Standards for Language Encoding

Tomaž Erjavec
Department for Intelligent Systems
Institute Jožef Stefan
Jamova 39
SI-1000 Ljubljana
Slovenia
tomaz.erjavec@ijs.si

Course notes for
ESSLLI'99
16 – 20 August 1999
Utrecht University

Contents

1 Course Description

Recent years have seen a rapid growth of interest in the acquisition, distribution and utilisation of large scale natural language resources. For text based resources, such as text corpora and lexica, the de-facto standard of encoding has become the Text Encoding Initiative's Guidelines, which are based on the ISO Standard Generalized Markup Language. The course introduces SGML and TEI and gives examples of their use in the encoding of natural language resources.

The course starts with an introduction to the SGML standard, covers the motivation and principles behind SGML, the structure of SGML document type definitions, entities and documents. Presented are some better known SGML document types and related standards, as well as expected future developments of the standard (XML). Conversion to and from SGML documents is discussed, and some freely available tools for processing SGML documents are presented.

The TEI Guidelines are described next. We give the structure of conformant TEI documents, the documents types covered by TEI and the method of applying and parameterizing TEI for various text types and specific projects. The mark-up of text corpora is covered in more depth, esp. for the purpose of language engineering. We discuss the Corpus Encoding Standard, an application of TEI, and overview TEI corpus and text headers, structural and linguistic markup. Case studies of existing corpora are given, concentrating on multilingual and parallel corpora. Finally, we introduce the encoding of machine readable dictionaries and the issues in moving from the so called editorial view of the dictionary into its lexical view are mentioned.

The course should give a good grounding in TEI and enable students to construct or exploit TEI annotated language resources.

This component of the ESSLLI'99 programme is sponsored by the European Chapter of the Association for Computational Linguistics (EACL).

2 What is SGML?

From *The Gentle Introduction*:[1]

> "SGML is an international standard for the definition of device-independent, system-independent methods of representing texts in electronic form"
>
> "SGML is a metalanguage, that is, a means of formally describing a language, in this case, a markup language"

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

The official designation of SGML:

> International Organization for Standardization, `ISO 8879: Information processing---Text and office systems---Standard Generalized Markup Language (SGML)`, ([Geneva]: ISO, 1986).

Hence, SGML is *not proprietary.*

[1] For this and other references see last Section

2.1 Some SGML history

In 1969, Charles Goldfarb leads an **IBM** research project on integrated law office information systems. With E. Mosher and R. Lorie he invented the Generalized Markup Language (GML) as a means of allowing text editing, formatting, and information retrieval subsystems to share documents.

The first working draft of the SGML standard was published in 1980 by **ANSI**. By 1983, the sixth working draft is recommended as an industry standard (GCA 101-1983). Major adopters included US IRS and DoD.

A draft **ISO** standard was published in October 1985, and was adopted by the Office of Official Publications of the EU. Another year of review and comment resulted in the final text, which was published in record time after approval (**ISO 8879:1986**).

2.2 Characteristics of SGML

- **Descriptive Markup**

  Markup codes **categorize** parts of a document; they do not tell what processing is to be carried out at particular points in a document (procedural markup).

  E.g.:

  - "the following item is a paragraph"
  - "skip down one line, move 5 quads right"

  In SGML, instructions needed to process a document for some particular purpose (for example, to format it) are sharply distinguished from the descriptive markup which occurs within the document. Usually, they are collected outside the document in separate procedures or programs.

- **Document Types**

  Documents are regarded as having types, and these are expressed by document type definitions (**DTD**), which enforce markup for that document type.

- **Data Independence**

  SGML encoded documents should be transportable from one hardware and software environment to another without loss of information: platforms differ in character sets, file-naming conventions, interpretation of bytes...

  SGML provides a general purpose mechanism for **string substitution**, that is, a simple machine-independent way of stating that a particular string of characters in the document should be replaced by some other string when the document is processed.

2.3 Where is SGML used?

- managing large amounts of valuable, (predominantly) textual data
- ensuring longevity of the texts
- enabling interchange of data between computer platforms
- facilitating interchange of data between people
- allowing multiple exploitation of texts

For example:

- industry: technical manuals
- publishers
- law firms
- libraries
- WWW
- language data for "language technologies":
  - **corpora**
  - dictionaries, lexicons
  - ...

Some important initiatives:

- Text Encoding Initiative (**TEI**)
- EAGLES and other EU projects
- DocBook (computer manuals)

2.4 Examples of SGML markup

- simple ASCII
- markup distinguished from text
- markup encloses text
- markup contains other markup

From the Gentle Introduction:

```
<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>
          <!-- more poems go here    -->
</anthology>
```

An HTML example:

```
<P><BR>
Doctors recommend this resort for the treatment of
circulatory and central nervous systems illnesses. At
the same time Tusnad Spa is an ideal place for
tourists who enjoy an active relaxation.
<P>
<H2>Entertainment</H2>
The walks to the Hawk's Rock, to Lake St. Anna, to
the Lover's Clearing, the wanderings among fir trees,
the swimming pools with mezothermal water, the
special taste of the local bread rolls, the day and
night cafes, the rides by carriages drawn by horses,
the tours by coach to Lacul Rosu - Bicaz Pass or to
Sovata - Corund are some of the special attractions
of the resort.
<BR>
<CENTER>St. Anna's Lake <IMG SRC="t51.gif"></CENTER>
<BR>
<BR>
The resort has also convenient sporting facilities
for those who would like to play tennis or soccer.
Excellent possibilities for hickings. Sauna and
swimming pool.  Discotheques all evenings.
```

