Extreme Java
G22.3033-007

Session 3 - Sub-Topic 3
XML Information Modeling

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

- XML-Based Software Development
- Business Engineering Methodology
- XML Physical Entities
- Logical Structure of XML Documents
- XML Document Navigation
- Java APIs
- Custom Markup Languages
- XML Metadata Management
- XML Linking/Pointer Language
- XML Data Binding
- Industry Specific Markup Languages
**XML-Based Software Development**

- Business Engineering Methodology
  - Language + Process + Tools
  - e.g., Rational Unified Process (RUP)

- XML Application Development Infrastructure
  - Metadata Management (e.g., XMI)
  - XML APIs (e.g., JAXP, JAXB)
  - XML Tools (e.g., XML Editors, XML Parsers)

- XML Applications:
  - Application(s) of XML
  - XML-based applications/services
    - MOM & POP
    - Other Services
  - Application Infrastructure Frameworks

**Business Engineering Methodology**

- Business Model/Architecture
  - Use Case View/Model

- Application Model/Architecture
  - Logical and Process View/Models
    - Content, Data, and Process Model (e.g., OIM’s knowledge management, and database/datawarehousing models)

- Application Infrastructure Model/Architecture
  - Implementation View
    - Component Model (e.g., OIM’s component and object model)

- Technology Model/Architecture
  - Deployment View/Model

See Session 4 Sub-Topic 1, and Sub-Topic 2 presentations
See Session 4 Handout on “Business and Application Architecture Engineering”
XML Physical and Logical Structure

- **Physical Structure**
  - Governs the content in a document in form of storage units
  - Storage units are referred to as entities
  - See [http://www.w3.org/TR/REC-xml#sec-physical-struct](http://www.w3.org/TR/REC-xml#sec-physical-struct)

- **Logical Structure**
  - What elements are to be included in a document
  - In what order should elements be included
  - See [http://www.w3.org/TR/REC-xml#sec-logical-struct](http://www.w3.org/TR/REC-xml#sec-logical-struct)

XML Physical Entities

- Allow to assign a name to some content, and use that name to refer to it
- Eight Possible Combinations:
  - Parsed vs. Unparsed
  - General vs. Parameter
  - Internal vs. External
- Five Actual Categories:
  - Internal parsed general
  - Internal parsed parameter
  - External parsed general
  - External parsed parameter
  - External unparsed general
Logical Structure: Namespaces

- See Namespaces 1.0
- Sample Element:
  \[
  <z:a z:b="x" c="y" xmlns:z="http://www.foo.com"/>
  \]
- Corresponding DTD Declaration
  \[
  <!ELEMENT z:a EMPTY>
  <!ATTLIST z:a
    z:b CDATA #IMPLIED
    c CDATA #IMPLIED
    xmlns:z CDATA #FIXED "http://www.foo.com">
  \]

Logical Structure: DTDs

- Shortcomings
  - Separate Syntax
    \[
    <!ELEMENT Para (#PCDATA)>
    <Para>Some paragraph</Para>
    vs.
    <ElementType name="Para">
    <ContentModel><PCData/></ContentModel>
    </ElementType>
  \]
  - Lack of Support for Data-typing
    - DTD Treats an XML Structure as a String of Characters
      \[
      <Price currency="USD">1450</Price>
      <Price currency="USD">too high</Price>
      \]
Logical Structure: XML Schemas

- **Structures**
  - How elements and attributes are setup in an XML document

- **Datatypes**
  - Built-in datatypes (e.g., String, Boolean, numbers)
  - Generated datatypes (e.g., dates, times, real values)
  - Support for user generated datatypes
  - Backward compatibility with functional subset (DTD)
    - ID, IDREF, NMTOKEN, and SGML-based types

- **Grouping of Elements/Attributes**
  - Archetypes and Attribute Groups

- **Inheritance**
  - Via Basetypes, and Archetypes/Attribute Groups

---

<datatype name='AgeInYears'>
  <basetype name='integer'
    URI="http://www.w3.org/xmlschemas/datatypes"/>
  <minInclusive>0</minInclusive>
  <maxInclusive>140</maxInclusive>
</datatype>

<attribute name="employeesAge"
    type="AgeInYears"/>
Logical Structure: Navigation

- URIs/URLs
  - Syntax for encapsulating a name in any registered namespace, and label it with the namespace
  - Produce a member of the universal set of reachable objects
  - See http://www.w3.org/Addressing/
- XPath
  - Used to locate certain parts of an XML document
  - See Session 3 handout on “Processing XML documents in Java using XPath and XSLT”

