TEI BY EXAMPLE

MODULE 0: INTRODUCTION TO TEXT ENCODING AND THE TEI

Edward Vanhoutte

Ron Van den Branden

Melissa Terras

Centre for Scholarly Editing and Document Studies (CTB) , Royal Academy of Dutch Language and Literature, Belgium, Gent, 9 July 2010

Last updated September 2020

TABLE OF CONTENTS

1. Introduction

Computers can only process texts whose characters are represented by a system that relates to the binary system computers can interpret. This is called *character encoding*. One such character encoding scheme based on the English alphabet is ASCII (American Standard Code for Information Interchange). Character encoding facilitates the storage of text in computers and the transmission of text through telecommunication networks. Character encoding, however, does not say anything about the semantics, interpretation or structure of a text. Such information on a text is called *meta-information*. If we want to add any meta-information to a text so that it can be processed by computers, we need to encode or markup texts. We can do this by inserting natural language expressions (or codes representing them) in the text with the same character encoding the text is using, but separated from the text by specific markers. One such an expression, we call a *tag*. All of the tags used to encode a text together constitute a *markup language*. The application of a markup language to a text, we call *text encoding*.

The Text Encoding Initiative (TEI) is a standard for the representation of textual material in digital form through the means of text encoding. This standard is the collaborative product of a community of scholars, chiefly from the humanities, social sciences, and linguistics, who are organised in the TEI Consortium (TEI-C, https://tei-c.org). The TEI Consortium is a non-profit membership organisation which governs a wide variety of activities such as the development, publication, and maintenance of the text encoding standard documented in the TEI Guidelines, the discussion and development of the standard on the TEI mailing list (TEI-L) and in Special Interest Groups (SIG), the gathering of the TEI community on yearly members meetings, and the promotion of the standard in publications, on workshops, training courses, colloquia, and conferences. These activities are generally open to non-members as well.

By *TEI Guidelines* one may refer both to the markup language and tag set proposed by the TEI Consortium and to its documentation online or in print. Informally *TEI Guidelines* is often abbreviated to *TEI*. In this these tutorials *TEI Guidelines* is used as the general term for the encoding standard. The TEI Guidelines are widely used by libraries, museums, publishers, and individual scholars to present texts for online research, teaching, and preservation. Since the TEI is expressed in terms of the eXtensible Markup Language (XML) and since it provides procedures and mechanisms to adapt to one's own project needs, the TEI Guidelines define an open standard that is generally applicable to any text and purpose.

This introductory tutorial first introduces the concepts of text encoding and markup languages in the humanities and then introduces the TEI encoding principles. Next, this tutorial provides a brief historical survey of the TEI Guidelines and ends with a presentation of the Consortium's organisation.

2. Text Encoding in the Humanities

Since the earliest uses of computers and computational techniques in the humanities at the end of the 1940s, scholars, projects, and research groups had to look for systems that could provide representations of data which the computer could process. Computers, as Michael Sperberg-McQueen has reminded us, are binary machines that "can contain and operate on patterns of electronic charges, but they cannot contain numbers, which are abstract mathematical objects not electronic charges, nor texts, which are complex, abstract cultural and linguistic objects." (Sperberg-McQueen 1991, 34). This is clearly seen in the mechanics of early input devices such as punched cards where a hole at a certain coordinate actually meant a `1` or `0` (true or false) for the character or numerical represented by this coordinate, according to the specific character set of the computer used. Because different computers used different character sets with a different number of characters, texts first had to be transcribed into that character set. All characters, punctuation marks, diacritics, and significant changes of type style had to be encoded with an inadequate budget of characters. This resulted in a complex of "flags" for distinguishing upper-case and lower-case letters, for coding accented characters, the start of a new chapter, paragraph, sentence, or word. These "flags" were also used for adding analytical information to the text such as word classes, morphological, syntactic, and lexical information. Ideally, each project used its own set of conventions consistently throughout. Since this set of conventions was usually designed on the basis of an analysis of the textual material to be transcribed to machine readable text, another corpus of textual material would possibly need another set of conventions. The design of these sets of conventions was also heavily dependent on the nature and infrastructure of the project, such as the type of computers, software, and devices such as magnetic tapes of a certain kind that were available.

Although several projects were able to produce meaningful scholarly results with this internally consistent approach, the particular nature of each set of conventions or encoding scheme had lots of disadvantages. Texts prepared in such a proprietary scheme by one project could not readily be used by other projects; software developed for the analysis of such texts could hence not be used outside the project due to an incompatibility of encoding schemes and non-standardisation of hardware. However, with the increase of texts being prepared

in machine-readable format, the call for an economic use of resources increased as well. Already in 1967, Martin Kay argued in favour of a "standard code in which any text received from an outside source can be assumed to be" (Kay 1967, 171). This code would behave as an exchange format which allowed the users to use their own conventions at output and at input (Kay 1967, 172).