An example from CES, the Corpus Encoding Standard:

```
 <div id="Oro.1.2" type=chapter n=1>
   <head>1</head>
   <p id="Oro.1.2.2">
     <s id="Oro.1.2.2.1">&Icirc;ntr-o zi
     senin&abreve; &scedil;i friguroas&abreve; de
     aprilie, pe c&acirc;nd ceasurile b&abreve;teau
     ora treisprezece, <name type="person">Winston
     Smith</name>, cu b&abreve;rbia
     &icirc;nfundat&abreve; &icirc;n piept pentru a
     sc&abreve;pa de v&acirc;ntul care-l lua pe sus,
     se strecur&abreve; iute prin u&scedil;ile de
     sticl&abreve; ale <name type="place">Blocului
     Victoria</name>, de&scedil;i nu destul de
     repede pentru a &icirc;mpiedica un v&acirc;rtej
     de praf &scedil;i nisip s&abreve;
     p&abreve;trund&abreve; o dat&abreve; cu el.
     </s>
   </p>
...
 </div>
```

3 The structure of SGML documents

1. The **document instance**:
   - contains data (text)
   - contains markup
   - must contain/refer to a DTD
2. DTD, the **document type definition**:
   - defines the structure of a document
   - may be stored externally to a document
3. The **SGML declaration**:
   - usually common to all documents in an SGML installation
   - if it isn't a part of the document, the default **Reference Concrete Syntax** (RCS) is used

3.1 The SGML declaration

Defines the concrete SGML syntax used on a system:

- **delimiter roles**
  what is markup, what is text? e.g. `STAGO` = `<`
- **reserved names**
  what are the strings that mean something to SGML?
  e.g. `ELEMENT` used in element definitions
- **disallowed characters**
  which characters are forbidden to appear?
  e.g. control characters (`^L, ^G`)
- **naming rules**
  what characters can be used in markup names?
  Case-sensitivity of markup. e.g. it defines that `POEM` and `Poem` are the same tag
- **quantities**
  max. length of documents, markup names,...
  e.g. max. length of GI is 32
- **allowed SGML features**
  which features are in force?
  e.g. `DATATAG, OMITTAG, FORMAL`

The SGML declaration usually used is (something close to) the RCS.
An example: the TEI SGML declaration:

```
<!SGML "ISO 8879:1986"
-- TEI SGML declaration for local processing with SGMLS:
         extended capacity points
         full ASCII character set instead of ISO 646 subset
```

```
         ~Z (26) defined as end-of-file
         Namelen 32, Litlen 512, otherwise as for TEI interchange
   made 16 Aug 1990, rev. 9 Mar 1991 by CMSMcQ
        increased LITLEN to 2048, GRPCNT to  100 (LB Jul 93) --
CHARSET
BASESET  "ISO 646-1983//CHARSET
          International Reference Version (IRV)//ESC 2/5 4/0"
DESCSET  0       9      UNUSED
         9       1      9
         10      1      10
         11      2      UNUSED
         13      1      13
         14     12      UNUSED
         26      1      UNUSED   -- eof --
         27      5      UNUSED
         32     95      32
        127      1      UNUSED
BASESET  "ISO Registration Number 100//CHARSET
          ECMA-94 Right Part of Latin Alphabet Nr. 1//ESC 2/13 4/1"
DESCSET  128    32      UNUSED
         160    95      32
         255    1       UNUSED
--
CAPACITY PUBLIC "ISO 8879:1986//CAPACITY Reference//EN"
--
CAPACITY  SGMLREF
TOTALCAP    10000000
ENTCAP        500000
ENTCHCAP      500000
ELEMCAP       500000
GRPCAP       1000000
EXGRPCAP      500000
EXNMCAP       500000
ATTCAP        500000
ATTCHCAP      500000
AVGRPCAP      500000
NOTCAP        500000
NOTCHCAP      500000
IDCAP         500000
IDREFCAP      500000
MAPCAP        500000
LKSETCAP      500000
LKNMCAP       500000

SCOPE    DOCUMENT
SYNTAX
SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
                  16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
                  127 255
BASESET  "ISO 646-1983//CHARSET
          International Reference Version (IRV)//ESC 2/5 4/0"
DESCSET   0     128     0
FUNCTION RE     13
         RS     10
         SPACE  32
```

```
         TAB    SEPCHAR 9
     --  EOF    SEPCHAR 26 --
NAMING   LCNMSTRT ""
         UCNMSTRT ""
         LCNMCHAR "-."
         UCNMCHAR "-."
         NAMECASE GENERAL YES
                  ENTITY  NO
DELIM    GENERAL SGMLREF
         SHORTREF SGMLREF
NAMES    SGMLREF
QUANTITY SGMLREF
         ENTLVL   32
         NAMELEN  32
         LITLEN  2048
         GRPCNT  100
         GRPGTCNT  180
         TAGLVL   24
FEATURES
MINIMIZE DATATAG NO     OMITTAG  YES    RANK     NO     SHORTTAG YES
LINK     SIMPLE  NO     IMPLICIT NO     EXPLICIT NO
OTHER    CONCUR  NO     SUBDOC   YES    FORMAL   NO
APPINFO NONE>
```

4 The Document Type Definition

Each SGML document must refer to a DTD.

The Document Type Definition consists of:

- the **formal part**

  formally specifies what markup can appear in the document type and in what relation to other markup and data

- the **documentation**

  specifies what the defined elements **mean**:

  `<date>Joe</date>` can be formally OK, but is nonsense.

