Structured-Document Processing Languages
Spring 2007
Course Review

Goals of the Course

- Learn about central models and languages for
  - manipulating
  - representing
  - transforming and
  - querying
  structured documents (or XML)
- "Generic XML processing technology"

Methodological Goals

- Central professional skills
  - consulting technical specifications
  - experimenting with SW implementations
- Ability to think...
  - to find out relationships
  - to apply knowledge in new situations
- "(Pidgin English" for scientific communication)

XML?

- Extensible Markup Language is not a markup language!
  - does not fix a tag set nor its semantics
  (like markup languages like HTML do)
- XML is
  - A way to use markup to represent information
  - A metallanguage
    - supports definition of specific markup languages through XML DTDs or Schemas
    - E.g. XHTML a reformulation of HTML using XML

XML Encoding of Structure: Example

```
<S>
  S E
  <W> <W> </W> <E A='1'></E> Hello world!
  <W>
  </W>
</S>
```

Basics of XML DTDs

- A Document Type Declaration provides a grammar (document type definition, DTD) for a class of documents
- Syntax (in the prolog of document instance):
  ```
  <!DOCTYPE rootElemType SYSTEM "ex.dtd"
   <!-- "external subset" in file ex.dtd -->
  [ <!-- "internal subset" may come here -->
  ]>
  ```
  - DTD = union of the external and internal subset

How do Declarations Look Like?

```xml
<!ELEMENT invoice (client, item+)>
<!ATTLIST invoice num NMTOKEN #REQUIRED>
<!ELEMENT client (name, email?)>
<!ATTLIST client num NMTOKEN #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT item (#PCDATA)>
<!ATTLIST item
  price NMTOKEN #REQUIRED
  unit (FIM | EUR) "EUR" >
```

Element type declarations

- The general form is
  ```xml
  <!ELEMENT elementType (E)>
  ```
  where E is a content model
- regular expression of element names
- Content model operators:
  - E | F: alternation
  - E: zero or more
  - E+: one or more
  - (E): grouping
XML Schema Definition Language

- XML syntax
  - schema documents easier to manipulate by programs (than the DTD syntax)
- Compatibility with namespaces
  - can validate documents using declarations from multiple sources
- Content datatypes
  - 44 built-in datatypes (including primitive Java datatypes, datatypes of SQL, and XML attribute types)
  - + user-defined datatypes

XML Namespaces

```xml
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns="http://www.w3.org/TR/xhtml1/strict">
 <!-- XHTML is the 'default namespace' -->
 <xsl:template match="doc/title">
   <h1>
     <xsl:apply-templates />
   </h1>
 </xsl:template>
</xsl:stylesheet>
```

3. XML Processor APIs

- How can applications manipulate structured documents?
  - Overview of document parser interfaces
    3.1 SAX: an event-based interface
    3.2 DOM: an object-based interface
    3.3 JAXP: Java API for XML Processing

A SAX-based application

```java
<?xml version='1.0'?>
<A i="1">Hi!</A>
```

DOM: What is it?

- Object-based, language-neutral API for XML and HTML documents
  - Allows programs/scripts to
    » build
    » navigate and
    » modify documents
- "Directly Obtainable in Memory" vs "Serial Access XML"

Overview of XSLT Transformation

```java
<invoice form="00" type="estimated">
  <addressdata>
    <name>John Doe</name>
    <address>
      <streetaddress>Pyynpolku 1</streetaddress>
      <postoffice>70460 KUOPIO</postoffice>
    </address>
  </addressdata>
</invoice>
```

JAXP (Java API for XML Processing)

- An interface for "plugging-in" and using XML processors in Java applications
  - includes packages
    » org.xml.sax: SAX 2.0 interface
    » org.w3c.dom: DOM Level 2 interface
    » javax.xml.parsers: initialization and use of parsers
    » javax.xml.transform: initialization and use of transformers (XSLT processors)
- Included in standard Java
4. Introduction to Style Sheets

- Specify and produce **visual representation** for structured documents
- by defining a mapping from document structure+content to formatting tasks, and
  - inserting/generating new text
  - numbering
  - rearranging
- by rules based on contextual conditions

Process of Transformation (**muunnos**)

- Structured document
- Transformation
- Style sheet
- TeX, FOT
  - \( \text{LaTeX style file, CSS, XSLT} \)
- Formatter input

Process of Formatting (**muotoilu**)

- Creates a detailed description of presentation
- > style sheet may not have complete control of the final formatted presentation!

Process of Rendering (**hahmonnus**)

- Display/play the document on output medium

**CSS - Cascading Style Sheets**

- A stylesheet language
  - mainly to specify the representation of web pages by attaching style (fonts, colours, margins, ...) to HTML/XML documents
- Example style rule:
  ```
  h1 {color: blue; font-weight: bold;}
  ```
CSS Processing Model (simplified)

0. Parse the document
1. Match style rules to elements of the doc tree
   - annotate each element with values assigned for properties
     » inheritance and elaborate “cascade” rules applied to select which value is assigned
2. Generate a formatting structure
   - of nested rectangular boxes
3. Render the formatting structure
   - display, print, audio-synthesize, ...

XSL: Transformation & Formatting

Page regions

- A simple page can contain 1-5 regions, specified by child elements of the simple-page-master

Top-level formatting objects

- Slightly simplified:

XQuery in a Nutshell

- Functional expression language
  - A query is a side-effect-free expression
  - Operates on sequences of items
    - atomic values or XML nodes
  - Strongly-typed: (XML Schema) types may be assigned to expressions statically, and results can be validated
  - Extends XPath 2.0 (but not all axes required)
    - common for XQuery 1.0 and XPath 2.0:
      » Functions and Operators, W3C Rec. 01/2007
  - Roughly: XQuery ≈ XPath 2.0 + XSLT' + SQL'

FLWOR ("flower") Expressions

- for, let, where, order by and return clauses (~SQL select-from-where)
- Form: (ForClause | LetClause)*
  WhereClause? OrderByClause? "return" Expr
- binds variables to values, and uses these bindings to construct a result (an ordered sequence of nodes)

XQuery Example

```xquery
for $pn in distinct-values(
doc("sp.xml")//pno)
let $sp:=doc("sp.xml")//sp_tuple[pno=$pn]
where count($sp) &gt;= 3
order by $pn
return
<well_supplied_item>
<pno>{$pn}</pno>
<avgprice> {avg($sp/price)} </avgprice>
<well_supplied_item>
```

Course Main Message

- XML is a universal way to represent information as tree-like data structures
- Specialized and powerful technologies for processing it
  - Worst hype has settled
  - R&D still active