3. Markup Languages in the Humanities

3.1 Procedural and Descriptive Markup

When human beings read texts, they perceive both the information stored in the linguistic code of the text and the meta-information which is inferred from the appearance and interpretation of the text. By convention, italics are, for instance, used as a code signalling a title of a book, play, or movie; a foreign word or phrase; or emphatic use of the language. Through their cognitive abilities, readers usually have no problems selecting the most appropriate interpretation of an italic string of text. Computers, however, need to be informed about these issues in order to be able to process them. This can be done by way of a markup language that provides rules to formally separate information (the text in a document) from meta-information (information about the text in a document). Whereas markup languages in use in the typesetting community were mainly of a procedural nature —that is, they indicate procedures that a particular application should follow—(e.g., printing a string of text in italics), the humanities were also and mainly considered with descriptive markup that identifies the entity type of tokens (e.g., identifying that a string of text is a title of a book or a foreign word). Unlike procedural or presentational markup, descriptive markup establishes a one to one mapping between logical elements in the text and their markup. In order to achieve this, descriptive markup languages tend to formally separate information (the text in a document) from meta-information (information about the text in a document).

3.2 Early Attempts

Some sort of standardisation of markup for the encoding and analysis of literary texts was reached by the COCOA encoding scheme originally developed for the COCOA program in the 1960s and 1970s (Russel 1967), but used as an input standard by the Oxford Concordance Program (OCP) in the 1980s (Hockey 1980) and by the Textual Analysis Computing Tools (TACT) in the 1990s (Lancashire et al. 1996). For the transcription and encoding of classical Greek texts, the Beta-transcription/encoding system reached some level of standardised use (Berkowitz, Squitier, and Johnson 1986).

3.3 The Standard Generalized Markup Language (SGML)

The call for a markup language that could guarantee reusability, interchange, system- and software-independence, portability and collaboration in the humanities was answered by the publication of the Standard Generalized Markup Language (SGML) as an ISO standard in 1986 (ISO 8879:1986) (Goldfarb 1990). Based on IBM's Document Composition Facility Generalized Markup Language, SGML was developed mainly by Charles Goldfarb as a metalanguage for the description of markup schemes that satisfied at least seven requirements for an encoding standard (Barnard, Fraser, and Logan 1988, 28–29):

1. The requirement of comprehensiveness;
2. The requirement of simplicity;
3. The requirement that documents be processable by software of moderate complexity;
4. The requirement that the standard not be dependent on any particular characteristic set or text-entry devise;
5. The requirement that the standard not be geared to any particular analytic program or printing system;
6. The requirement that the standard should describe text in editable form;
7. The requirement that the standard allow the interchange of encoded texts across communication networks.

In order to achieve universal exchangeability and software and platform independence, SGML made use exclusively of the ASCII codes. As mentioned above, SGML is not a markup language itself, but a metalanguage by which one can create separate markup languages for separate purposes. This means that SGML defines the rules and procedures to specify the vocabulary and the syntax of a markup language in a formal Document Type Definition (DTD). Such a DTD is a formal description of, for instance, names for all elements, names and default values for their attributes, rules about how elements can nest and how often they can occur, and names for re-usable pieces of data (entities). The DTD enables full control, parsing, and validation of SGML encoded documents. By and large the most popular SGML DTD is the Hypertext Markup Language (HTML) developed for the exchange of graphical documents over the internet.

A markup scheme with all these qualities was exactly what the humanities were looking for in their quest for a descriptive encoding standard for the preparation and interchange of electronic texts for scholarly research. There was a strong consensus among the computing humanists that SGML offered a better foundation for research oriented text encoding than other such schemes (Barnard, Fraser, and Logan 1988, 26–31; Barnard et

al. 1988). From the beginning, however, SGML was also criticised for at least two problematic matters: SGML's hierarchical perspective on text, i.e., the representation of text as a hierarchical tree structure, and SGML's verbose markup system (Barnard et al. 1988). These two issues have since been central to the theoretical and educational debates on markup languages in the humanities.

3.4 The eXtensible Markup Language (XML)

The publication of the eXtensible Markup Language (XML) 1.0 as a W3C recommendation in 1998 (Bray, Paoli, and Sperberg-McQueen 1998) brought together the best features of SGML and HTML and soon achieved huge popularity. Among the power XML borrowed from SGML are the explicitness of descriptive markup, the expressive power of hierarchic models, the extensibility of markup languages, and the possibility to validate a document against a DTD. From HTML it borrowed simplicity and the possibility to work without a DTD. Technically speaking, XML is a subset of SGML and the recommendation was developed by a group of people with a long standing experience in SGML, many of whom were TEI members.

Because of its advantages and widespread popularity, XML became the metalanguage of choice for expressing the rules for descriptive text encoding in TEI.