The formal DTD consists of:

- **ELEMENT** declarations
- **ATTRIBUTE** declarations
- **ENTITY** specifications

4.1 Element declarations

```
<!ELEMENT anthology      - -  (poem+)>
```

Is composed of:

`<!`	Markup declaration open delimiter
`ELEMENT`	declaration type keyword
`anthology`	**generic identifier**
`- -`	omitted tag minimization
`(poem+)`	**content model**
`>`	Markup declaration close delimiter

Some more examples:

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

The document according to the declarations:

```
<anthology>
   <poem>
     <stanza>
          <line>The first stanza.</line>
     <stanza>
          <line>Of a sad song</line>
          <line>this was.</line>
  </poem>
</anthology>
```

4.2 Model Group Operators

- (open bracket for grouping
-) close bracket
- , follows
- | or
- ? maybe
- * repeated 0 or more times
- + repeated once or more times

What to watch out for:

- **Content model ambiguity**:

  ```
  <!ELEMENT a   - - ((b, c)|(b, d))>
  ```

- **Mixed content**

  carriage return = record end (**RE**)

  In the DTD:

  ```
  <!ELEMENT listitem  - - (#PCDATA, para+)>
  <!ELEMENT para     - - (#PCDATA)>
  ```

In the document:

```
<listitem>
Oranges
<para>Oranges are a lovely fruit.</para>
<para>They grow in warm climates.</para>
</listitem>
```

Some further twists:

- A more complex poem type:
```
<!ELEMENT poem - O
          (title?,
          ( (line+)
          | (refrain?, (stanza, refrain?)+ ) ))>
```
- **Model Groups**:
```
<!ELEMENT (note | variant) - - (#PCDATA)>
```
- **EMPTY content**:
```
<!ELEMENT poemref - O EMPTY >
```
- **ANY content**:
```
<!ELEMENT memo - - ANY>
```
- **Inclusion / Exclusion Exceptions**:
```
<!ELEMENT poem - O
          (title?, (stanza+ | couplet+ | line+))
          +(note | variant) >

<!ELEMENT title - O
          (#PCDATA)
          -(note | variant) >

<!ELEMENT (note | variant) - -
          (#PCDATA)
          -(note | variant) >
```

4.3 Attribute declarations

```
<!ATTLIST poem
          id     ID                           #IMPLIED
          status (draft|revised|published) draft   >
```

In document: `<poem id=P1 status=revised> ... </poem>`
Some possible declared values:

- `CDATA`: any valid character data;
- `SDATA`: system dependent data;
- `NUMBER`: composed only of numerals;

- `ID`: a unique identifier, composed of any valid character data
- `IDREF`: a pointer to an ID

Some possible keywords:

- `#REQUIRED`: a value must be specified;
- `#IMPLIED`: a value need not be supplied;
- `#CURRENT`: if no value is supplied, the last specified value should be used.

4.4 Identifiers

In the DTD:

```
<!ATTLIST poem     id       ID     #IMPLIED  >

<!ELEMENT poemref  - O    EMPTY>

<!ATTLIST poemref  target   IDREF  #REQUIRED >
```

In the document:

```
    <anthology>
      <poem id="p001"><title>The SICK ROSE</title>
         Text of the poem
      </poem>
      <poem id="p002"><title>The Jabberwocky</title>
         Text of the poem
      </poem>
<!-- more poems go here -->
      <poem><title>Hail to thee, o SGML</title>
         This poem has no identifier
      </poem>
    </anthology>

<!-- and somewhere else in the document: -->

Blake's poem on the sick rose <POEMREF TARGET="p001">
```

5 Entities

Entities are divided into:

- **General entities**

  used in the document instance and divided into:

  - **parameter literals**
  - **external entity specifications**
  - (data text)
  - (bracketed text)

- **Parameter entities**

  used in the DTD

5.1 Parameter literals

In the DTD:

```
<!ENTITY tei "Text Encoding Initiative">
```

In the document:

```
The work of the &tei; has only just begun.
```

After processing:

```
The work of the Text Encoding Initiative has only just begun.
```

Parameter literals (data text) are very important for system independent character set representations:

In the DTD:

```
<!ENTITY abreve SDATA "[abreve]"--=small a, breve-->
<!ENTITY Abreve SDATA "[Abreve]"--=capital A, breve-->
<!ENTITY amacr  SDATA "[amacr ]"--=small a, macron-->
<!ENTITY Amacr  SDATA "[Amacr ]"--=capital A, macron-->
<!ENTITY aogon  SDATA "[aogon ]"--=small a, ogonek-->
<!ENTITY Aogon  SDATA "[Aogon ]"--=capital A, ogonek-->
...
```

In the document:

```
<s>&Icirc;ntr-o zi senin&abreve; &scedil;i
friguroas&abreve; de aprilie, pe c&acirc;nd ceasurile
...
```

If the system does not support the character, it can be displayed sloppily:

```
<!ENTITY abreve SDATA "a"--=small a, breve-->
```

If the system does support the character, but it cannot be entered directly, the replacement text is a **character entity**, e.g.:

```
<!ENTITY abreve SDATA "&#123;"--=small a, breve-->
```

5.2 External entity specifications

On processing, these entities expand to the contents of the files.

In the DTD:

```
<!ENTITY ChapTwo SYSTEM "sgmlmkup.txt">
```

In the document:

```
<body>
&ChapOne;
&ChapTwo;
...
</body>
```

External entity specifications can refer to `SYSTEM` or `PUBLIC` identifiers.

```
<!ENTITY mte0-ro PUBLIC "-//MTE//TEXT CES1 1984//RO">
```

If `FORMAL=YES` then `PUBLIC` identifier must be properly structured.

5.3 Parameter Entities

Used in the **DTD**.

