Web Engineering

Chapter 3: The Web – An Information System

SGML

- Standard Generalized Markup Language
  - developed from GML, IBM 1969 (Goldfarb, Mosher, Lorie)
  - distinction between content and presentation

- dilemma of specific markup language:
  - What is an appropriate set of tags?

- generalized markup
  - documents are described in three parts
    - SGML declaration: definition of tags and their meaning
      - mapping of the abstract SGML Syntax onto concrete characters
      - e.g. '<' or 'TAG' e.g. '>' or charset
    - Document Type Definition, DTD:
      - definition of tags and their meaning
      - the document (content) in the markup defined in the DTD
    - semantic of markup is context dependent

A General SGML Parser

SGML Declaration
Document Type Definition
Document
SGML Processing System (Parser)
Specialized SGML Parser I

- SGML syntax is predefined (included in the parser)
  - E.g. XML-parser (Extensible Markup Language)

Document Type Definition

SGML Declaration

SGML Processing System (Parser)

SGML Concepts

- descriptive instead of procedural markup
  - “this is an X” instead of “do X here”

- entity
  - collection of characters that can be referenced as unit

- element
  - component of the hierarchical structure defined by a document type definition DTD

- in SGML tags are used to the structure of elements

- entities are reference points

Specialized SGML Parser II

- Document Type Definition and SGML syntax is predefined
  - E.g. HTML (Hypertext Markup Language)

Document Type Definition

SGML Declaration

SGML Processing System (Parser)

Document Type Definition - Example

```xml
<!ELEMENT announcement (head, content)>
<!ELEMENT head (title, date?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT content (abstract?, paragraph+)>
<!ELEMENT abstract (#PCDATA|course|name)>
<!ELEMENT paragraph (#PCDATA|course|name)>
```
Document Type Definition (SGML)

- **a DTD defines:**
  - elements of a document class
  - rules how these elements are combined
- **structure:**
  - MDO\ ELEMENT name model MDC
  - <!ELEMENT name model>
- **name**
  - is the name of the element e.g. content or
  - a or conjunction of names of elements e.g. (abstract|course|name)
- **model**
  - specifies the content that is allowed for this element
  - combinations are possible

Model Group I (SGML)

- **(#PCDATA) characterdata**
- **connecting elements:**
  - , sequence in the order specified
  - & all elements must exist (order not specified)
  - | one, or an arbitrary combination must exist
  - example:
    - <!ELEMENT announcement (head, content)> 
    - <!ELEMENT head (name & author & date)> 
    - <!ELEMENT content (image | paragraph | table)> 
- **occurrence of elements:**
  - + at least once, repeatable
  - * optional, may be repeated
  - ? optional, can occur but at most once
  - example:
    - <!ELEMENT document (author, chapter+, appendix*, abstract?)> 

Model Group II (SGML)

- **exceptions**
  - Inclusions
    - the element can be used anywhere in the model group
    - the element must not be used in this model group
  - Exclusion
    - the element must not be used in this model group
  - example:
    - <!ELEMENT document {author, chapter+}+(italic)}
    - <!ELEMENT italic (#PCDATA) –(italic)}
- **minimizing markup OMITTAG**
  - only if tag omission is enabled
  - specify the tags that can be omitted
  - example – start tag may not be omitted, end tag may be omitted:
    - <!ELEMENT author - 0 (#PCDATA) >

DTD Attributes

- **structure - attribute definition list declaration:**
  - MDO\ ATTLIST name attribute-definition MDC
  - <!ATTLIST name attribute-definition >
- **structure - Attribute definition:**
  - attribute-name declared-value default-value 
  - declared-value – alpha numeric, 
    (like variables or reserved words) 
  - default-value - alpha numeric or reserved word 
    - #REQUIRED – must be provided 
    - #IMPLIED – if not available the program interpreting the 
      document may imply a value 
    - #FIXED – fixed attribute 
- **example**
  - <!ATTLIST image> 
    name CDATA #REQUIRED 
    type (bw|color) bw 
    date CDATA #IMPLIED>
**Document Processing / Layout I**

