Standards for digital encoding

Tomaž Erjavec

Institut für Informationsverarbeitung
Geisteswissenschaftliche Fakultät
Karl-Franzens-Universität Graz
Lecture 1: XML
3.11.2006

Overview

1. a few words about me
2. a few words about you
3. a short introduction to standards
4. and some words on XML
Lab session:
writing a (small) valid XML document

Lecturer

• Tomaž Erjavec
  Department of Knowledge Technologies
  Jožef Stefan Institute
  Ljubljana
• http://ni.ijs.si/et/
• tomaz.erjavec@ijs.si
• corpora and other language resources, standards, annotation, text-critical editions
• Web page for this course:
  http://ni.ijs.si/et/teach/graz06/standards/
Students

- background: field of study, exposure to XML/TEI?
- emails?
- expectations?

Overview of the course

1. XML
2. Introducing TEI
3. TEI for corpora
4. TEI for text critical editions
5. ...

Standards

- dictionary: an obligatory uniform regulation for measurement, quantity or quality // that which specifies how something can or must be
- consensually accepted regulations, which are public and contain explicit definitions
- the main purpose is to harmonise industrial practice in various fields in order to enable interchange
Standardisation bodies

publish standards according to strictly defined procedures:

• national standards: ANSI, DIN, ÖN, SIST
• international standards: IEC, ISO
• ISO: International Organization for Standardization, Geneva (1947)
• ISO structure: Technical Committees are composed of members from participating countries, who then develop and approve standards from their field
• ISO TCs can be further composed of sub-committees (SC) and these can containing Working Groups (WG)

ISO TC 37

• Technical Committee on Terminology
• important for all other standards, as each standard must contain a section on terminology
• basic definitions, .... ISO 639, MARTIF
• in 2001 name of TC 37 changed to: .... and other language and content resources
• ISO TC 34 SC4: Language Resources Management

W3C

• The World Wide Web Consortium
• first recommendation was HTML (1992)
• best known versions of HTML: 3.2, 4.1
• XML 1.0 released February 1998
• Many XML related standards:
  – DOM Level 1 V1.0 (October 1998)
  – XML Namespaces V1.0 (January 1999)
  – XPath V1.0 (November 1999)
  – XSLT V1.0 (November 1999)
  – XSL, V1.0 (January 2001)
  – XML Schema V1.0 (May 2001)
  – XLink 1.0 (June 2000)
  – XML Information Set V1.0 (December 2001)
  – XSLT V1.0 (October 2001)
  – XML Information Set V1.0 (October 2001)
  – XPath 2.0 WD (April 2002)
Why standards for encoding of digital data?
The encoding of digital data is (was) typically bound to a particular piece of software e.g. a text editor.
Problems:
• longevity: rapid advances in technology make programs obsolete very soon, and the data bound to these programs becomes unreadable
• interchange: difficult to use data on other platforms
• exploitation: difficult to re-use the data for other purposes
• intelligibility: the data are understandable only to the program (no public and stable specifications of the format)
• validation: we don’t know whether certain data is written according to the format specification or not

How to encode language data?
• format of text editors: very loose encoding, too oriented to the visual appearance of text
• databases: too rigid encoding, does not allow for mixture of content (text) and structure (markup)
• ISO 8879 SGML (Standard Generalised Markup Language), 1986
• defined a language for the representation of texts that will be processed by computer programs

SGML
defined an encoding which is:
• very general, as it is a “meta-language” (a language for describing other languages) and lets you design your own customised markup languages for different types of documents
• interchangeable between computer platforms
• resistant to changes in technology
• enables the use of documents for various purposes
• enables automatic validation whether a certain document is compliant with the standard
Problems with SGML

- the standard is very complex
- software for using it was either very expensive or very “academic”
- the conversion of existing documents into SGML was expensive
- so, the use of SGML was limited to large companies or academia

The Web

- best known application of SGML: HTML
- but SGML compliant HTML is used by very few web pages
- HTML is also not expressive enough for the encoding of arbitrary (web) data
- the need for a new standard for encoding web data that would have all the advantages of SGML without its weaknesses
- ➔ eXtended Markup Language, XML

XML now

- XML became very popular, and is becoming the universal medium for interchange of (language) data
- many supporting standards
- many freely available tools for processing XML
- many programs implement importing and exporting data in XML
What is XML?