Including entities for the **character set**:

```
<!ENTITY % ISOlat1 PUBLIC
         "ISO 8879-1986//ENTITIES Added Latin 1//EN">
%ISOlat1;
```

Compact **repetitive definitions**:

```
<!ENTITY % a.global '
           id        ID       #IMPLIED
           n         CDATA    #IMPLIED
           lang      IDREF    #IMPLIED' >

<!ATTLIST cesCorpus  %a.global;
          type       CDATA    #IMPLIED
          version    CDATA    #REQUIRED >
```

5.4 Marked Sections

In the DTD:

```
<!ENTITY % ENGLISH  "IGNORE">
<!ENTITY % ROMANIAN "INCLUDE">
<!ENTITY % SLOVENE  "IGNORE">
```

In the document:

```
<![ %ENGLISH [
<s>The hallway smelt of boiled cabbage and
old rag mats.</s>
]]>

<![ %ROMANIAN [
<s>Holul blocului mirosea a varz&abreve;
c&abreve;lit&abreve; &scedil;i a pre&scedil;uri
vechi.</s>
]]>

<![ %SLOVENE [
<s>Ve&zcaron;a je smrdela po kuhanem zelju
in starih, cunjastih predpra&zcaron;nikih.</s>
]]>
```

Modularisation in TEI (the **Chicago Pizza** model):
In the TEI DTD:

```
<![ %TEI.dictionaries [
<!ENTITY % TEI.dictionaries.dtd PUBLIC
 "-//Text Encoding Initiative//DTD
  P3 12.1: Base Tag Set for Printed Dictionaries//EN//"
>
%TEI.dictionaries.dtd;
]]>
```

When using it:

```
<!ENTITY % TEI.dictionaries  "INCLUDE">
```

6 SGML Documents

- every document must refer to a DTD
- every document must contain a DOCTYPE declaration

A complete document:

```
<!DOCTYPE anthology [
    <!ELEMENT anthology   - -  (poem+)>
    <!ELEMENT poem        - -  (title?, stanza+)>
    <!ELEMENT title       - O  (#PCDATA) >
    <!ELEMENT stanza      - O  (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>
              <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>
      </poem>
    </anthology>
```

6.1 External DTDs

Usually a DTD is external to the document:

```
<!DOCTYPE anthology SYSTEM "antology.dtd">

<!DOCTYPE anthology SYSTEM "antology.dtd" []>

<!DOCTYPE cesCorpus PUBLIC "-//CES//DTD cesDoc//EN">

<!DOCTYPE cesCorpus PUBLIC "-//CES//DTD cesDoc//EN"
[
  <!ENTITY mte1984-en PUBLIC
           "-//MTE//TEXT CES1 1984//EN">
...
  <!ENTITY mte1984-ro PUBLIC
           "-//MTE//TEXT CES1 1984//RO">
  <!ENTITY mte1984-sl PUBLIC
           "-//MTE//TEXT CES1 1984//SL">
]>
<cesCorpus id="o" version="0.1">
```

```
  <cesHeader> ... </cesHeader>
  &mte1984-en;
...
  &mte1984-ro;
  &mte1984-sl;
</cesCorpus>
```

6.2 Public identifiers & Catalog

(SGML Open Technical Resolution TR9401:1995)

PUBLIC identifiers:

```
"ISO 8879-1986//ENTITIES Added Latin 1//EN"

"-//Text Encoding Initiative//DTD
  P3 3.6.1: Main TEI document type declaration//EN//"

"-//CES//DTD cesDoc//EN"

"-//MTE//ENTITIES Romanian//EN"
```

Mapping to system identifiers (filenames) is performed in the **catalog** file, e.g. for HTML:

```
-- $Id: HTML3.2.catalog,v 1.1
        1996/05/06 connolly Exp $ --

PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN"
       /usr/local/sgml/HTML/HTML32.dtd
PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"
       /usr/local/sgml/HTML/HTML32.dtd
PUBLIC "-//W3C//DTD HTML 3.2//EN"
       /usr/local/sgml/HTML/HTML32.dtd
```

6.3 The complete picture

- the **SGML declaration**,
  if not using the RCS
- the **catalog file(s)**,
  if using public identifiers
- the **DTDs**,
  stored as specified in the catalog or SYSTEM identifiers
- the **document instances**,
  that you are creating or using
- **SGML aware software**,
  to do something with all this

7 Related Standards

There are many standards or recommendation related to SGML, which define SGML DTDs, ipmlement style sheets, or linking.

SGML DTDs:

- DocBook: DTD for computer manuals (Industry consortium)
- ISO 12083:1993: DTD for printed materials (article, book, math, serial)
- Computer Graphics Metafile (CGM). ISO 8632:1992.

Style languages:

- DSSSL: ISO/IEC 10179:1996 DSSSL (Document Style Semantics and Specification Language)

  DSSSL, like SGML, is declarative in nature, and has no biases toward national language, writing direction, or character set. All style specifications in DSSSL are made by describing final formatting results, not by describing the algorithms to be used create the formatting result. In other words, DSSSL enables us to specify how data appears in output (print or electronic) but not how to create that format.

- CSS: Cascading stylesheets, developed by the W3C.

  a style sheet language that allows authors and users to attach style (e.g., fonts, spacing, and aural cues) to structured documents (e.g., HTML documents and XML applications). By separating the presentation style of documents from the content of documents, CSS2 simplifies Web authoring and site maintenance.

Linking languages:

- HyTime (ISO/IEC 10744:1992)

  allows the description of virtually any kind of connected or temporal information in SGML. It covers many different aspects on linking, as well as a full repertoir of features for multimedia purposes, including virtual time, scheduling, synchronisation and so on. Is incredibly complicated.

- TEI Extended Pointer syntax

  "Linking, Segmentation, and Alignment." Chapter 14 (pages 393-454) of the TEI Guidelines for Electronic Text Encoding and Interchange.

8 Extensible Markup Language: XML

XML is a data format for structured document interchange on the Web. The current W3C Recommendations are XML 1.0, Feb '98.

XML is designed 'to make it easy and straightforward to use SGML on the Web: easy to define document types, easy to author and manage SGML-defined documents, and easy to transmit and share them across the Web.'

It defines 'an extremely simple dialect of SGML which is completely described in the XML Specification. The goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML.'