- **HTML**
  - layout can be generated automatically
  - restricted to a specific document type with defined tags

- **SGML**
  - possible to define arbitrary document type
  - layout based on Document Style Specification and Semantics Language, DSSSL

**Diagram:**
- SGML document (incl. DTD)
- DSSSL processor
- SPDL = Standard Page Description Language
- SPDL style sheet

**Fully-Tagged SGML**

- Fully-tagged SGML document
  - parsing without DTD possible
  - all information required is included in the tags
  - requirement: type validation when creating the document

- Extensible Markup Language (XML)
  - restricted sub set of SGML
  - to make parsing simpler
  - only fully-tagged documents are valid

**Document Processing / Layout II**

- **DSSSL**
  - Document Style Specification and Semantics Language
    - transformation of SGML documents of one type into another document type
    - transformation in a output format
      - SPDL: Standard Page Description Language (similar to Postscript)

- transforming SGML documents
  - DSSSL creates a tree structure of the SGML document based on the DTD
  - Standard Tree Formatting Process, STFP, produces an output according to the style sheet

**Extensible Markup Language (XML)**

**Design Goals** (http://www.w3.org/TR/REC-xml):

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.
XML

- general way to describe structured information
- XML-name space
  - XML-name space
    - `http://www.w3.org/TR/REC-xml-names/`
- „well-formed-documents“
  - End-Tags are required, or `</..>`
  - minimizing tagging is not allowed
  - case sensitive
- user / developer defined DTDs
- family of languages
  - XLink
  - XPointer
  - XPath
  - XSL
  - ...

XML Tree Structure

- Any XML-Document has a hierarchical structure. Each document can be visualized in tree structure.

DTD - Example

- example DTD:
  ```xml
  <?xml version="1.0"?>
  <!DOCTYPE nameregister [
    <!ELEMENT nameregister (entry*) >
    <!ELEMENT entry (name, phone?, email?) >
    <!ELEMENT name (#PCDATA) >
    <!ELEMENT phone (#PCDATA) >
    <!ELEMENT email (#PCDATA) >
  ]>
  ```

XML Example

- name-register
  ```xml
  <name-register>
    <entry><name>Albrecht Schmidt</name>
    <phone>690229</phone></entry>
    <entry><name>Bill Clinton</name>
    <email>bill@white_house.gov</email></entry>
    <entry><name>James Bond</name>
    <phone>+44007</phone>
    <email>007@mi5.gov.uk</email></entry>
  </name-register>
  ```
XML Presentation

- XML Browser
  - XML document type
  - XSL style sheet
  - Presentation
  - XML document

Extensible Stylesheet Language (XSL)

- XML does not describe presentation!
- description of a transformation necessary
- Extensible Style Language, XSL
  - XML DTD for formatting, design and layout
  - subset of DSSSL-Online

- “XSL is a language for expressing stylesheets. Given a class of structured documents or data files in XML, designers use an XSL stylesheet to express their intentions about how that structured content should be presented; that is, how the source content should be styled, laid out and paginated onto some presentation medium such as a window in a Web browser or a set of physical pages in a book, report, pamphlet, or memo.”
  (http://www.w3.org/TR/WD-xsl/)

XML Name Space - Example

Example 1:
```xml
<?xml version="1.0"?>
<!-- all elements here are explicitly in the HTML namespace -->
<html xmlns='http://www.w3.org/TR/REC-html40'>
  <head><title>Frobnostication</title></head>
  <body>
    <p>Moved to <a href='http://frob.com'>here.</a></p>
  </body>
</html>
```

Example 2:
```xml
<?xml version="1.0"?>
<!-- initially, the default namespace is "books" -->
<book xmlns='urn:loc.gov:books'
  <title>Cheaper by the Dozen</title>
  <isbn:number>1568491379</isbn:number>
  <notes>
    <!-- make HTML the default namespace for some commentary -->
    <p xmlns='urn:w3-org-ns:HTML'>
      This is a <i>funny</i> book!
    </p>
  </notes>
</book>
```

