Отображение модели данных XML в объектную модель языка СИНТЕЗ

Осипов М.А.,
Московский Государственный
Университет им. М.В.Ломоносова, ф-т ВМиК
e-mail: osipov@newtech.ru

Мачульский О.Л.,
Институт Проблем Информатики
Российской Академии Наук,
e-mail: machulsk@synth.ipi.ac.ru

Калиниченко Л.А.,
Институт Проблем Информатики
Российской Академии Наук,
e-mail: leonidk@synth.ipi.ac.ru

Abstract

Данная работа выполняется в рамках проекта РФФИ "Создание интегрированных библиотек на основе неоднородных распределенных электронных коллекций научной информации", грант 98-07-91061. Одной из задач, решаемых в рамках этого проекта, является приведение моделей данных разнородных информационных ресурсов к каноническому виду. В качестве таких ресурсов могут выступать как структурированные данные (например, реляционные или объектно-ориентированные базы данных), так и слабоструктурированные (например, коллекции HTML-документов) В качестве канонической модели в посреднике, реализующем доступ к электронным коллекциям, используется язык СИНТЕЗ. Одной из задач проекта является отображение моделей данных электронных коллекций в каноническую модель.

В данной статье расматривается отображение структуры документов в формате XML в каноническую модель. Структура XML документов определяется посредством языка описания типов документов (DTD). На основе DTD строится спецификация на языке СИНТЕЗ, при этом каждая декларация элемента в DTD отображается в тип данных языка СИНТЕЗ.

1 Введение

В связи с интенсивной разработкой консорциумом W3C технологий, связанных с XML, можно ожидать, что в ближайшее время они будут активно использоваться в электронных библиотеках в качестве одного из средств внешнего представления данных. Причины этого раскрываются при рассмотрении двух основных подходов к представлению текстовой информации. Эти подходы условно можно назвать контентной разметкой (content-markup) и форматной разметкой (layout-markup).

При форматной разметке управляющие конструкции языка разметки слабо связаны со структурой документа и определяют исключительно особенности форматирования (визуального представления). Ярким примером такого подхода является язык разметки HTML, получивший широкое распространение. При контентной разметке управляющие конструкции языка разметки жестко связаны с логической структурой документа, и именно они задают ее. Примером контентной разметки является язык разметки текстов, используемый в издательской системе LaTeX.

Несмотря на широкую распространенность подходов, ориентированных на форматную разметку, их применение в электронных библиотеках как средства внутреннего представления данных не дает никакого выигрыша по сравнению с использованием обычного неформатированного текста и требует применения техники обработки полнотекстовой информации.

Достоинства технологий, ориентированных на контентную разметку, в свете задач, решаемых при проектировании и разработке электронных библиотек, заключаются в том, что в случае их использования появляется возможность не только представлять, но и хранить информацию в структурированном виде. Кроме того, появляется возможность более оптимально решать такие специфичные для электронных библиотек задачи, как индексирование массива документов и поиск.

Язык XML [1], позволяющий с использованием DTD (document type declaration) эффективно создавать языки разметки, является в рамках задач, решаемых в сфере электронных библиотек, чрезвычайно удобным средством. В силу того что технология XML стандартизована, появляется возможность использования стандартных программных библиотек для работы с XML данными, что будет способствовать повышению интероперабельности разрабатываемых программных систем и сокращению времени разработки.

Типичная электронная библиотека, представленная в WWW, может выглядеть следующим образом: данные хранятся в СУБД или непосредственно на файловой системе сервера, а взаимодействие с пользователем, работающим с электронной библиотекой, осуществляется при помощи WWW сервера приложений, преобразующего данные из СУБД в формат HTML, который и отображается пользовательским броузером. К сожалению формат HTML не передает, а скорее скрывет те модели данных, которые положены в основу электронной библиотеки. Вследствие этого усложняется задача построения систем, объединяющих множество электронных библиотек и предоставляющих пользователю интегрированный доступ к данным информационным ресурсам.

Задача "отыскания" схемы данных электронной кол-

Первая Всероссийская научная конференция
ЭЛЕКТРОННЫЕ БИБЛИОТЕКИ:
ПЕРСПЕКТИВНЫЕ МЕТОДЫ И ТЕХНОЛОГИИ,
ЭЛЕКТРОННЫЕ КОЛЛЕКЦИИ
19 - 21 октября 1999 г., Санкт-Петербург

лекции, скрытой в HTML интерфейсе решается в проекте Araneus [6, 7].