4. XML: Ground Rules

4.1 Recommendation

XML is a metalanguage by which one can create separate markup languages for separate purposes. It is platform-, software-, and system-independent and no-one "owns" XML, although specific XML markup languages can be owned by their creators. Generally speaking, XML empowers the content provider and facilitates data integration, exchange, maintenance, and extraction. XML is currently the de facto standard on the World Wide Web partly because HTML (Hypertext Markup Language) was rephrased as an XML encoding language. XML is edited and managed by the W3C which also published the specification as a recommendation in 1998 (Bray, Paoli, and Sperberg-McQueen 1998).

The big selling point of XML is that it is text-based. This means that each XML encoding language is entirely built up in ASCII (American Standard Code for Information Interchange), or plain text, and can be created and edited using a simple text-editor like Notepad or its equivalents on other platforms. However, when you start

working with XML, you will soon find that it is better to edit XML documents using a professional XML editor. While plain text-editors don't know that you're writing TEI, XML editors will help you write error-free XML documents, validate your XML against a DTD or a schema, force you to stick to a valid XML structure, and enable you to perform transformations.

Since ASCII only provides for characters commonly found in the English language, different character encoding systems have been designed such as Isolat-1 (ISO-8859-1) for Western languages, and Unicode (UTF-8 and UTF-16). By using these character encoding systems, non-ASCII characters such as French *é*, *à*, *ç*, Norwegian *æ ø å*, or Hebrew ק can be used in XML documents. These systems rely on ASCII notation for the expression of these non-ASCII characters. The French *à*, for instance, is represented by the string **`agrave`** in Isolat-1 and by the number **`00E0`** in Unicode.

> REFERENCE
>
> See XML Resources, section 1 in the TBE Toolkit.

4.2 Components

Any XML encoding language consists of five components:

- Processing Instructions
- Elements
- Attributes
- Entity References
- (P)CDATA

For example, a simple two-paragraph document could be encoded as follows in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <!-- paragraphs go here -->
  <paragraph number="1">Paragraph one of <title>an XML example</title>.</paragraph>
  <paragraph number="2">Paragraph two of this example.</paragraph>
</document>
```

Example 1. A sample XML document.

4.2.1 XML Declaration

An XML document is introduced by the XML declaration:

```
<?xml version="1.0"  encoding="UTF-8"?>
```

Example 2. An XML declaration.

The question mark ? in the XML declaration signals that this is a processing instruction. The following bits state that what follows is XML which complies with version 1.0 of the recommendation and that the used character encoding is UTF-8 or Unicode. An XML declaration is optional, but can only appear as the first content of an XML file.

4.2.2 Elements and Attributes

The two-paragraph document above is an example of an XML document, representing both information and meta-information. Information (plain text) is contained in XML elements, delimited by start tags (e.g., `<document>`) and end tags (e.g., `</document>`). Additional information to these XML elements can be given in attributes, consisting of a name (e.g., `@number`) and a value (e.g., `"1"`). Attributes can only occur within the start tag of an element. XML comments are delimited by start markers (`<!--`) and end markers (`-->`). Everything inside comments is ignored by XML processing software: it is said to be "commented out."

4.2.3 Entity References

Entity references are predefined strings of data that a parser must resolve before parsing the XML document.[2] Entity references may be useful in a number of cases:

- representing character data which cannot easily be keyboarded or which is illegal in XML because some characters are reserved
- escaping reserved characters in XML

……

2 A parser is a piece of software that recognises a programming or an encoding language with the possible intent to process or interpret it. An XML parser can for instance be used to validate an XML document, transform it to another format, or process information from the document.

- providing “boilerplate text,” that is text which is or can be reused in new contexts or applications

An entity reference starts with an ampersand `&` and closes with a semicolon `;`. The entity name is the string between these two symbols. For instance, the entity reference for the “less than” sign `<` is `<`. The entity reference for the ampersand is `&`.

Not all computers necessarily support the Unicode encoding scheme XML works with. Portability of individual characters from the Unicode system, however, is supported by entity references that refer to their numeric or hexadecimal notation. For example, the character `ø` is represented within an XML document as the Unicode character with hexadecimal value `00F8` and decimal value `0248`. For exporting an XML document containing this character, it may be represented with the corresponding character reference `&#x00F8;` or `&#0248;` respectively, with the `x` indicating that what follows is a hexadecimal value. References of this type do not need to be predefined, since the underlying character encoding for XML is always the same.

For legibility purposes, however, it is also possible to refer to this character by use of a mnemonic name, such as **`oslash`**, provided that each such name is mapped to the required Unicode value by means of an ENTITY declaration.

```
<!ENTITY oslash "#x00F8">
```

Example 3. An entity declaration.

The ENTITY declaration uses a non-XML syntax inherited from SGML and starts with an opening delimiter `<` followed by an exclamation mark `!` signalling that this is a declaration. The keyword `ENTITY` names that an entity is being declared here. What follows next is the entity name - here the mnemonic name `&oslash;` - for which a declaration is given and the declaration itself inside quotation marks. In this example, it is the hexadecimal value of the character.

The same character can also be declared in the following ways:

```
<!ENTITY oslash "ø">
```

```
<!ENTITY oslash "#0248">
```

Example 4. Alternative (but equivalent) entity declarations.

Character entities must also be used in XML to escape the “less than” sign < and the ampersand & which are illegal because the XML parser could mistake them for markup.