• XML is a definition of device-independent, system-independent methods of storing and processing texts in electronic form
• XML is a project of W3C; hence, it is an open and non-proprietary specification
• XML is a subset of SGML
• XML is a “metalanguage” -- a language for describing other languages -- which lets you design your own customised markup languages for different types of documents

What is a Markup Language?

• markup (equivalently, encoding)
  -- making explicit an interpretation of text
• markup language
  -- a set of markup conventions used together for encoding texts.
• A markup language must specify:
  -- how markup is to be distinguished from text,
  -- what the markup means,
  -- what markup is allowed,
  -- what markup is required

Structure of XML documents

```
<poem>
  <title>The SICK ROSE</title>
  <stanza>
    <line>O Rose thou art sick.</line>
    <line>The invisible worm,</line>
    <line>That flies in the night</line>
    <line>In the howling storm:</line>
  </stanza>
  <stanza>
    <line>Has found out thy bed</line>
    <line>Of crimson joy:</line>
    <line>And his dark secret love</line>
    <line>Does thy life destroy.</line>
  </stanza>
</poem>
```

- document = text + mark-up
- element = start tag + content + end tag
- generic identifier = name of the tag
- element contains text or elements or both (or nothing)
The SICK ROSE

O Rose thou art sick.
The invisible worm,
That flies in the night
In the howling storm:

Has found out thy bed
Of crimson joy:
And his dark secret love
Does thy life destroy.

Serialization

XML data model

Empty elements

- elements with content:
  <tag> ... </tag>
- empty elements have no content:
  <tag/>
- used for indicating “points” in the document, for example page breaks
- formally
  <tag/> = <tag></tag>

Attributes

used to describe properties of elements
Example: <table id="P1" status='revised'> ... </table>

- given as attribute-value pairs inside the start-tag
- value must be inside matching quotation marks, single or double;
- order in which attribute-value pairs are supplied inside a tag has no significance;
- an XML processor can use the values of the attributes in any way it chooses; the id attribute is a slightly special case in that, by convention, it is always used to supply a unique value to identify a particular element occurrence, which may be used for cross reference purposes.
Comments

- Comments can appear anywhere in text (but not in markup)
- Comments start with `<!--` and end with `-->`
- Comments cannot be nested and cannot contain `--`
  
  *e.g.*
  
  `<poem>`
  `<title>The SLICK is this an typo? Rose</title>`
  `<stanza>`
  `<line>O Rose thou art sick.</line>`
  `<line>`
  `<!-- some lines missing -->`
  `<line>`
  `<!-- here comes the second stanza -->`
  `<line>`
  `</stanza>`
  `</poem>`
- Note that in XML 'meta-markup' starts with `<!` or `?`

Example: annotated corpus

```xml
<s id="Osl.1.2.2.1">
  <w lemma="biti" ana="Vcps-smal">Bi</w>
  <w lemma="biti" ana="Afps-smn">jasen</w>
  <w lemma="mrzel" ana="Afps-smn">mrzel</w>
  <w lemma="apriški" ana="Afps-smn">apriški</w>
  <w lemma="dan" ana="Ncnsn">dan</w>
  <w lemma="in" ana="Ccs">in</w>
  <w lemma="tura" ana="Nefpn">tura</w>
  <w lemma="biti" ana="Vczp3p">so</w>
  <w lemma="biti" ana="Vnsp-pla">bile</w>
  <w lemma="trinajst" ana="Mnqpl">trinajst</w>
</s>
```

Example: dictionary

```xml
<entry id="dscd.4345"/>
```

Example: annotated corpus

Example: dictionary
Entities

- XML documents can also contain entity references, which are, when processing the document, substituted by their interpretation (the entity).
- An entity reference starts with the character ampersand and ends with the semicolon: &...
- A few entities are predefined in XML:
  &lt; = <
  &gt; = >
  &amp; = 
  ’ = ‘
  " = "
- < and & are "magic" characters and must always be escaped when using them in the text:
- 1 < 2 must be written as 1 &lt; 2
- Procter & Gamble must be written as Procter &amp; Gamble
- Entities are also used for other purposes

XML declaration

Every XML document must begin with an XML declaration which does two things:

- Specifies that this is an XML document, and which version of the XML standard it follows
- Specifies which character encoding the document uses:
  - <?xml version="1.0" ?>
  - <?xml version="1.0" encoding="iso-8859-1" ?>
- The default, and recommended, encoding is UTF-8