Использование XML интерфейса электронных коллекций позволит более эффективно создавать и поддерживать коллекции неоднородных слабоструктурированных информационных ресурсов.

Описанные выше причины должны привести к появлению большого количества электронных коллекций, построенных с использованием XML технологий.

Представляется целесообразным отображать структуру данных XML документов в одну из существующих моделей данных (реляционная, объектная и др.), чтобы затем иметь возможность работать с коллекциями XML документов как с базами данных. Аналогичный подход к работе со слабоструктурированными данными рассматривается в статье [5]. В этой работе предлагается рассматривать дерево разбора, возникающее при компиляции документов, а затем каждому узлу дерева разбора ставить в соответствие объект в объектно-ориентированной базе данных.

Одна из задач, возникающих при реализации посредника, осуществляющего доступ к разнородным электронным коллекциям, состоит в построении отображений моделей данных электронных коллекций в каноническую модель. На основе данных отображений генерируются адаптеры, обеспечивающие доступ к электронным коллекциям.

Задача настоящей работы состоит в том, чтобы осуществить отображение модели данных XML в объектно-ориентированную модель языка СИНТЕЗ, играющего роль канонической модели данных в интегрированной электронной библиотеке, создаваемой на основе неоднородных распределенных электронных коллекций научной информации.

Структура статьи следующая: в разделе 2 в сжатом виде приводится описание языка DTD, представляющего собой основное средство для порождения языков разметки. В разделе 3 вкратце рассматриваются особенности объектной модели и языка спецификации информационных ресурсов СИНТЕЗ, который используется в проекте. В разделе 4 описываются принципы отображения типа документа, определяемого DTD в типы данных на языке СИНТЕЗ. В разделе 5 приводится пример отображения DTD, построенного при помощи генератора адаптеров w4f [8] из электронной коллекции музея Уффици [9], в каноническую модель данных.

2 Язык DTD, как язык определения данных

Каждый XML-документ содержит заголовок, в котором определяются версия XML, тип документа (document type definition) и также могут определяться язык документа и прочие его параметры. Непосредственно после заголовка следует тело документа, включающее корневой элемент, который в свою очередь может включать другие элементы.

Пример 1. Простейший XML-документ выглядит так:

```
<?XML version="1.0"?>
<!DOCTYPE greeting SYSTEM "hello.DTD">
<greeting>Hello, world!!!</greeting>
```

В этом документе строка

```
<!DOCTYPE greeting SYSTEM "hello.DTD">
```

является декларацией типа XML-документа, который определяет грамматику для класса документов.

Тело документа состоит из единственного элемента `greeting`, являющегося корневым. В общем случае, каждый элемент содержит список атрибутов, а также может включать другие элементы. Атрибуты элемента имеют имя и значение. Декларация типа документа определяет правила, по которым формируются списки атрибутов у элемента, а также правила включения элементов друг в друга.

Рассмотрим подробнее декларацию типа документа. Имя, следующее за словом `DOCTYPE`, соответствует имени описанного в DTD элемента; в нашем примере это `greeting`. Этот элемент теперь определяет тип документа и называется корневым элементом (root element). `hello.DTD` — имя файла, содержащего DTD.

Пример 2. Декларация типа элемента:

```
<!ELEMENT greeting (#PCDATA)>
```

Декларация элемента определяет тип элементов и то, какие элементы могут содержаться в данном элементе. В данном случае было специфицировано, что содержимым элемента `greeting` являются строковые данные. Содержимым элемента называется то, что находится в XML документе между открывающим и закрывающим тэгами.

Пример 3. Вложенные элементы:

```
<p>
    <h1>Заголовок 1</h1>
    <h2>Заголовок 2</h2>
</p>
```

В примере 3 элемент `<p>` содержит вложенные элементы `<h1>` и `<h2>`.

Пример 4. Декларация элемента вида `EMPTY`:

```
<!ELEMENT br EMPTY>
```

В примере 4 был специфицирован элемент с именем `br`, содержимое которого пусто, то есть в XML документе этот элемент используется следующим образом: `<br/>`. Символ "/" в тэге означает что при разборе XML документа, парсеру не следует искать закрывающий тэг.