'For this reason, XML has been designed for ease of implementation, and for interoperability with both SGML and HTML' [quotes from the XML spec].

Main differences between XML and SGML:

- no tag minimisation (many 'NO' in the SGML declaration)
- it is also different from the RCS (unlimited quantities, capacities; case sensitive names which can use Unicode characters)
- 'and' connector not allowed
- element inclusion and exclusions not allowed
- content models for mixed content have a restricted form (repeatable "or" group only)
- INCLUDE/IGNORE marked sections are not allowed in the document instance
- **EMPTY elements** are closed with `/>`, e.g. `<anchor id="x.2.3"/>`

Also important:

- valid XML document: DTD + document instance
- well-formed XML document: document instance only

Related recommendations:

- XSL: Extensible Stylesheet Language
- XLL: XML Linking Languages

9 Processing SGML

No SW implements the complete SGML standard. But some come close.

SGML aware SW is either free or very expensive.

SGML software components:

- editing tools and environments
- conversion engines and applications
- the document management system
- formatting engines for rendering on paper and screen
- other transformation engines
- search and retrieval engines
- additional processing applications
- the "document engineering toolbox"

9.1 SGML validating parser

The basic question for a developer:

- Is the document well formed?
- getting a normalised instance of the document

The best choice:
James Clark's `SP`
SP is a "object-oriented toolkit for SGML parsing and entity management."

- implements most of the standard
- freely available
- runs on most platforms
- bug-free
- works with XML

Validating with `nsgmls`:
The document `mixed.sgml`:

```
<!doctype listitem [
   <!ELEMENT listitem  - - (#PCDATA, para+)>
   <!ELEMENT para      - - (#PCDATA)>
]>
<listitem>
Oranges
<para>Oranges are a lovely fruit.</para>
<para>They grow in warm climates.</para>
</listitem>
```

`nsgmls` as validator:

```
nsgmls -s mixed.sgml
nsgmls:mixed.sgml:7:40:E: character data is not allowed here
nsgmls:mixed.sgml:8:40:E: character data is not allowed here

SGML validation exited abnormally with code 1
```

Parsing with `nsgmls`
Fixed document:

```
<!doctype listitem [
   <!ELEMENT listitem  - - (#PCDATA, para+)>
   <!ELEMENT para      - - (#PCDATA)>
]>
<listitem>
Oranges
<para>
Oranges are a lovely fruit.
</para><para>
They grow in warm climates.
</para></listitem>
```

nsgmls ESIS output:

```
nsgmls mixed.sgml
(LISTITEM
-Oranges\n
(PARA
-Oranges are a lovely fruit.
)PARA
(PARA
-They grow in warm climates.
)PARA
)LISTITEM
C

SGML validation finished at Fri Jul 11 13:26:51
```

9.2 Editing

The only freely available editor is Emacs (GNU or X) in conjunction with **psgml mode**. From the mode help:

```
SGML mode:
Major mode for editing SGML.
Makes > display the matching <.  Makes / display matching /.
Use C-c C-v to validate your document with an SGML parser.

You can find information with:
C-c C-c  Show the nesting of elements at cursor position.
C-c C-t  Show the tags valid at cursor position.

Insert tags with completion of contextually valid tags with C-c <.
End the current element with C-c /.  Insert an element (i.e.
both start and end tag) with C-c C-e.  Or tag a region with
C-c C-r.

To tag a region with the mouse, use transient mark mode or secondary selection.

Structure editing:
M-C-b  Moves backwards over the previous element.
M-C-f  Moves forward over the nex element.
M-C-d  Move forward and down one level in the element structure.
M-C-u  Move backward out of this element level.
M-C-a  Move to after the start tag of the current element.
M-C-e  Move to before the end tag of the current element.
M-C-k  Kill the element following the cursor.

Finding interesting positions
C-c C-d  Move forward to next point where data is allowed.
C-c C-o  Move forward to next point where something is
        amiss with the structure.

Folding and unfolding
C-c C-f C-e  Fold the lines comprising the current element, leaving
        the first line visible.
```

```
C-c C-f C-s  Fold the elements in the content of the current element.
        Leaving the first line of every element visible.
C-c C-s  Show hidden lines in current line.
```

9.3 Conversion Engines

Often it is simplest to use Unix scripts or Perl (Python), esp. on normalised instances of SGML documents.

A freely available SGML processing engine is Omnimark Light (Omle). A snippet from an Omnimark program:

```
;
; Handle Images...
;
ELEMENT IMG
        OUTPUT "<%q "
        DO WHEN ATTRIBUTE src IS SPECIFIED
                OUTPUT "  src=%"%v(src)%" "
        DONE
        DO WHEN ATTRIBUTE align IS SPECIFIED
                OUTPUT "  align=%"%v(align)%" "
        DONE
        DO WHEN ATTRIBUTE ismap IS SPECIFIED
                OUTPUT "  ISMAP"
        DONE
        OUTPUT ">%sc"
```

9.4 Formatting

Jade, by J. Clark, is an application of DSSSL. Jade takes as input an SGML DTD, a document marked up in the language specified by the DTD, and a DSSSL stylesheet. It produces as output an SGML representation of the resulting DSSSL flow object tree or, alternatively, RTF of TeX output. RTF and TeX backends are provided with the distribution.

9.5 DTD induction

FRED automatically builds a DTD from a document instance. Very useful for 'bottom up' DTD design or for inducing minimal DTDs of documents.

9.6 Processing of SGML corpora

LT NSL (by the Language Technology Group at the Human Communication Research Centre, University of Edinburg) was developed to allow direct processing of very large SGML-marked text collections called corpora. As such, the requirements of this type of processing exceed those of most publishing and information-management environments in some respects and undershoot them in others. A typical body of work used in linguistic processing can be millions of words of data, with a markup density exceeding a pair of tags for every word, but there are no requirements for rendering, and little expectation that anyone will actually read what is processed.