Minimal requirements

- The document starts with the XML declaration
- Tags and entity references are written correctly
  Wrong: &lt;p 1 &lt; 2&lt;/p
- The document must be a tree:
  - Every start tag has a matching end-tag
    (vname ≠ <Name> ≠ <NAME> )
  - Elements are correctly nested
    Wrong: &lt;a&gt;...&lt;b&gt;...&lt;/a&gt;...&lt;/b&gt;
  - The document has a single top-level element

→ A well-formed XML document
Splot the mistake

```
<greeting>Hello world!</greeting>
<greeting>Hello world!</Greeting>
<greeting><grunt>Ho</grunt> world!</greeting>
<grunt>Ho <greeting>world!</greeting></grunt>
<grunt type=loud>Ho</grunt>
<grunt type="loud"><greeting>
<grunt type = "loud"/>
```

Another bad XML document

```
<HTML>
<HEAD><TITLE>Links</TITLE></HEAD>
<BODY>
<H1 align=center>Interesting<br>WWW links</H1>
<UL>
  <LI><A HREF="http://www.w3.org/XML">W3C XML</A>
  <LI><A HREF="http://xml.coverpages.org">Cover's pages</A>
</UL>
<Form action="http://www.google.com/search" method=get>
  <A href="http://www.google.com">Google</a>
  <input type=text name=q size=28 maxlength=256>
  <input type=hidden name=meta value="lr=&hl=en">
</Form>
</BODY>
</HTML>
```

Defining the rules

- A valid XML document conforms to rules which are stated in an external schema ("element grammar") of some sort.
- A schema specifies:
  - names for all elements used
  - names and datatypes and (occasionally) default values for their attributes
  - rules about how elements can nest
  - and a few other things, depending on the schema
  - language
- n.b. A schema does not specify anything about what elements "mean"
In XML a schema is optional!

- XML allows you to make up your own tags, and doesn’t require a schema...
- The XML concept is dangerously powerful:
  - XML elements are light in semantics
  - one man’s <p> is another’s <para> (or is it?)
  - the appearance of interchangeability may be worse than its absence
- But XML is too good to ignore
  - mainstream software development
  - proliferation of tools
  - the language of the web

What can a schema (or DTD) do for you?

- ensure that your documents use only predefined elements, attributes, and entities
- enforce structural rules such as ‘every chapter must begin with a heading’ or ‘recipes must include an ingredient list’
- make sure that the same thing is always called by the same name
- schema languages vary in the amount of validation they support

Schema languages

- Schemas can be written in:
  - XML DTD Language
    (inherited from SGML)
  - The W3C schema language
    (main successor of DTDs)
  - The ISO Relax NG schema language
    (mostly used by latest version of TEI)
A simple DTD

XML document:
```xml
<city>
  <name>Graz</name>
  <inhabitants>285,470</inhabitants>
  <country>Austria</country>
</city>
```

DTD:
```xml
<!ELEMENT city (name, inhabitants, country)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT inhabitants (#PCDATA)>
<!ELEMENT country (#PCDATA)>
```

A more complex DTD

```xml
<!ELEMENT anthology (poem+)>
<!ELEMENT poem (title?, stanza+)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT stanza (line+)>
<!ELEMENT line (#PCDATA)>
```

```
<anthology>
  <poem>
    <title>The SICK ROSE</title>
    <stanza>
      <line>O Rose thou art sick.</line>
      <line>The invisible worm,</line>
      <line>In the howling storm:</line>
    </stanza>
    <stanza>
      <line>Has found out thy bed</line>
      <line>Of crimson joy:</line>
      <line>Does thy life destroy.</line>
    </stanza>
  </poem>
</anthology>
```

An element definition gives:
- the name of the element
- its content model

Content Model Operators

- { open bracket for grouping
- } close bracket
- ? follows
- | or
- * maybe
- + repeated 0 or more times
- + repeated once or more times

```xml
<!ELEMENT poem (title?,
  (line* |
    (refrain?, (stanza, refrain*)+)
  )>
```
Mixed content