Пример 5. декларация элемента типа `ANY`:

```
<!ELEMENT font ANY>
```

В примере 5 был специфицирован элемент с именем `font`, в качестве содержимого которого может использоваться любая комбинация других элементов и строковых данных.

Пример 6. Декларация атрибутов для элемента form:

```
<!ELEMENT form
    (#PCDATA|input|textarea|select)*>
<!ATTLIST form
    method (GET|POST) #IMPLIED
    action CDATA #IMPLIED>
```

Декларация атрибутов элементов используется для ассоциации пар имя-значение с элементами. В примере 6 конструкция

```
<!ATTLIST form
    method (GET|POST) #IMPLIED
    action CDATA #IMPLIED>
```

является декларацией атрибутов для элемента `form`. О принадлежности этого списка к классу элементов form говорит имя, указанное после слова `ATTLIST`.

2.1 Параметры-сущности

Параметры-сущности в DTD являются аналогом макроопределений в языках программирования и не влияют на логическую структуру документа. Далее приведен синтаксис определения сущностей в DTD.

Декларация параметра-сущности

```
<!ENTITY' S '%' S Name S PEDef S? '>
```

Здесь `S` означает наличие последовательности пробелов, табуляций и других символов-разделителей, `Name` означает имя сущности, а `PEDef` соответственно значение сущности. Этим значением может быть любая строка символов, допустимых в DTD. Подстановка значения параметра-сущности происходит при разборе DTD.

3 Язык СИНТЕЗ как средство описания абстрактных типов данных

Одним из основных понятий в языке СИНТЕЗ является понятие типа. Тип определяет свойства и интерфейс объектов данного типа. Спецификация типа состоит из спецификаций атрибутов, функций и инвариантов. Инвариант представляет собой утверждение, которое на экземплярах данного типа является истинным при любых допустимых значениях атрибутов. С каждым определением атрибута может быть связан метаслот для задания дополнительных характеристик атрибута (инвариантов, начальных значений, и др.). Более подробное описание языка СИНТЕЗ можно найти в [4].

В качестве примера спецификации типа на языке СИНТЕЗ можно рассмотреть следующее:

```
{ My_Type;
  in : type;
  attribute1 : integer;
  metaslot
    init: 0
  end
  attribute2 : string;
  metaslot
    invar1 : invariant, {{obligatory}}
  end
  attribute3 : integer
};
```

Здесь специфицирован тип `My_Type`, содержащий атрибуты `attribute1`, `attribute2` и `attribute3`. Атрибуты `attribute1` и `attribute3` имеют тип `integer`, и `attribute2` имеет тип `string`.

С атрибутом `attribute1` связан метаслот, задающий начальное значение данного атрибута. С атрибутом `attribute2` связан метаслот, содержащий инвариант `invar1`, в котором содержится утверждение `obligatory`, означающее что значение атрибута `attribute2` должно быть определено.

4 Преобразование DTD в объектную модель данных

Отображение DTD в объектную модель должно сохранять информацию, содержащуюся в DTD, а значит и логическую структуру XML-документов, представленную в DTD.

Основной идеей отображения является представление каждого типа элемента в DTD типом данных на языке СИНТЕЗ со списком атрибутов соответствующих атрибутам элемента DTD.

Также в задачу преобразования входит отображение типов атрибутов DTD, таких как тип строковый, тип токена, тип перечисления в типы языка СИНТЕЗ.

4.1 Отображение спецификации содержимого класса элементов.

Содержимое элементов, определяемое в DTD, отображается в атрибут `content` типа СИНТЕЗа, имеющий тип, определяемый спецификацией содержимого элементов. При этом конструкции грамматики, специфицирующие содержимое элемента, отображаются в типы СИНТЕЗа. Ниже показано соответствие между конструкциями грамматики, используемой в DTD для определения содержимого элементов, и базовыми типами СИНТЕЗа.

Содержимое элементов вида `EMPTY` используется для определения элементов, не имеющих содержимого. Поэтому в типах, сгенерированных по элементам вида `EMPTY`, если и будут содержаться атрибуты, они будут относиться к списку атрибутов этого элемента.

Элементу вида `ANY`, ставится в соответствие тип с атрибутом типа `XML_ANY`, где `XML_ANY` является супертипом для всех типов, получившихся в результате отображения деклараций элементов в типы языка СИНТЕЗ.