```
<p>Gimme pepper &amp; salt!</p>
<p>A &lt; B</p>
```

Example 5. Escaping reserved XML charactes with character entities.

Entities are not only capable to refer to character declarations but can also refer to strings of text with an unlimited extent. This way, repetitive keying of repeated information can be avoided (aka string substitution), or standard expressions or formulae can be kept up to date. The first is useful, for instance, for the expansion of `&TBE;` to “TEI by Example” before the test is validated. This corresponding ENTITY declaration is as follows:

```
<!ENTITY TBE "TEI by Example">
```

Example 6. An entity declaration.

The second is used in contracts, books of laws, etc. in which updating would otherwise mean the complete re-keying of the same (extensive) string of text. For example, the expression: `This contract is concluded between &party1; and &party2; for the duration of 10 years starting from &date;` in legal texts can be updated simply by changing the value of the ENTITY declarations:

```
<!ENTITY party1 "Rev Knyff ">
<!ENTITY party2 "Lt Rosen">
<!ENTITY date "2010-01-01">
```

Example 7. More entity declarations.

The substitution of the entities by their values in the given example results in the following expression: `This contract is concluded between Rev Knyff and Lt Rosen for the duration of 10 years starting from 2007-01-01.`

3

ENTITY declarations are placed inside a DOCTYPE declaration which follows the XML declaration at the beginning of the XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rootelement [
  <!ENTITY party1 "Rev Knyff ">
  <!ENTITY party2 "Lt Rosen">
  <!ENTITY date "2010-01-01">
]>
```

Example 8. A DOCTYPE declaration.

The DOCTYPE declaration starts with the opening delimiter `<!` which is followed by the keyword `DOCTYPE`. Next comes the name of the root element of the document. In the case of a TEI document, this will be `TEI`. The entity references which must be interpreted by the XML processor are put inside square brackets. An XML parser encountering this DOCTYPE declaration will expand the entities with the values given in the ENTITY declaration before the document itself is validated.

4.2.4 (P)CDATA

All text in an XML document will normally be parsed by a parser. When an XML element is parsed, the text between the XML tags is also parsed. The parser does that because XML elements can nest as in the following example:

```
<paragraph number="1">Paragraph one of <title>an XML example</title>.</paragraph>
```

Example 9. Nesting XML elements.

.....

3 The term boilerplate text dates back to the early 1900s, referring to the thick, tough steel sheets used to build steam boilers. From the 1890s onwards, printing plates of text for widespread reproduction such as advertisements or syndicated columns were cast or stamped in steel (instead of the much softer and less durable lead alloys used otherwise) ready for the printing press and distributed to newspapers around the United States. They came to be known as “boilerplates.”

The XML parser will break this string up into an element `<paragraph>` with a subelements `<title>`. Text data that will be parsed by an XML parser is called parsed character data or PCDATA.

An XML document often contains data which need not be parsed by an XML parser. For instance, characters like < and & are illegal in XML elements because the parser will interpret them as the beginning of new elements or the beginning of an entity reference which will result in an error message. Therefore, these characters can be escaped by the use of the entity references `&lt;` and `&amp;`. When programming or scripting code (such as Javascript, which contains many occurrences of `&lt;` and `&amp;`) is included in an XML document, it should not be parsed by the XML parser. We can avoid this by treating it as *unparsed character data* or CDATA in the document:

```
<script><![CDATA[
  function matchwo(a,b) {
    if (a < b && a < 0) then {
      return 1;
    }
    else {
      return 0;
    }
  }
]]></script>
```

Example 10. Escaping a block of text as CDATA section.

A CDATA section starts with `<![CDATA[` and ends with `]]>`. Everything inside a CDATA section is treated as plain text by the parser.

4.3 Using XML

Depending on the nature of your XML documents and what you want to use them for, you will need different tools, ranging from freely available open source tools to expensive industrial software. In principle, the simplest plain text editor suffices to author or edit XML. In order to validate or transform XML, additional tools will be needed which often come included in dedicated XML editors: a validating parser, an XSLT processor, a tree-structure viewer etc. For publishing purposes, XML documents may be transformed to other XML formats, HTML or PDF—to name just a few of the possibilities—using XSLT and XSLFO scripts which are processed by an off-

the-shelf or custom-made XSL processor. These published documents can be viewed in generic web browsers or PDF viewers. XML documents can further be indexed, excerpted, questioned, and analysed with tools specifically designed for the job.