JAXP and Associated XML APIs

- JAXP: Java API for XML Parsing
  - Common interface to SAX, DOM, and XSLT APIs in Java, regardless of which vendor's implementation is actually being used.
- JAXB: Java Architecture for XML Binding
  - Mechanism for writing out Java objects as XML (marshalling) and for creating Java objects from such structures (unmarshalling).
- JDOM: Java DOM
  - Provides an object tree which is easier to use than a DOM tree, and it can be created from an XML structure without a compilation step.
- JAXM: Java API for XML Messaging
  - Mechanism for exchanging XML messages between applications.
- JAXR: Java API for XML Registries
  - Mechanism for publishing available services in an external registry, and for consulting the registry to find those services.
Simple API for XML (SAX) Parsing APIs

DOM Parsing APIs
XSLT APIs

Transformer Factory

Source Transformer Result

Transformation Instructions

Java API Packages

- **java.xml.parsers**
  - The JAXP APIs, which provide a common interface for different vendors’ SAX and DOM parsers.
  - Two vendor-neutral factory classes: SAXParserFactory and DocumentBuilderFactory that give you a SAXParser and a DocumentBuilder, respectively. The DocumentBuilder, in turn, creates DOM-compliant Document object.

- **org.w3c.dom**
  - Defines the Document class (a DOM), as well as classes for all of the components of a DOM.

- **org.xml.sax**
  - Defines the basic SAX APIs.

- **jaxax.xml.transform**
  - Defines the XSLT APIs that let you transform XML into other forms. 

SAX API Packages

- **org.xml.sax**
  - Defines the SAX interfaces.

- **org.xml.sax.ext**
  - Defines SAX extensions that are used when doing more sophisticated SAX processing, for example, to process a document type definitions (DTD) or to see the detailed syntax for a file.

- **org.xml.sax.helpers**
  - Contains helper classes that make it easier to use SAX -- for example, by defining a default handler that has null-methods for all of the interfaces, so you only need to override the ones you actually want to implement.

- **javax.xml.parsers**
  - Defines the SAXParserFactory class which returns the SAXParser. Also defines exception classes for reporting errors.

DOM API Packages

- **org.w3c.dom**
  - Defines the DOM programming interfaces for XML (and, optionally, HTML) documents, as specified by the W3C.

- **javax.xml.parsers**
  - Defines the DocumentBuilderFactory class and the DocumentBuilder class, which returns an object that implements the W3C Document interface. The factory that is used to create the builder is determined by the javax.xml.parsers system property, which can be set from the command line or overridden when invoking the newInstance method. This package also defines the ParserConfigurationException class for reporting errors.
XSLT API Packages

- **javax.xml.transform**
  - Defines the TransformerFactory and Transformer classes, which you use to get an object capable of doing transformations. After creating a transformer object, you invoke its transform() method, providing it with an input (source) and output (result).

- **javax.xml.transform.dom**
  - Classes to create input (source) and output (result) objects from a DOM.

- **javax.xml.transform.sax**
  - Classes to create input (source) from a SAX parser and output (result) objects from a SAX event handler.

- **javax.xml.transform.stream**
  - Classes to create input (source) and output (result) objects from an I/O stream.

Content of Jar Files

- **jaxp.jar** (interfaces)
  - javax.xml.parsers
  - javax.xml.transform
    - javax.xml.transform.dom
    - javax.xml.transform.sax
    - javax.xml.transform.stream

- **crimson.jar** (interfaces and helper classes)
  - org.xml.sax
    - org.xml.sax.helpers
    - org.xml.sax.ext
    - org.w3c.dom

- **xalan.jar** (contains all of the above implementation classes)
XML Information Modeling

- Steps
  - Documenting the Information Structure
  - Representing the Information Structure in XML Form
  - Defining XML DTDs and/or Schemas

- Modeling Techniques
  - UML: object modeling
  - XML: content modeling
  - ORM: data modeling
  - See Session 3 handout on “XML Information Modeling”

- UML, MOF and XMI
  - See Session 7 handouts on “UML, MOF, and XMI” and “OMG’s XML Metadata Interchange Format (XMI)”

Open Information Model

- Analysis and Design Model
  - Unified Modeling Language (UML) - uml.dtd
  - UML Extensions - umlx.dtd
  - Common Data Types - dtm.dtd
  - Generic Elements - gen.dtd

- Components and Object Model
  - Component Description Model - cde.dtd

- Database and Warehousing Model
  - Database Schema Elements - dbm.dtd
  - Data Transformation Elements - tfm.dtd
  - OLAP Schema Elements - olp.dtd
  - Record Oriented Legacy Databases - rec.dtd

- Knowledge Management Model
  - Semantic Definition Elements - sim.dtd
Custom Markup Languages