Конструкция выбор вида:

```
( name1 | name2 )
```

отображается в базовый тип `union` следующего вида:

```
{union U; name1, name2;}
```

Последовательность элементов:

```
( name1,name2 )
```

отображается в набор атрибутов типа языка СИНТЕЗ:

```
{
  xml_Attribute_name1:name1;
  xml_Attribute_name2:name2;
}
```

Такие конструкции, как

```
name*
name?
```

означающие последовательность элементов и необязательное присутствие элемента отображаются в

```
{sequence;
  type_of_element: {name}
}

{union U;
  name,
  Tnone;
}
```

Если в декларации типа документа встречается описание атрибута, содержащего строковые данные, например, такого вида:

```
<!ELEMENT p (#PCDATA)>
```

то ему в соответствие ставится тип в СИНТЕЗе:

```
{XML_p;
 in:type;
 supertype: XML_ANY;
 content:{String};
}
```

При отображении спецификации содержимого элемента в типы СИНТЕЗа, нужно представить грамматику, описывающую элемент, в виде дерева, листьями которого являются имена типов элементов, а узлами являются слова: 'or', 'and' и 'seq'. При этом некоторые символы грамматики, такие как '?' и '+' не используются в дереве разбора, так как они выразимы с помощью других символов. Например, конструкцию (name?) можно интерпретировать как

```
 ( name | Tnone)
```

После построения дерева разбора преобразование спецификации содержимого не представляет особого труда. Действительно, при спуске рекурсивно от корня дерева к листьям конструируется определение типа СИНТЕЗа.

4.2 Отображение списка атрибутов

Каждый атрибут из списка атрибутов, связанного с данным классом элементов DTD, отображается в атрибут абстрактного типа данных языка СИНТЕЗ, соответствующего этому классу элементов.

```
AttlistDecl ::=  '<!ATTLIST' S Name AttDef* S? '>'
AttDef ::=  S Name S AttType S DefaultDecl
```

Декларация атрибута состоит из объявления имени и типа атрибута, а также декларации того, имеет ли атрибут значение по умолчанию и, если имеет, то каково поведение атрибута по умолчанию.

4.2.1 Типы атрибутов

Грамматика

```
AttType ::=  StringType | TokenizedType
           | EnumeratedType
StringType ::=  'CDATA'
TokenizedType ::=  'ID'
           | 'IDREF'
           | 'IDREFS'
           | 'ENTITY'
           | 'ENTITIES'
           | 'NMTOKEN'
           | 'NMTOKENS'

EnumeratedType ::=
        '(' S? Nmtoken (S?
        '|' S? Nmtoken)* S? ')'
```

Пример

```
<!ATTLIST termdef
    id      ID      #REQUIRED
    name    CDATA   #IMPLIED>
<!ATTLIST list
    type
      (bullets|ordered|glossary)
      "ordered">
<!ATTLIST form
    method  CDATA   #FIXED "POST">
```

Правило отображения Для отображения атрибутов строкового типа (`StringType`) в СИНТЕЗ, создается специальный тип `DTD_CDATA`, после чего этим типом будут типизированы те атрибуты в языке СИНТЕЗ, которые соответствуют атрибутам элементов типа `CDATA` в DTD.

Что же касается типов токенов (`TokenizedType`), то каждый тип токена отображается в отдельный абстрактный тип СИНТЕЗа с таким же именем, как и соответствующий ему тип атрибута в XML. Семантика XML предполагает некоторые ограничения на использование атрибутов, и эти ограничения также переносятся в СИНТЕЗ в виде метаинформации. Например, атрибут типа `ID` должен быть различен у всех элементов того класса элементов, к которому принадлежит данный атрибут. Это ограничение изображается в языке СИНТЕЗ с помощью встроенного утверждения уникальности (`{Unique;}`).

Рассмотрим подробнее все типы атрибутов и их образы в спецификации на языке СИНТЕЗ.

Ниже представлено соответствие типов атрибутов элементов в DTD, таких как `ID` и `IDREF`, абстрактным типам СИНТЕЗа:

```
{DTD_ID;
  in:type;
  value: string;
  metaslot
    xml_Assertion_ID:
      invariant,{unique;};
  end;
};


{ DTD_IDREF;
  in: type;
  value:DTD_ID;
};
```

В спецификации типа `DTD_ID` используется утверждение метауровня, а именно, метаслот. Это утверждение представляет ограничение на использование атрибутов, которое предполагает семантика XML. В спецификации на СИНТЕЗе, соответствующей атрибутам типа `ID`, используется встроенное утверждение уникальности.