XSL Process

- input:
  - XML document with DTD
  - XSL style sheet

- “tree transformation”
  - creating a tree from the given XML-document
  - using rules from the style sheet
  - template matching
  - XSL Transformations (XSLT) Specification
    (http://www.w3.org/TR/1999/WD-xslt-19990421)

- “formatting”
  - all elements from the tree a formatted according to the style sheet, similar to Cascading Style Sheets (CSS)
  - vocabulary xsl:fo
XSL Transformations

- This specification defines the syntax and semantics of XSLT, which is a language for transforming XML documents into other XML documents.
- XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary.
- A transformation expressed in XSLT describes rules for transforming a source tree into a result tree.
- The transformation is achieved by associating patterns with templates. A pattern is matched against elements in the source tree. A template is instantiated to create part of the result tree.
- [http://www.w3.org/TR/xslt](http://www.w3.org/TR/xslt)

XML Path - Location

- A location step has three parts:
  - an axis, which specifies the tree relationship between the nodes selected by the location step and the context node,
  - a node test, which specifies the node type and expanded-name of the nodes selected by the location step (* = wildcard), and
  - zero or more predicates, which use arbitrary expressions to further refine the set of nodes selected by the location step.
- The syntax for a location step is the axis name and node test separated by a double colon, followed by zero or more expressions each in square brackets.
- For example, in `child::para[position()=1]`,
  - `child` is the name of the axis,
  - `para` is the node test
  - `[position()=1]` is a predicate.

XML Path Language I

- [http://www.w3.org/TR/xpath](http://www.w3.org/TR/xpath)
- "XPath is the result of an effort to provide a common syntax and semantics for functionality shared between XSL Transformations [XSLT] and XPointer [XPointer]. The primary purpose of XPath is to address parts of an XML [XML] document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document."

XML Path – Axes

```
AxisName ::=   
    'ancestor' | 'ancestor-or-self' | 'attribute'
    | 'child'    | 'descendant' | 'descendant-or-self'
    | 'following' | 'following-sibling'
    | 'namespace'
    | 'parent'
    | 'preceding'
    | 'preceding-sibling' | 'self'
```

- [http://www.w3.org/TR/xslt](http://www.w3.org/TR/xslt)
XML Path – Axes (selection)

- the child axis contains the children of the context node
- the descendant axis contains the descendants of the context node; a descendant is a child or a child of a child and so on
- the parent axis contains the parent of the context node, if there is one
- the ancestor axis contains the ancestors of the context node; the ancestors of the context node consist of the parent of context node and the parent's parent and so on; thus, the ancestor axis will always include the root node, unless the context node is the root node
- the following axis contains all nodes in the same document as the context node that are after the context node in document order, excluding any descendants and excluding attribute nodes and namespace nodes
- the preceding axis contains all nodes in the same document as the context node that are before the context node in document order, excluding any ancestors and excluding attribute nodes and namespace nodes
- the attribute axis contains the attributes of the context node; the axis will be empty unless the context node is an element
- the namespace axis contains the namespace nodes of the context node; the axis will be empty unless the context node is an element
- the self axis contains just the context node itself

XML Path by Example I

- The child axis contains the children of the context node. The child axis is the default axis and it can be omitted.
  - `/AAA` Equivalent of `child::AAA`
  - `/AAA/BBB` Equivalent of `child::AAA/child::BBB`
- The basic XPath syntax is similar to file system addressing. If the path starts with the slash `/`, then it represents an absolute path to the required element.
  - `/AAA` Selects the root element AAA
  - `/AAA/BBB/CCC` Selects all elements BBB which are children of AAA
  - `/AAA/BBB` Selects all elements BBB which are children of the root element AAA
- If the path starts with `@` then all elements in the document which fulfill following criteria are selected.
  - `//@AAA` Selects all attributes of AAA
  - `//@AAA/@BBB` Selects all attributes BBB which are children of AAA
- Expressions in square brackets can further specify an element. A number in the brackets gives the position of the element in the selected set. The function last() selects the last element in the selected set.
  - `/AAA/BBB[1]` Selects the first BBB child of element AAA
  - `/AAA/BBB[last()]` Selects the last BBB child of element AAA