LT NSL is an integrated set of SGML querying/manipulation tools and a C-language application program interface (API) designed to ease the writing of C programs which manipulate SGML documents. Its API is based on the idea of using 'normalised' SGML (i.e. an expanded, easily parsable subset of SGML) as a data format for inter-program communication of structured textual information. The API defines a powerful query language which makes it easy to access (either from the shell or in a program) those parts of an SGML document which you are interested in. Both event based and (sub-)tree based views of SGML documents are supported.

10 The Text Encoding Initiative

From the Preface of the **TEI Guidelines for Electronic Text Encoding and Interchange**:

> These Guidelines are the result of over five years' effort by members of the research and academic community within the framework of an international cooperative project called the Text Encoding Initiative (TEI), established in 1987 under the joint sponsorship of the Association for Computers and the Humanities, the Association for Computational Linguistics, and the Association for Literary and Linguistic Computing.
>
> The impetus for the project came from the humanities computing community, which sought a common encoding scheme for complex textual structures in order to reduce the diversity of existing encoding practices, simplify processing by machine, and encourage the sharing of electronic texts. It soon became apparent that a sufficiently flexible scheme could provide solutions for text encoding problems generally. The scope of the TEI was therefore broadened to meet the varied encoding requirements of any discipline or application. Thus, the TEI became the only systematized attempt to develop a fully general text encoding model and set of encoding conventions based upon it, suitable for processing and analysis of any type of text, in any language, and intended to serve the increasing range of existing (and potential) applications and use.
>
> What is published here is a major milestone in this effort. It provides a single, coherent framework for all kinds of text encoding which is hardware-, software- and application-independent. Within this framework, it specifies encoding conventions for a number of key text types and features. The ongoing work of the TEI is to extend the scheme presented here to cover additional text types and features, as well as to continue to refine its encoding recommendations on the basis of extensive experience with their actual application and use.
>
> We therefore offer these Guidelines to the user community for use in the same spirit of active collaboration and cooperation with which they have so far been developed. The TEI is committed to actively supporting the wide-spread and large-scale use of the Guidelines which, with the publication of this volume, is now for the first time possible. In addition, we anticipate that users of the TEI Guidelines will in some instances adapt and extend them as necessary to suit particular needs; we invite such users to engage in the further development of the Guidelines by working with us as they do so.
>
> Like any standard which is actually used, these Guidelines do not represent a static finished work, but rather one which will evolve over time with the active involvement of its community of users. We invite and encourage the participation of the the user community in this process, in order to ensure that the TEI Guidelines become and remain useful in all sorts of work with machine-readable texts.

Recently (March '99), a **TEI consortium** has been founded. From the WWW page:

A new consortium has been formed for the maintenance and continuing work of the Text Encoding Initiative (TEI). The TEI is an international project to develop guidelines for the encoding of textual material in electronic form for research purposes; until now, it had been organized as a simple cooperative effort of the three sponsors, and funded solely by grant funds. Now four universities have agreed to serve as hosts for the new consortium, and the three organizations which founded the TEI and have governed it until now have agreed to transfer the responsibility for maintaining and revising the TEI Guidelines to the new consortium.

11 Structure of the TEI DTD

The TEI encoding scheme consists of number of modules ('tagsets') or DTD fragments. The DTD fragments from which the main TEI DTD is constructed are classified as follows:

- core DTD fragments

  standard components of the TEI main DTD in all its forms; these are always included without any special action by the encoder;

- base DTD fragments

  basic building blocks for specific text types; exactly one base must be selected by the encoder (unless one of the combined bases is used);

- additional DTD fragments

  extra tags useful for particular purposes. All additional tag sets are compatible with all bases and with each other; an encoder may therefore add them to the selected base in any combination desired.

11.1 Core tagset

Consists of:

- core tags:

  These include paragraph, `<p>`, highlighting (`<emph>`, `rend`), quotation, `<q>`, names, numbers, dates, abbreviations, etc.

- the TEI header:

  describes an encoded work so that the text itself, its source, its encoding, and its revisions are all thoroughly documented

11.2 The TEI Header

The header consits of four parts:

- a file description, `<fileDesc>`, containing a full bibliographical description of the computer file itself, from which proper bibliographic citation can be derived. The file description includes information about the source or sources from which the electronic text was derived.

- an encoding description, <`encodingDesc`>, which describes the relationship between an electronic text and its source or sources. It allows for detailed description of whether (or how) the text was normalized during transcription, how the encoder resolved ambiguities in the source, what levels of encoding or analysis were applied, and similar matters.
- a text profile, <`profileDesc`>, containing classificatory and contextual information about the text, such as its subject matter, the situation in which it was produced, the individuals described by or participating in producing it, and so forth. Such a text profile is of particular use in highly structured composite texts such as corpora or language collections, where it is often highly desirable to enforce a controlled descriptive vocabulary or to perform retrievals from a body of text in terms of text type or origin.
- a revision history, <`revisionDesc`>, which allows the encoder to provide a history of changes made during the development of the electronic text. The revision history is important for version control and for resolving questions about the history of a file.