Отображение остальных типов токенов в СИНТЕЗ не представляет никакой сложности. Так, например, тип `IDREFS` представляет собой массив с элементами типа `IDREF`. Тип `NMTOKEN` является строковым типом с некоторым ограничением на символы в строке, а тип `NMTOKENS` соответственно является массивом с элементами типа `NMTOKEN`.

При отображении перечислимых типов атрибутов (`EnumeratedType`)в язык СИНТЕЗ для каждого из содержащихся в DTD перечислений необходимо сгенерировать тип данных enum, после чего типом данных атрибутов становится это перечисление.

4.2.2 Значения по умолчанию для атрибутов

Грамматика

```
DefaultDecl ::=
  '#REQUIRED' |
  '#IMPLIED'  |
  (('#FIXED' S)? AttValue)
```

Пример использвания спецификации поведения атрибутов по умолчанию приводился выше.

Правило отображения Спецификация значений атрибутов по умолчанию отображается в метаслот, содержащий метаинформацию об атрибуте.

Спецификатор `#REQUIRED` означает, что в XML документе значение данного атрибута должно быть определено. В интерпретации описания такого элемента на языке СИНТЕЗ метаинформация, связанная со слотом, также означает, что значение данного слота при создании экземпляра данного типа должно быть определено. Эти аспекты поведения выражаются в спецификации типа метаслотом, содержащим утверждение `{obligatory}`:

```
metaslot
  xml_Assertion:
    invariant, {{obligatory}}
end;
```

Спецификатор `#IMPLIED` означает что указывать значение атрибута не обязательно, а кроме того, значение атрибута по умолчанию не определено. Вследствие этого нет потребности создавать в спецификации типа на языке СИНТЕЗ метаслот, связанный с атрибутом, для которого указан спецификатор `#IMPLIED`.

Если не используется ни `#REQUIRED` ни `#IMPLIED`, тогда указывается начальное значение атрибута, перед которым может присутствовать спецификатор `#FIXED`, который фиксирует данное значение для атрибута. Для выражения этого в спецификации типа используется метаинформация с слотом `init` и утверждением `{constant}`:

```
metaslot
  init : <value>;
  xml_Assertion :
    invariant, {{constant}};
end
```

4.3 Изменение имен при преобразовании.

При отображении DTD в спецификацию типов СИНТЕЗа возникает проблема именования некоторых структур в спецификации. Предлагается следующее решение этой проблемы:

1. Каждый слот в определении типа СИНТЕЗа, соответствующий атрибуту DTD, имеет имя, полученное из имени этого атрибута с добавлением к нему префикса `xml_Attribute_`, т.е. атрибуту с именем `name` будет соответствовать слот в спецификации типа с именем `xml_Attribute_name`.

2. Каждый абстрактный тип данных, соответствующий классу элементов XML, приобретает имя, полученное из имени данного класса путем добавления к нему префикса `XML_Type_`, например, классу элементов с именем `name` соответствует тип СИНТЕЗа с именем `XML_Type_name`.

3. Каждый слот в определении содержимого именуется по следующему правилу: если типом слота является именованный тип, то имя слота формируется из имени класса элементов, которому соответствует этот тип, с добавлением к нему префикса `xml_Content_`. Если же это не так, то имя слота получается добавлением номера к идентификатору `xml_Content`.

4. При наличии в именах атрибутов или элементов символов, недопустимых в именах типов и атрибутов в языке СИНТЕЗ, вместо них в имени появляется последовательность символов вида _<шестнадцатеричный код символа>. Вместо символа подчеркивания также вставляется последовательность символов. Например, вместо имени элемента `xmlroi:the_type` будет сгенерирован тип с именем `XML_Type_xmlroi_3athe_2dtype`, поскольку `3a` – это шестнадцатеричный код двоеточия, а `2d` – код подчеркивания.

5 Пример отображения DTD в СИНТЕЗ.

В качестве примера, иллюстрирующего некоторые из выше перечисленных аспектов отображения, используются страницы сервера музея Уффици во Флоренции. Эти страницы хранят информацию о залах этого музея: номер зала, название зала, а также список картин, включающий имена авторов и названия картин.