- Mathematical Markup Language (MathML)
- OpenMath
- Chemical Markup Language (CML)
- Geography Markup Language (GML)
- Wireless Markup Language (WML)
- Synchronized Multimedia Integration Language (SMIL)
- Synchronized Vector Graphics (SVG)
- Extensible 3D (X3D)
- XML-Based User Interface Language (XUL)
- Extensible Log Format (XLF)

XML Metadata Management

- Issue: UML may not provide enough modeling views and enough expressive power in each view to represent a complete application
- Possible Solutions:
  - Extend UML
    - See OIM’s Analysis and Design Model
  - Use Different Modeling Languages:
    - See Session 3 handout on “XML Information Modeling”
      (uses different models such as UML, XML, and ORM)
  - Use a Meta-Model: MOF and XMI
    - See Session 3 handouts on “UML, MOF, and XMI” and “OMG’s XML Metadata Interchange Format (XMI)”
XML Linking/Pointer Language

- **XLink**
  - Allows elements to be inserted into XML documents in order to create and describe links between resources

- **XML Base**
  - Equivalent of HTML BASE functionality generically in XML documents by defining an XML attribute named xml:base

- **Xpointer**
  - Language used as a fragment identifier for any URI-reference that locates a resource of Internet media type text/xml or application/xml
  - Based on XPath

---

**XLink Example**

```
<my:crossReference
xmlns:my="http://example.com/
xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple"
xlink:href="students.xml"
xlink:role="studentlist"
xlink:title="Student List"
xlink:show="new"
xlink:actuate="onRequest">
Current List of Students
</my:crossReference>
```
**XPointer Example**

- `<DOCTYPE SPEECH [`  
  `<!ELEMENT SPEECH (#PCDATA|SPEAKER|DIRECTION)*>`  
  `<!ATTLIST SPEECH  
  ID ID #IMPLIED>`  
  `<!ELEMENT SPEAKER (#PCDATA)>`  
  `<!ELEMENT DIRECTION (#PCDATA)>]>
  `<SPEECH ID="a27"><SPEAKER>Polonius</SPEAKER>`  
  `<DIRECTION>crossing downstage</DIRECTION>Fare you well, my lord. <DIRECTION>To Ros.</DIRECTION>`  
  `You go to seek Lord Hamlet? There he is.</SPEECH>`

- `id(a27).child(2,DIRECTION)`  
  Selects the second "DIRECTION" element (whose content is "To Ros.")

- `id(a27).child(2,#text)`  
  Selects the second text region,"Fare you well, my lord." (The line break between the SPEAKER and DIRECTION elements is the first text region.)

**XML Base Example**

- `<?xml version="1.0"?>`  
  `<html xmlns="http://www.w3.org/TR/xhtml1/strict" xml:base="http://somewhere.org">`  
  `<head>`  
    `<title>Virtual Library</title>`  
  `/head>`  
  `<body>`  
    `<p>See <a href="new.xml">what's new</a>!</p>`  
    `<p>Check out the hot picks of the day!</p>`  
    `<ol xml:base="/hotpicks">`  
    `<li><a href="pick1.xml">Hot Pick #1</a></li>`  
    `<li><a href="pick2.xml">Hot Pick #2</a></li>`  
    `<li><a href="pick3.xml">Hot Pick #3</a></li>`  
  `/ol>`  
  `/body>`  
`/html>`

- "what's new" resolves to the URI "http://somewhere.org/new.xml"
- "Hot Pick #1" resolves to the URI "http://somewhere.org/hotpicks/pick1.xml"
- "Hot Pick #2" resolves to the URI "http://somewhere.org/hotpicks/pick2.xml"
- "Hot Pick #3" resolves to the URI "http://somewhere.org/hotpicks/pick3.xml"
XML Data Binding

- Aims to automatically generate substantial portions of the Java platform code that processes XML data
- A Sun project, codenamed “Adelard”
- See JSR-31 XML Data Binding Specification
- See [http://java.sun.com/xml/jaxp-1.0.1/docs/binding/DataBinding.html](http://java.sun.com/xml/jaxp-1.0.1/docs/binding/DataBinding.html)

Industry Specific Markup Languages

(see [http://www.oasis-open.org/cover/xml.html#contentsApps](http://www.oasis-open.org/cover/xml.html#contentsApps))

- Ontology Interchange Language (OIL)
- OMG Common Warehouse MetaData Interchange (CWMI)
- OMG Model Driven Architecture (MDA)
- Open Financial Exchange (OFX)
- Straight Through Processing Markup Language (STPML)
- Electronic Commerce Modeling Language (ECML)
- OASIS Business Transactions Technical Committee (OASISBT)
- BizTalk Framework (BizTalk)
- Commerce XML (cXML)
- RosettaNet (RosettaNet)
- Business Process Modeling Language (BPML)