11.3 Base tagsets

- TEI.prose: the base tag set for prose
- TEI.verse: the base tag set for verse
- TEI.drama: the base tag set for drama
- TEI.spoken: the base tag set for transcriptions of spoken texts
- TEI.dictionaries: the base tag set for print dictionaries
- TEI.terminology: the base tag set for terminological data files
- TEI.general: the generic mixed-mode base base tag set
- TEI.mixed: the base tag set for free mixed-mode texts

11.4 Additional tagsets

- TEI.linking: tags for linking, segmentation, and alignment
- TEI.analysis: tags for simple analytic mechanisms
- TEI.fs: tags for feature structure analysis
- TEI.certainty: tags for indicating uncertainty and probability in the markup
- TEI.transcr: tags for manuscripts, analytic bibliography, and transcription of primary sources
- TEI.textcrit: tags for critical editions
- TEI.names.dates: specialized tags for names and dates
- TEI.nets: tags for graphs, digraphs, trees, and other networks
- TEI.figures: tags for graphics, figures, illustrations, tables, and formulae
- TEI.corpus: tags for additional tags for language corpora

11.5 Parametrisation

```
<!DOCTYPE teiCorpus.2 PUBLIC
  "-//TEI P3//DTD Main Document Type//EN" [
  <!-- base tag set for prose: -->
  <!ENTITY % TEI.prose    'INCLUDE'>
  <!-- add basic linguistic analysis: -->
  <!ENTITY % TEI.analysis 'INCLUDE'>
  <!-- add pointer mechanisms: -->
  <!ENTITY % TEI.linking  'INCLUDE'>
  <!-- add local extensions: -->
  <!ENTITY % TEI.extensions.ent
    SYSTEM "teitmx.ent">
  <!ENTITY % TEI.extensions.dtd
    SYSTEM "teitmx.dtd">
]>
```

The two extension files:

```
teitmx.ent:
<!ENTITY % body 'IGNORE' >

teitmx.dtd:
<!ELEMENT %n.body;      - -  (tu+)>
<!ELEMENT tu            - -  (seg, seg)>
<!ATTLIST tu                 %a.global;>
```

In the SGML document:

```
<text lang="sl-en" id="vade.T">
  <body>
    <tu lang="sl-en" id="vade.1">
      <seg lang="sl"><w>Uvod</w><c>:</c></seg>
      <seg lang="en"><w>Introduction</w><c>:</c></seg>
    </tu>
    <tu lang="sl-en" id="vade.2">
      <seg lang="sl"><w>Kultura</w> <w>samozdravljenja</w> <w>pri</w>
<w>nas</w> <w>in</w> <w>po</w> <w>svetu</w></seg>
      <seg lang="en"><w>The</w> <w>culture</w> <w>of</w>
<w>selfmedication</w> <w>in</w> <w>Slovenia</w> <w>and</w>
<w>world-wide</w></seg>
</tu>
```

11.6 Global attributes

- `id`

  provides a unique identifier for the element bearing the ID value.

- `n`

  gives a number (or other label) for an element, which is not necessarily unique within the document.

- `lang` indicates the language of the element content, usually using a two- or three-letter code from ISO 639.

- `rend`

  indicates how the element in question was rendered or presented in the source text.

11.7 TEI Lite

A particular parametrisation of TEI (a DTD), which implements a useful 'starter set', comprising the elements which almost every user should know about.

Some characteristcs of TEI Lite:

- includes most of the TEI 'core' tag set, since this contains elements relevant to virtually all text types and all kinds of text-processing work
- handles a reasonably wide variety of texts, at the level of detail found in existing practice
- is useful for the production of new documents as well as encoding of existing ones;
- is usable with a wide range of existing SGML software;
- is derivable from the full TEI DTD using the extension mechanisms described in the TEI Guidelines;
- is as small and simple as is consistent with the other goals.

TEI Lite has also been ported to XML.

12 Corpus Encoding Standard

CES is an effort supproted by EAGLES (the EAGLES Text Representation subgroup), European projects MULTEXT (LRE) and MULTEXT-East. It aims to develop a Corpus Encoding Standard (CES) optimally suited for use in **language engineering**, which can serve as a widely accepted set of encoding standards for corpus-based work. The overall goal is the identification of a minimal encoding level that corpora must achieve to be considered standardized in terms of descriptive representation (marking of structural and linguistic information) as well as general architecture (so as to be maximally suited for use in a text database). It also provides encoding conventions for more extensive encoding and for linguistic annotation. CES is, to a large extent, TEI conformant.

CES provides:

- a set of metalanguage level recommendations (particular profile of SGML use, character sets, etc.);
- tagsets and recommendations for documentation of encoded data;
- tagsets and recommendations for encoding primary data, including written texts across all genres, for the purposes of corpus-based work in language engineering;
- tagsets and recommendations for encoding linguistic annotation commonly associated with texts in language engineering, including:
  - segmentation of the text into sentences and words (tokens),
  - morpho-syntactic tagging,
  - parallel text alignment.

Linguistic annotation is encoded in separate documents, linked to the primary data.

12.1 The cesDoc DTD and levels of encoding

This DTD is used for encoding primary data. It is very similar to TEI Lite, but redefines some TEI elements, e.g. the title in the header is `<h.title>`, not `<title>`.

For the encoding of primary data the CES identifies three levels of encoding:

- Level 1
  This is the minimum encoding level required for CES conformance, requiring markup for gross document structure (major text divisions), down to the level of the paragraph, conformant to the cesDoc DTD.

- Level 2
  This level requires that paragraph level elements are correctly marked, and (where possible) the function of rendition information at the sub-paragraph level is determined and elements marked accordingly.

- Level 3
  This is the most restrictive and refined level of markup for primary data. It places additional constraints on the encoding of s-units and quoted dialogue, and demands more sub-paragraph level tagging.