5. TEI: Ground Rules

5.1 Guidelines

The conclusions and the work of the TEI community are formulated as guidelines, rules, and recommendations rather than standards, because it is acknowledged that each scholar must have the freedom of expressing their own theory of text by encoding the features they think important in the text. A wide array of possible solutions to encoding matters is demonstrated in the TEI Guidelines which therefore should be considered a reference manual rather than a tutorial. Mastering the complete TEI encoding scheme implies a steep learning curve, but few projects require a complete knowledge of the TEI. Therefore, a manageable subset of the full TEI encoding scheme was published as TEI Lite, currently describing 140 elements (Burnard and Sperberg-McQueen 2006). Originally intended as an introduction and a didactic stepping stone to the full recommendations, TEI Lite has, since its publication in 1995, become one of the most popular TEI customisations and proves to meet the needs of 90% of the TEI community, 90% of the time.

5.2 TEI Modules

A significant part of the rules in the TEI Guidelines apply to the expression of descriptive and structural meta-information about the text. Yet, the TEI defines concepts to represent a much wider array of textual phenomena, amounting to a total of 580 elements and 265 attributes. These are organised into 21 modules, grouping related elements and attributes:

The TEI Infrastructure (`tei`)

Definition of common datatypes and modular class structures used to define the elements and attributes in the other modules.

The TEI Header (`header`)

Definition of the elements that make up the header section of TEI documents. Its major parts provide elements to encode detailed metadata about bibliographic aspects of electronic texts, their relationship with the source materials from which they may have been derived, non-bibliographic details, and a complete revision history.

Elements Available in All TEI Documents (`core`)

Definition of elements and attributes that may occur in any TEI text, of whatever genre. These elements cover textual phenomena like paragraphs, highlighting and quotation, editorial changes (marking of errors, regularisations, additions), data-like structures (names, addresses, dates, numbers, abbreviations), cross-reference mechanisms, lists, notes, graphical elements, bibliographic references, and passages of verse or drama.

Default Text Structure (`textstructure`)

Definition of elements and attributes that describe the structure of TEI texts, like front matter and title pages, text body, and back matter. These may contain further divisions, possibly introduced by headings, salutations, opening formulae, and/or concluded by closing formulae, closing salutations, trailing material and postscripts.

Characters, Glyphs, and Writing Modes (`gaiji`)

Definition of specific provisions for representing characters for which no standardised representation (such as defined by the Unicode Consortium) exists.

Verse (`verse`)

Definition of specific elements and attributes for dedicated analysis of verse materials, such as caesurae, metrical systems, rhyme schemes, and enjambments.

Performance Texts (`drama`)

Definition of specific elements and attributes for dedicated analysis of drama materials. These include provisions for encoding specific phenomena in front and back matter, like details about performances, prologues, epilogues, the dramatic setting, and cast lists. Other drama-specific structures include speeches and stage directions. For multimedia performances, elements for the description of screen contents, camera angles, captions, and sound are provided.

Transcriptions of Speech (`spoken`)

Definition of elements and attributes for (general purpose) transcription of different kinds of spoken material. These cover phenomena like utterances, pauses, non-lexical sounds, gestures, and shifts in vocal quality. Besides this, specific header elements for describing the vocal source of the transcription are provided.

Dictionaries (`dictionaries`)

Definition of elements and attributes for representing dictionaries, with provisions for unstructured and structured dictionary entries (possibly grouped). Dictionary entries may be structured with a number of specific elements indicating homonyms, sense, word form, grammatical information, definitions, citations, usage, and etymology.

Manuscript Description (`msdescription`)

Definition of specific header and structural elements and attributes for the encoding of manuscript sources. Header elements include provisions for detailed documentation of a manuscript's or manuscript part's identification, heading information, contents, physical description, history, and additional information. Dedicated text elements cover phenomena like catchwords, dimensions, heraldry, watermarks, and so on.

Representation of Primary Sources (`transcr`)

Definition of elements and attributes for detailed transcription of primary sources. Phenomena covered are facsimiles, more complex additions, deletions, substitutions and restorations, document hands, damage to the source material and illegibility of the text.

Critical Apparatus (`textcrit`)

Definition of elements and attributes for the representation of (different versions texts as) scholarly editions, listing all variation between the versions in a variant apparatus.

Names, Dates, People, and Places (`namesdates`)

Definition of elements and attributes for more detailed analysis of names of persons, organisations, and places, their referents (persons, organisations, and places) and aspects of temporal analyses.

Tables, Formulæ, Graphics and Notated Music (`figures`)

Definition of specific elements and attributes for detailed representation of graphical elements in texts, like tables, formulae, images, and notated music.

Language Corpora (`corpus`)

Definition of elements and attributes for the encoding of corpora of texts that have been collected according to specific criteria. Most of these elements apply to the documentation of these sampling criteria, and contextual information about the texts, participants, and their communicative setting.

Linking, Segmentation, and Alignment (`linking`)

Definition of elements and attributes for representing complex systems of cross-references between identified anchor places in TEI texts. Recommendations are given for either in-line or stand-off reference mechanisms.