XML Path by Example II

- Attributes are specified by `@` prefix.
  - `/BBB[@id]` Selects BBB elements which have attribute id
  - `/BBB[@*]` Selects BBB elements which have any attribute
  - `/BBB[not(@*]]` Selects BBB elements without attribute
- Values of attributes can be used as selection criteria. Function normalize-space removes leading and starting spaces and replaces sequences of whitespace characters by a single space.
  - `/BBB[@id='b1']` Selects BBB elements which have attribute id with value b1
  - `Function count()` counts the number of selected elements.
  - `//*[count(BBB)=2]` Selects elements which have two children
  - `Function name()` returns name of the element, the starts-with function returns true if the first argument string starts with the second argument string, and the contains function returns true if the first argument string contains the second argument string.
  - `//*[name()='BBB']` Selects all elements with name BBB
  - `//*[starts-with(name(), 'B')` Selects all elements in which starts with letter B
  - `//*[contains(name(), 'c')]` Selects all elements whose name contains letter C

Source: [http://www.zvon.org/xxl/XPathTutorial/General/examples.html](http://www.zvon.org/xxl/XPathTutorial/General/examples.html)

XML Path by Example III

- Several paths can be combined with `|` separator.
  - `//CCC | //BBB` Selects all elements CCC and BBB
  - `//AAA/EEE | //BBB` Selects all elements BBB and elements EEE which have children of root element AAA
  - `//AAA/EEE | //DDD | //AAA` Number of combinations is not restricted
- The descendant axis contains the descendants of the context node; a descendant is a child or a child of a child and so on; thus the descendant axis never contains attribute or namespace nodes.
  - `/AAA/BBB//descendant::*` Selects all descendants of AAA
  - `//CCC//descendant::*` Selects all elements which have CCC among its ancestors
  - `//CCC//descendant::*` Selects all elements which have CCC among its ancestors
- The parent axis contains the parent of the context node, if there is one.
  - `//DDD//parent::*` Selects all parents of DDD element
- The ancestor axis contains the ancestors of the context node; the ancestors of the context node consist of the parent of context node and the parent's parent and so on; thus, the ancestor axis will always include the root node, unless the context node is the root node.
  - `/AAA/BBB/DDD//ancestor::*` Selects all elements given in this absolute path
  - `//FFF//ancestor::*` Selects all ancestors of FFF element

Source: [http://www.zvon.org/xxl/XPathTutorial/General/examples.html](http://www.zvon.org/xxl/XPathTutorial/General/examples.html)
XPointer Language I

- Identification of any fragment of a XML document
- No need to change the target document
- In HTML
  - Defining the target: e.g. `<a name="C1">`
  - Fragment identifier in URI after #, e.g. `/doc.html#C1`
- Design goals ([http://www.w3.org/TR/WD-xptr](http://www.w3.org/TR/WD-xptr)):
  - XPointers shall address into XML documents
  - XPointers shall be straightforwardly usable over the Internet.
  - XPointers shall be straightforwardly usable in URIs.
  - The XPointer design shall be formal and concise.
  - The XPointer syntax shall be reasonably compact and human readable.
  - XPointers shall be optimized for usability.
  - XPointers must be feasible to implement.

XPointer Language II

- XPointer is built on top of the XML Path Language [XPath], which is an expression language underlying the XSL Transformations (XSLT) language. XPointer's extensions to XPath allow it to:
  - Address points and ranges as well as whole nodes
  - Locate information by string matching
  - Use addressing expressions in URI references as fragment identifiers (after URI-escaping)
- XPointer supports addressing into the internal structures of XML documents. It allows for examination of a document's hierarchical structure and choice of its internal parts based on various properties, such as element types, attribute values, character content, and relative position. In particular, it provides for specific reference to elements, character strings, and other parts of XML documents, whether or not they bear an explicit ID attribute.