If an element contains #PCDATA and element content, #PCDATA must always appear as the first option in an alternation; the group containing it must use the star operator; it may appear once only, and in the outermost model group.

```xml
<!ELEMENT item1 (#PCDATA | para)*> <!-- OK -->
<!ELEMENT item2 (#PCDATA | para | note)*> <!-- OK -->
<!ELEMENT item3 (#PCDATA , para)*> <!-- WRONG! -->
<!ELEMENT item4 (para | #PCDATA)*> <!-- WRONG! -->
<!ELEMENT item5 (#PCDATA | para)+> <!-- WRONG! -->
<!ELEMENT item6 (para | #PCDATA | note)*> <!-- WRONG! -->
```

Content model ambiguity

XML parsing is deterministic so content model must not be ambiguous.

```xml
<!ELEMENT x (a, (b | c))> <!-- OK -->
<!ELEMENT x ((a, b)|(a, c))> <!-- WRONG! -->
```

Empty Content

Empty elements do not have content. To distinguish them from those with content in well-formed XML documents, they have a special form: the tag ends with a slash.

- In the DTD:
  ```xml
  <!ELEMENT pageBreak EMPTY>
  ```
- In the document:
  ```xml
  ... <p> The page ends here. <pageBreak/> Here starts a new one. </p> ...
  ```
Attributes

- In the DTD:
  ```xml
  <!ATTLIST table
type CDATA #IMPLIED allowed
id ID #REQUIRED necessary
status (draft | revised | final) "draft" default value
  >
  ```

- In the XML document:
  ```xml
  <table id="tab.12" type="summary" status="revised">
  </table>
  ```

Entities

- In the DTD:
  ```xml
  <!ENTITY xml-url "http://www.w3.org/XML">
  <!ENTITY xml-ref "<A href='&xml-url;'></A>">
  ```

- In the document:
  ```xml
  <hint>Read about XML at &xml-ref;.</hint>
  ```

- After processing:
  ```xml
  <hint>Read about XML at <A href='http://www.w3.org/XML/'>http://www.w3.org/XML/</A>.</hint>
  ```

Character references

- Character references are used for cases where certain characters cannot be represented (entered, stored, transmitted, displayed) directly.
- Character reference starts with
  - `&#` followed by the decimal number of the character, e.g.: Saarbrücken
  - `&` followed by the hexadecimal number of the character, e.g.: Saarbrücken
- When processing, such references are substituted by their codepoint
- Codepoints can be found on the Unicode Web pages
External Entities

• External entity references are substituted by the contents of files:
  <!ENTITY Chap1 SYSTEM "P4X/p4chap2.xml">
  <!ENTITY Chap2 SYSTEM "http://www.tei-c.org/P4X/p4chap2.xml">

• External entities are referenced in the document just as internal ones are:
  <body> &Chap1; &Chap2; </body>

The Document Type Declaration

specifies:
• the root element of the document,
• the external entity containing the DTD
• and/or the (part of the) DTD contained in the internal subset
  e.g.
  • external:
    <!DOCTYPE anthology SYSTEM "anthology.dtd">
  • internal:
    <!DOCTYPE anthology [ 
      <!ELEMENT anthology (poem+)>
      <!ELEMENT poem (title?, stanza+)>
      <!ELEMENT title (#PCDATA) >
      <!ELEMENT stanza (line+) >
      <!ELEMENT line (#PCDATA) >
    ]>
  • mixed:
    <!DOCTYPE anthology SYSTEM "anthology.dtd" [ 
      <!ENTITY jbw "Jabberwocky"> 
    ]>

A Complete Valid XML Document

<?xml version="1.0" encoding="us-ascii"?>
<!DOCTYPE anthology [ 
  <!ELEMENT anthology (poem+)>
  <!ELEMENT poem (title?, stanza+)>
  <!ELEMENT title (#PCDATA) >
  <!ELEMENT stanza (line+) >
  <!ELEMENT line (#PCDATA) >
]>
<anthology>
  <poem>
    <title>The SICK ROSE</title>
    <stanza>
      <line>O Rose thou art sick.</line>
      <line>The invisible worm,</line>
      <line>That flies in the night</line>
      <line>In the howling storm:</line>
    </stanza>
    <stanza>
      <line>Has found out thy bed</line>
      <line>Of crimson joy:</line>
      <line>And his dark secret love</line>
      <line>Does thy life destroy.</line>
    </stanza>
  </poem>
</anthology>
Conclusion

- presented a brief introduction to XML
- Lab session: writing a (small) document in XML
  - select document, choose elements, write DTD, validate
  - maybe Bavarian-Style Pork Roast with Cabbage and Knödel?