Simple Analytic Mechanisms (`analysis`)

Definition of elements and attributes that allow the association of simple analyses and interpretations with text elements. Mechanisms for the representation of both generic and particularly linguistic analyses are discussed.

Feature Structures (`iso-fs`)

Definition of elements and attributes for constructing complex analytical frameworks that can be used to represent specific analyses in TEI texts.

Graphs, Networks, and Trees (`nets`)

Definition of elements and attributes for the analytical representation of schematic relationships between nodes in graphs and charts.

Certainty, Precision, and Responsibility (`certainty`)

Definition of elements for detailed attribution of certainty for the encoding in a TEI text, as well as the identification of the responsibility for these encodings.

Documentation Elements (`tagdocs`)

Definition of elements and attributes for the documentation of the encoding scheme used in TEI texts. This module provides means to define elements, attributes, element and attribute classes, either by changing existing definitions or by creating new ones.

Each of these modules and the use of the elements they define are discussed extensively in a dedicated chapter of the TEI Guidelines.

5.3 Using TEI

Among more technical ones, Steven DeRose pointed out substantial advantages of XML to the TEI community: by allowing for more flexible automatic parsing strategies and easy delivery of electronic documents with cheap ubiquitous tools such as web browsers, XML could spread the notion of descriptive markup to a wide audience that will thus be acquainted with the concepts articulated in the TEI Guidelines (DeRose 1999, 19).

In order to use TEI for the encoding of texts, users must make sure that their texts belong to the TEI namespace (**`http://www.tei-c.org/ns/1.0`**) and adhere to the requirements of the text model proposed by the TEI. In order to facilitate this conformance, it is possible (and strongly suggested) to associate TEI texts with formal representations of this text model. These formal "structural grammars" of a TEI compatible model of the text can be expressed in a number of ways, commonly referred to as a TEI schema. Technically, a TEI schema can be expressed in a variety of formal languages such as Document Type Definition (DTD), W3C XML Schema, or the RELAX NG schema language. It is important to notice that no such thing as "*the* TEI schema" exists. Rather, users are expected to select their desired TEI elements and attributes from the TEI modules, possibly with alterations or extensions where required. In this way, TEI offers a stable base with unambiguous means for the representation

of basic textual phenomena, while providing standardised mechanisms for user customisation for uncovered features. It is a particular feature of TEI that these abstract text models themselves can be expressed as TEI texts, using the documentation elements defined in the dedicated **tagdocs** module for documentation elements. A minimal TEI customisation file looks as follows:

```
<TEI xmlns="http://www.tei-c.org/ns/1.0" xml:lang="en">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>A TBE customisation</title>
        <author>The TBE Crew</author>
      </titleStmt>
      <publicationStmt>
        <p>for use by whoever wants it</p>
      </publicationStmt>
      <sourceDesc>
        <p>created on Thursday 24th July 2008 10:20:17 AM by the form at https://roma
        .tei-c.org/</p>
      </sourceDesc>
    </fileDesc>
  </teiHeader>
  <text>
    <front>
      <divGen type="toc"/>
    </front>
    <body>
      <p>My TEI Customisation starts with modules tei, core, header, and
       textstructure</p>
      <schemaSpec ident="TBEcustom" docLang="en" xml:lang="en">
        <moduleRef key="tei"/>
        <moduleRef key="header"/>
        <moduleRef key="core"/>
        <moduleRef key="textstructure"/>
      </schemaSpec>
    </body>
  </text>
</TEI>
```

Example 11. A minimal TEI customisation file.

Besides the common minimal TEI structure (<teiHeader> and <text>), a TEI customisation file has one specific element which defines the TEI schema (<schemaSpec>). A TEI schema must minimally include the modules which define the minimal TEI text structure: the **tei** module, the **core** module with all common TEI elements, the **header** module defining all teiHeader elements, and the **textstructure** module defining the elements representing the minimal structure of TEI texts.

In the vein of "Literary Programming," a TEI customisation file not only contains the formal declaration of TEI elements inside <schemaSpec>, but may also contain prose documentation of the TEI encoding scheme it defines. Consequently, TEI customisation files are commonly called "ODD files" (One Document Does it all), because they serve as a source for the derivation of:

- a formal TEI schema
- human-friendly documentation of the TEI encoding scheme

In order to accommodate the process of creating customised TEI schemas and prose documentation, the TEI has developed a dedicated piece of software called "Roma," available at https://roma.tei-c.org/. This is a dedicated ODD processor, offering an intuitive web-based interface for the creation and basic editing of ODD files, generation of according TEI schemas and prose documentation in a number of presentation formats.