XPointer Language III

- Definition: sub-resource
  The portion of an XML resource that is identified by an XPointer. For example, the whole resource being referred to is an XML document, but a sub-resource might be a particular element inside the document. Following a link to such a sub-resource might result, for example, in highlighting that element or scrolling to that point in the document.
- Definition: location-set
  An ordered list of document nodes, points, and/or ranges, such as produced by an XPointer expression. This corresponds to the node-set that is produced by XPath expressions, except for the generalization to include points and ranges.
- Definition: point
  A position in XML information. This notion comes from the DOM Level 2 specification's notion of positions; this specification refers to DOM positions by the term "point" to avoid confusion with XPath positions.
- Definition: range
  An identification of a contiguous selection of all the XML information between a pair of endpoints. This notion comes from the DOM Level 2 specification.
- Definition: singleton
  A location that consists of a single, contiguous portion of a document. Some XPointers can locate multiple data portions, such as three contiguous `aaa` element nodes in a list or six noncontiguous occurrences of a string in a document. When an XPointer instead locates only a single contiguous data portion such as a range, string range, or single node, that location is said to be a singleton.

XPointer Example

**XPointer:** `xpointer(range(//AAA/BBB[2]))`

```
<AAA>
  <BBB bbb="111">
    Text in the first element BBB.
  </BBB>
  <BBB bbb="222">
    Text in another element BBB.
    <DDD ddd="999">
      Text in more nested element.
    </DDD>
  </BBB>
  <BBB bbb="333">
    Text in the third element BBB.
  </BBB>X
<![CDATA[
  <CCC ccc="123" xxx="321">
    Again some text in some element.
  </CCC>
]]>
</AAA>
```
XML Linking Language I

- Links are not described in document structure
- Linking, XML-Style
  - XLink is a small markup language that allows XML resources to contain links themselves. It can be used to create hyperlinks, but also has new features that make it easier to manage links over time.
  - XPointer is an adjunct to URIs that allows XML resources to be linked into. It can be used to get to any arbitrary region inside an XML file

XML Linking Language II

- Design Goals
  - It is a requirement to allow for "open systems" of linking where not all resources are under the control of a single person or organization (along with easier "closed systems"). For example, broken links must be tolerated.
  - Both unidirectional links (common on the Web today) and multidirectional links (commonly used in commercial hypermedia systems) must be supported.
  - The XLink Expression Language Shall Be XML
  - XLink Shall Be Human-Readable
  - XLink May Reside Outside the Documents in Which the Participating Resources Reside
  - XPointers Shall Represent the Abstract Structure and Significance of Links
  - XPointers Shall Address into XML Documents
  - XPointers Shall Be Straightforwardly Usable in URIs
  - The XPointer Syntax Shall Be Reasonably Compact and Human Readable

XML Linking Language III

- The XML Linking Language (XLink) allows elements to be inserted into XML documents in order to create and describe links between resources.
- XLink provides a framework for creating both basic unidirectional links and more complex linking structures. It allows XML documents to:
  - Assert linking relationships among more than two resources
  - Associate metadata with a link
  - Express links that reside in a location separate from the linked resources
- Definitions:
  - An XLink link is an explicit relationship between resources or portions of resources.
  - Using or following a link for any purpose is called traversal.
  - The source from which traversal is begun is the starting resource and the destination is the ending resource.
  - Information about how to traverse a pair of resources, including the direction of traversal and possibly application behavior information as well, is called an arc.
  - An arc that has a local starting resource and a remote ending resource goes outbound.
  - If an arc’s ending resource is local but its starting resource is remote, the arc goes inbound.
  - If neither the starting resource nor the ending resource is local, then the arc is a third-party arc.
- Documents containing collections of inbound and third-party links are called link databases, or linkbases

XML Link - Examples

- Example 1 - XML:
  
