```

関数呼出しに関しては，例えば以下のようなものが考えられる．

```
<CALL href="#id(関数名)">
  <ARG> ... </ARG>
        ...
  <ARG> ... </ARG>
</CALL>
```

ここでも変数参照と同じように関数名から，関数定義へのリンクを持つように，CALL の DTD を与えることが出来る．

しかし，こうしたリンクを記述する負担をプログラマに課したのでは，読み易さと書き易さを損なってしまう．そこで，infix 記法で普通に書き下ろされた数式を，上記のような prefix 記法の表現に変換するトランスレータを Bison 等のコンパイラ・コンパイラツールで手軽に記述する実験を行っている．

4.2. オブジェクト指向プログラミング

クラスつまりプログラムは，一つのエンティティ，すなわち例えば単独のファイルに格納された XML 文書とする．

```
<CLASS id="クラス名">
   <EXTENDS id="クラス名 2" />
 <VAR id="変数名 1" type="...">
          コメント </VAR>
 <VAR id="変数名 2" type="...">
          コメント </VAR>
 <METHOD id="変数名"> コメント
   <RETURN type="..."/>
   <PARAM id="仮引数名 1">
            コメント</PARAM>
   <PARAM id="仮引数名 2">
            コメント</PARAM>
   <DEF> ... </DEF>
 </METHOD>
</CLASS>
```

クラスの指定には，そのクラス記述を持ったファイルの URL を使う．

```
<NEW href="クラスを指定する URL">
  <ARG> ... </ARG>
        ...
  <ARG> ... </ARG>
</NEW>
```

オブジェクト(インスタンス)を，実行時情報を蓄積する一つのエンティティ，すなわち例えば単独のファイルに格納された XML 文書とする．すなわち，上記の NEW タグの要素が解釈実行されると，下記の OBJECT タグの要素を持ったエンティティ(XML 文書ファイル)が一個生成されるということになる．こうすることで，この XML 文書の保存が，いわゆるシリアリゼーションに相当することになり，その配送によって，モーバイルオブジェクト(モーバイルエージェント)が実現できることになる．但し，下記の例は，属性値しか保存しておら

ずプログラムカウンタなどは保存していないので，継続実行には対応していない．

```
<OBJECT>
  <CLASS href="クラスの URL" />
  <VAR id="変数名"> 値 </VAR>
  <VAR id="変数名"> 値 </VAR>
</OBJECT>
```

オブジェクトの指定にURILを用いることにより，ネーミングに関していえば，基本的に，いわゆる分散オブジェクトまで，シームレスに取り扱っていることになる．VisiBroker 等のCORBA 準拠の ORB や RMI (Remote Method Invocation) のような分散オブジェクト環境でもサーバのネーミングに URL を用いるものがあるが，上記の例ではそれが自然に取りいれられている．

5. おわりに

本稿では，XML を用いたプログラミング言語記述のためのプロジェクトの概要とプロジェクト立ち上げにあたって予備的に行った実験の現状を紹介した．XML の用途は，今後，更にいろいろと広がっていくかもしれないが，今回は，その一つとしてプログラミング言語記述(文法記述と実行時データ記述)への展開を試み，XML の現時点で知られている諸用途との関連を考察した．

参考文献

[1] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen (Eds.): Extensible Markup Language (XML) 1.0, REC-xml-19980210, W3C Recommendation 10/02/98, http://www.w3.org/TR/REC-xml

[2] Eve Maler, and Steve DeRose(Eds.): "XML Linking Language (XLink), " (03/03/98) http://www.w3c.org/TR/1998/WD-xlink-19980303

[3] XML Pointer Language (XPointer), http://www.w3c.org/TR/1998/WD-xptr-19980303

[4] W3C:"Web Style Sheets, " (04/23/98) http://www.w3.org/Style/

[5] W3C:"Cascading Style Sheets,"(08/14/97) http://www.w3.org/Style/CSS/

[6] Sharon Adler 他; "A Proposal for XSL," (08/27/97) http://www.w3.org/TR/NOTE-XSL-970910.html

[7] XML/SGMLサロン:「標準XML完全解説」，技術評論社(1998)

[8] Tim Bray: " An Introduction to XML Processing with Lark and Larval," (01/05/98) http://www.textuality.com/Lark/

[9] Jon Bosak: "XML, Java, and the future of the Web," (03/10/97) http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm,

[10] Bill La Forge, "Coins: Tightly Couples JavaBeans and XML Elements," (5/13/98), http://www.camb.opengroup.org/~laforge/coins/, http://www.jxml.com/coins/

[11] Donald E. Knuth(有澤誠(訳)):「文芸的プログラミング」，アスキー出版局(1984)

[12] Charles Donnelly and Richard Stallman (訳: 玉井浩): 「YACC互換パーサ生成系Bison」，(1998)

[13] 森野和好，戸部美春:「プロトコル構文規定言語-ASN.1」，カットシステム,(1994)

[14] Arthur van Hoff, Hadi Partovi, Tom Thai: "Open Software Description (OSD)," (08/13/97) http://www.w3.org/TR/NOTE-OSD.html

[15] OMG:" CORBA/IIOP 2.2 Specification," (02//98),http://www.omg.org/corba/corbiiop.htm

[16] 飯島正:文法に基づくデータモデル，情報処理学会第52回(平成8年前期)全国大会(Vol.4),pp.261-262