A TEI schema, declaring all structural conditions and restraints for the elements and attributes in TEI texts can then be used to automatically validate actual TEI documents with an XML parser. Consider, for example, following fragments:

[A]

```
<TEI xmlns="http://www.tei-c.org/
ns/1.0">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>A sample TEI
         document</title>
      </titleStmt>
      <publicationStmt>
```

[B]

```
<TEI xmlns="http://www.tei-c.org/
ns/1.0">
  <text>
    <body>
      <p>This is a sample paragraph,
       illustrating a <orgName>T
      EI</orgName> document.</p>
    </body>
  </text>
```

```
      <publisher> KANTL</publisher>
      <pubPlace>Ghent</pubPlace>
      <date when="2009"/>
    </publicationStmt>
    <sourceDesc>
      <p>No source, born digital</p>
    </sourceDesc>
  </fileDesc>
 </teiHeader>
 <text>
  <body>
   <p>This is a sample
    paragraph, illustrating
    a <name type="organisation">T
   EI</name> document.</p>
  </body>
 </text>
</TEI>
```

Example 12. A sample (valid) TEI document.

```
</TEI>
```

Example 13. A sample (invalid) TEI document.

When validated against a TEI schema derived from the previous ODD file, file [A] will be recognised as a valid TEI document, while file [B] won't:

- The TEI prescribes that the <teiHeader> element *must* be present in a TEI document, and that it precedes the <text> part.
- The minimal set of TEI modules does not include the specialised <orgName> element. Although it is a TEI element, using it requires selection of the appropriate TEI module in the ODD file (in this case, the **namesdates** module, defining specialised elements for the identification of names, dates, people, and places in a text).

6. TEI: History

After the concise overview of the most recent version of TEI (P5) in section 5, this section explains the historical development of the TEI Guidelines.

6.1 Poughkeepsie Principles

Shortly after the publication of the SGML specification as an ISO Standard, a diverse group of 32 humanities computing scholars gathered at Vassar College in Poughkeepsie, New York in a two-day meeting (11 & 12 November 1987) called for by the Association for Computers and the Humanities (ACH), funded by the National Endowment for the Humanities (NEH), and convened by Nancy Ide and Michael Sperberg McQueen. The main topic of the meeting was the question how and whether an encoding standard for machine-readable texts intended for scholarly research should be developed. Amongst the delegates were representatives from the main European text archives and from important North American academic and commercial research centres. Contrary to the disappointing outcomes of other such meetings in San Diego in 1977 or in Pisa in 1980, this meeting did reach its goal with the formulation of and the agreement on the following set of methodological principles—the so called Poughkeepsie Principles—for the preparation of text encoding guidelines for literary, linguistic, and historical research (Burnard 1988, 132–133; Ide and Sperberg-McQueen 1988, E.6–4; Ide and Sperberg-McQueen 1995, 6):

1. The guidelines are intended to provide a standard format for data interchange in humanities research.
2. The guidelines are also intended to suggest principles for the encoding of texts in the same format.
3. The guidelines should
   - 3.1 define a recommended syntax for the format,
   - 3.2 define a metalanguage for the description of text-encoding schemes,
   - 3.3 describe the new format and representative existing schemes both in that metalanguage and in prose.
4. The guidelines should propose sets of coding conventions suited for various applications.
5. The guidelines should include a minimal set of conventions for encoding new texts in the format.
6. The guidelines are to be drafted by committees on
   - 6.1 text documentation
   - 6.2 text representation
   - 6.3 text interpretation and analysis
   - 6.4 metalanguage definition and description of existing and proposed schemes

   coordinated by a steering committee of representatives of the principal sponsoring organizations.
7. Compatibility with existing standards will be maintained as far as possible.

8. A number of large text archives have agreed in principle to support the guidelines in their function as an interchange format. We encourage funding agencies to support development of tools to facilitate this interchange.
9. Conversion of existing machine-readable texts to the new format involves the translation of their conventions into the syntax of the new format. No requirements will be made for the addition of information not already coded in the texts.

For the implementation of these principles the ACH was joined by the Association for Literary and Linguistic Computing (ALLC) and the Association for Computational Linguistics (ACL). Together they established the Text Encoding Initiative (TEI) whose mission it was to develop the "Poughkeepsie Principles" into workable text-encoding guidelines. The Text Encoding Initiative very soon came to adopt SGML, published a year before as ISO standard, as its framework. Initial funding was provided by the US National Endowment for the Humanities, Directorate General XIII of the Commission of the European Communities, the Canadian Social Science and Humanities Research Council, and the Andrew W. Mellon Foundation.

6.2 TEI P1 and TEI P2

From the Poughkeepsie Principles the TEI concluded that the TEI Guidelines should:

- Provide a standard format for data interchange;
- Provide guidance for encoding of texts in this format;
- Support the encoding of all kinds of features of all kinds of texts studied by researchers;
- Allow the rigorous definition and efficient processing of texts;
- Provide for user-defined extensions;
- Be application independent;
- Be simple, clear, and concrete;
- Be simple for researchers to use without specialized software.

A Steering Committee consisting of representatives of the ACH, the ACL, and the ALLC appointed Michael Sperberg-McQueen as editor-in-chief and Lou Burnard as European editor of the Guidelines.