DTD для этого примера был получен с помощью набора программ, предоставляемого World Wide Web Wraper Factory [8], который преобразует HTML документы в XML.

```
<!ELEMENT Room (RoomTitle,PaintingsList)>

<!ELEMENT RoomTitle (RoomName)>
<!ATTLIST RoomTitle RoomNum CDATA #IMPLIED>

<!ELEMENT RoomName (#PCDATA)>
<!ELEMENT PaintingsList (Painting)*>
<!ELEMENT Painting (PainterName,PaintingName)>
<!ELEMENT PainterName (#PCDATA)>
<!ELEMENT PaintingName (#PCDATA)>
```

В данном примере типом документа является тип `Room`. Элементы типа `Room` содержат элементы типов `RoomTitle` и `PaintingsList`. У элементов типа `RoomTitle` есть атрибут `RoomNum`, а содержат они элементы типа `Roomname`. Элемент типа `PaintingsList` содержит ноль или более элементов типа `Painting`, элементы которого в свою очередь содержат пары элементов соответственно типов `PainterName` и `PaintingName`.

Образ, соответствующий данному DTD, содержит спецификации шести абстрактных типов данных СИНТЕЗа, каждый из которых соответствует определению типа элементов в DTD.

```
{XML_Type_Room;
 in: type;
 supertype: XML_ANY;
 content:{
    in : type;
    xml_Content_RoomTitle: XML_Type_RoomTitle;
    xml_Content_PaintingsList: XML_Type_PaintingsList
    }
}
```

```
{XML_Type_RoomTitle;
 in: type;
 supertype: XML_ANY;
 content: XML_Type_RoomName;
 xml_Attribute_RoomNum: string
}

{XML_Type_RoomName;
 in: type;
 supertype: XML_ANY;
 content: string;
}

{XML_Type_PaintingsList;
 in: type;
 supertype: XML_ANY;
 content:{
    sequence;
    type_of_element: XML_Type_Painting
    }
}

{XML_Type_Painting;
 in: type;
 supertype: XML_ANY;
 content:{
    in: type;
    xml_Content_PainterName: XML_Type_PainterName;
    xml_Content_PaintingName: XML_Type_PaintingName;
 };
}

{XML_Type_PainterName;
 in: type;
 supertype: XML_ANY;
 content: string
}

{XML_Type_PaintingName;
 in: type;
 supertype: XML_ANY;
 content: string
}
```

6 Заключение

В настоящей статье определено отображение спецификаций элементов в DTD в спецификацию типов на языке СИНТЕЗ. Приведенная схема отображения реализована средствами языка программирования Java в среде Windows NT. В дальнейшем предполагается создание объектно-ориентированных адаптеров коллекций XML документов. Помимо этого, предполагается отобразить в каноническую модель язык XML Schema [3], который предлагает средства описания структуры и типов данных содержимого XML документов. Язык XML-schema выражен средствами языка XML 1.0 и расширяет возможности XML-DTD. Об актуальности настоящей работы может свидетельствовать находящийся в данный момент в стадии разработки стандарт отображения XML в язык Java [10].

Библиография

[1] *Extensible Markup Language (XML) 1.0* W3C Recommendation 10-February-1998

[2] *Document Object Model (DOM) Level 1* W3C Recommendation 1 October 1998

[3] *XML-schema definition language* W3C Technical Report, 1999 `http://www.w3.org/TR/XMLschema-1`

[4] L.A.Kalinichenko. *SYNTHESIS: the language for desription, design and programming of the heterogeneous interoperable information resource environment.* Institute for Problems of Informatics, Russian Academy of Sciences, Moscow, 1993

[5] Serge Abiteboul, Sopie Cluet, Vassillis Christophides, Tova Milo, Guido Moerkotte, Jerome Simeon. *Querying documents in object datbases* International Journal on Digital Libraries Springer-Verlag 1997.

[6] S. Grumbach, G. Mecca. *In Search of the Lost Schema* In Proceedings of Intern. Conference on Database Theory (ICDT'99), 1999

[7] *The ARANEUS Project* `http://www.dia.uniroma3.it/Araneus/`

[8] *Wold Wide Web Wraper Factory* `http://db.cis.upenn.edu/W4F`

[9] *Virtual Uffizi, The Complete Catalogue* `http://www.arca.net/uffizi/`

[10] *JSR-000031 XML Data Binding Specification* `http://www.javasoft.com/aboutJava/communityprocess/jsr/jsr_031_xmld.html`