  ```xml
  <xlink:simple href="students.xml" role="student list" title="Student List" show= "new" actuate="user">The list of students.</xlink:simple>
  ```

- Example 1 - HTML:
  
  ```html
  <A href="students.html">The list of students.</A>
  ```

- Example 2 - XML:
  
  ```xml
  <example:2-XML:
  <xlink:extended
    xmlns:xlink=http://www.w3.org/XML/XLink/0.9
    xlink:locator href="#Fred" id="student" />
  <xlink:locator href="teachers.xml#Joe" id="teacher"/>
  <xlink:arc from="#student" to="teacher"
            show="replace"/>
  </xlink:extended>
  ```
### XLink

- **inline links**
  - Is part of the resource

- **out-of-line links**
  - Specified outside of the resource
  - Reference from resources to link elements
    - unidirectional
    - bi-directional
    - multi-directional

### XLink Attributes (selection)

- **Locator**
  - `href`
    - Elements that address the remote resources participating in the link

- **linking attributes**
  - `arc`
    - Elements that provide traversal rules among the link's participating resources, e.g. from to

- **Behaviour attribute**
  - `show`
    - Options: new, replace, embed, other, none
  - `actuate`
    - Options: onLoad, onRequest, other, none

- **Semantic attribute**
  - `title`
    - The title attribute is used to describe the meaning of a link or resource in a human-readable fashion


### XML Document Object Model

- **http://www.w3.org/DOM/**

- "The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page”

### XML vs. SGML & HTML

- **XML vs. SGML**
  - XML is a subset of SGML
  - Companion specifications XLink, XPath, XSL
  - To parse without DTD
  - Less complexity

- **XML vs. HTML**
  - Structured information
  - Extensions
    - E.g. semantic markup
  - Processing instructions
XML Applications – MathML I

- MathML - http://www.w3.org/Math/
- example: (presentational tags)
  \[ x^2 + 4x + 4 \leq 0 \]
  ```xml
  <mrow>
    <msup> <mi>x</mi> <mn>2</mn> </msup>
    + <mrow> <mn>4</mn> <mo>&invisibletimes;</mo> <mi>x</mi> </mrow>
    + <mn>4</mn>
  </mrow>
  = <mn>0</mn>
  ```

XML Applications – MathML II

- MathML - http://www.w3.org/Math/
- example: (semantic tags)
  \[ x^2 + 4x + 4 \]
  ```xml
  <apply>
    <plus/>
    <apply>
      <power/>
      <ci>x</ci>
      <cn>2</cn>
    </apply>
    <apply>
      <times/>
      <cn>4</cn>
      <ci>x</ci>
    </apply>
    <cn>4</cn>
  </apply>
  ```

XML/HTTP Application – SOAP Simple Object Access Protocol

- RPC mechanism based on
  - XML documents to describe calls
  - HTTP as basic transport protocol
- Example:
  ```xml
  POST /StockQuote HTTP/1.1
  Host: www.stockquoteserver.com
  Content-Type: text/xml-SOAP
  Content-Length: nnnn
  Message-Type: Call
  <GetLastTradePrice>
    <ticker>DIS</ticker>
  </GetLastTradePrice>

  HTTP/1.1 200 OK
  Connection: close
  Content-Type: text/xml-SOAP
  Content-Length: nnnn
  Message-Type: CallResponse
  <GetLastTradePriceResponse>
    <__return>34.5</__return>
  </GetLastTradePriceResponse>
  ```

XML Applications

- SMIL
  http://www.w3.org/AudioVideo/
- SVG
  http://www.w3.org/Graphics/SVG/
- RDF
  http://www.w3.org/RDF/
- ...
Selected URLs

- http://www.w3.org/XML/
- http://www.w3.org/XML/Linking.html
- http://www.w3.org/TR/xlink/
- http://www.w3.org/TR/xptr
- http://www.w3.org/TR/xpath
- http://www.oasis-open.org/cover/schemas.html
- http://msdn.microsoft.com/library/periodic/period00/xml0500.htm