And example of a Level 3 document fragment:

```
 <text  lang="sl">
  <body lang="sl" id="Osl">
   <div id="Osl.1"  type=part n=1>
    <head>Prvi del</head>
     <div id="Osl.1.2"  type=chapter n=1>
      <head>I</head>
      <p id="Osl.1.2.2">
       <s id="Osl.1.2.2.1">Bil je jasen, mrzel aprilski dan
in ure so bile trinajst.</s>
       <s id="Osl.1.2.2.2"><name>Winston Smith</name> je imel
brado zakopano v prsi, da bi u&scaron;el strupenemu vetru, ko
je stopil skozi steklena vrata bloka Zmaga, vendar ne dovolj
hitro, da ne bi vrtinec pe&scaron;&ccaron;enega prahu vstopil
skupaj z njim.</s>
</p>
...
<s id="Osl.1.2.8.3">S kraja, kjer je stal <name>Winston</name>,
se je ravno &scaron;e dalo prebrati tri partijske parole, ki
so se v lepih &ccaron;rkah odra&zcaron;ale z belega pro&ccaron;elja:
<q id="Osl.1.2.8.3.2" rend="CN CA" type=slogan>Vojna je mir</q>
<q id="Osl.1.2.8.3.3" rend="CN CA" type=slogan>Svoboda je
su&zcaron;enjstvo</q>
<q id="Osl.1.2.8.3.4" rend="CN CA" type=slogan>Nevednost je
mo&ccaron;</q></s>
</p>
...
```

12.2 The cesAlign DTD

The annotation document containing alignment information consists entirely of links between the documents that have been aligned.

```
<!doctype cesAlign PUBLIC "-//CES//DTD cesAlign//EN" []>
<cesAlign version="4.1">
  <linkList id="Oslen">
    <linkGrp id="Oslen.1" type="body" targtype="s" domains="Oen Osl">
      <link xtargets="Osl.1.2.2.1 ; Oen.1.1.1.1">
      <link xtargets="Osl.1.2.2.2 ; Oen.1.1.1.2">
...
      <link xtargets="Osl.1.2.6.6 ; Oen.1.1.5.6 Oen.1.1.5.7">
...
      <link xtargets="Osl.1.2.22.1.4 ; Oen.1.1.21.1.4">
      <link xtargets="Osl.1.2.22.1.5 Osl.1.2.22.1.6 Osl.1.2.22.1.7 ;
                      Oen.1.1.21.1.5 Oen.1.1.21.1.6 Oen.1.1.21.1.7">
      <link xtargets="Osl.1.2.22.1.8 Osl.1.2.22.1.9 ; Oen.1.1.21.1.8">
      <link xtargets="Osl.1.2.22.1.10 ; Oen.1.1.21.1.9">
...
```

12.3 The cesAna DTD

The cesAna DTD is used for segmentation and grammatical annotation, including:

- sentence boundary markup
- tokens, each of which consists of the following:
  - the orthographic form of the token as it appears in the corpus
  - grammatical annotation, comprising one or more sets of the following:
    - the base form (lemma)
    - a morpho-syntactic specification, in the EAGLES annotation style
    - a corpus tag

Example:

```
<chunkList type=TEXT lang=sl>
 <chunk type=BODY lang=sl>
  <par from='Osl.1.2.2'>
   <s from='Osl.1.2.2.1'>
    <tok type=WORD>
     <orth>Bil</orth>
     <disamb><base>biti</base><msd>Vcps-sma</msd></disamb>
     <lex><base>biti</base><msd>Vcps-sma</msd></lex>
     <lex><base>biti</base><msd>Vmps-sma</msd></lex>
    </tok>
    <tok type=WORD>
     <orth>je</orth>
     <disamb><base>biti</base><msd>Vcip3s-an</msd></disamb>
     <lex><base>biti</base><msd>Vcip3s-an</msd></lex>
     <lex><base>jesti</base><msd>Vmip3s--n</msd></lex>
     <lex><base>on</base><msd>Pp3fsg--y-n</msd></lex>
    </tok>
    <tok type=WORD>
     <orth>jasen</orth>
     <disamb><base>jasen</base><msd>Afpmsn</msd></disamb>
     <lex><base>jasen</base><msd>Afpmsn</msd></lex>
     <lex><base>jasen</base><msd>Afpmsa</msd></lex>
```

```
      </tok>
      <tok type=PUNCT>
       <orth>,</orth>
       <ctag>COMMA</ctag>
      </tok>
      <tok type=WORD>
       <orth>mrzel</orth>
       <disamb><base>mrzel</base><msd>Afpmsn</msd></disamb>
       <lex><base>mrzel</base><msd>Afpmsn</msd></lex>
       <lex><base>mrzel</base><msd>Afpmsa</msd></lex>
      </tok>
...
      <tok type=PUNCT>
       <orth>.</orth>
       <ctag>PERIOD</ctag>
      </tok>
     </s>
```

13 Further Reading

There is a wealth of information about SGML and TEI in books, and on the Internet. Below are just some pointers to get you started:

- A good introduction to SGML, from where lots of examples for this course have been taken:

  C.M. Sperberg-McQueen and Lou Burnard:
  Gentle Introduction to SGML.
  In *Guidelines for Electronic Text Encoding and Interchange* (C.M. Sperberg-McQueen, Lou Burnard eds.), Chicago and Oxford, 1994.

- The comprehensive SGML WWW pages:

  Robin Cover:
  The SGML Web Pages

- Where to get help and find the latest developments:

  the SGML newsgroup `comp.text.sgml`

- The ultimate reference (full and annotated standard):

  Charles F. Goldfarb:
  The SGML Handbook
  Clarendon Press, Oxford, 1990.
  ISBN 0-19-853737-9

- The TEI Green Book:

  C.M. Sperberg-McQueen and Lou Burnard:
  Guidelines for Electronic Text Encoding and Interchange (C.M. Sperberg-McQueen, Lou Burnard eds.), Chicago and Oxford, 1994.

  Available also at *http://etext.virginia.edu/TEI.html*

- The TEI home page:
  http://www-tei.uic.edu/orgs/tei/
  (also contains 'TEI Lite: An Introduction to Text Encoding for Interchange')
- The Corpus Encoding Standard home page:
  http://www.cs.vassar.edu/CES/CES1.html