The first public proposal for the TEI Guidelines was published in July 1990 under the title *Guidelines for the Encoding and Interchange of Machine-Readable Texts* with the TEI document number TEI P1 (for "Proposal 1"). This version was reprinted with minor changes and corrections, as version 1.1 in November 1990 (Sperberg-

McQueen and Burnard 1990). Further development of the TEI Guidelines was done by four Working Committees (Text Documentation, Text Representation, Text Analysis and Interpretation, Metalanguage and Syntax) and a number of specialist Working Groups amongst which groups on character sets, text criticism, hypertext and hypermedia, formulæ, tables, figures, and graphics, language corpora, manuscripts and codicology, verse, drama and performance texts, literary prose, linguistic description, spoken text, literary studies, historical studies, print dictionaries, machine lexica, and terminological data. The extensions and revisions resulting from this work, together with extensive public comment resulted in the drafting of a revised version, TEI P2, that was released chapter by chapter between March 1992 and the end of 1993 (Sperberg-McQueen and Burnard 1993) and that included substantial amounts of new material.

6.3 TEI P3

The following step was the publication of the TEI P3 *Guidelines for Electronic Text Encoding and Interchange* in 1994 (Sperberg-McQueen and Burnard 1994) that presented a further revision of all chapters published under the document number TEI P2, and the addition of new chapters. A final revised edition of these P3 Guidelines correcting several typographic and other errors, and introducing one new element was published in 1999 (Sperberg-McQueen and Burnard 1999). The publication of this 1,292 page documentation of the definitive guidelines defining some 439 elements marked the conclusion of the initial development work. With this work, the Poughkeepsie Guidelines were met by providing a framework for the encoding of texts in any natural language, of any date, in any literary genre or text type, without restriction on form or content and treating both continuous materials (“running text”) and discontinuous materials such as dictionaries and linguistic corpora.

6.4 TEI P4

Recognising the benefits for the TEI community, the P4 revision of the TEI Guidelines was published in 2002 by the newly formed TEI Consortium in order to provide equal support for XML and SGML applications using the TEI scheme (Sperberg-McQueen and Burnard). The chief objective of this revision was to implement proper XML support in the Guidelines, while ensuring that documents produced to earlier TEI specifications remained usable with the new version. The XML support was realised by the expression of the TEI Guidelines in XML and the specification of a TEI conformant XML DTD. The TEI P4 generated a set of DTD fragments that can be combined together to form either SGML or XML DTDs and thus achieved backwards compatibility with TEI P3 encoded texts. In other words, any document conforming to the TEI P3 SGML DTD was guaranteed to conform to the TEI

P4 XML version of it. This "double awareness" of the TEI P4 is the reason why this version was called an "XML-compatible edition" rather than an "XML edition." This was achieved by restricting the revisions needed to make the P4 version with its 441 elements to error correction only. During this process of revision, however, many possibilities for other, more fundamental changes have been identified. This led to the current TEI P5 version of the Guidelines.

6.5 TEI P5

In 2003 the TEI Consortium asked their membership to convene Special Interest Groups (SIGs) whose aim could be to advise revision of certain chapters of the Guidelines and suggest changes and improvements in view of the P5. With the establishment of the new TEI Council, which superintends the technical work of the TEI Consortium, it became possible to agree on an agenda to enhance and modify the Guidelines more fundamentally which resulted in a full revision of the Guidelines published as TEI P5 (TEI Consortium 2007). TEI P5 contains a full XML expression of the TEI Guidelines and introduces new elements, revises content models, and reorganises elements in a modular class system that facilitates flexible adaptations to users' needs. Contrary to its predecessor, TEI P5 does not offer backwards compatibility with previous versions of the TEI. The TEI Consortium has, however, maintained and corrected errors in the P4 Guidelines for 5 more years, up to the end of 2012. Since that date, the TEI Consortium has ceased official support for TEI P4, and deprecated it in favour of TEI P5. The P5 version is being updated continuously with regular releases: the most up-to-date version can be found at https://tei-c.org/guidelines/P5/.

7. TEI: Organisation

The TEI Consortium was established in 2000 as a not-for-profit membership organisation to sustain and develop the Text Encoding Initiative (TEI). The Consortium is supported by a number of host institutions. It is managed by a Board of Directors, and its technical work is overseen by an elected technical Council who take responsibility over the content of the TEI Guidelines.

The TEI charter outlines the consortium's goals and fundamental principles. Its goals are:

1. To establish and maintain a home for the Text Encoding Initiative (TEI) in the form of a permanent organisational structure.

2. To ensure the continued funding of TEI-C activities, for example: editorial maintenance and development of the TEI Guidelines and DTD, training and outreach activities, and services to members.
3. To create and maintain a governance structure for the TEI-C with broad representation of TEI user-communities.

The Consortium honours four fundamental principles:

1. The TEI Guidelines, other documentation, and DTD should be free to users;
2. Participation in TEI-C activities should be open (even to non-members) at all levels;
3. The TEI-C should be internationally and interdisciplinarily representative;
4. No role with respect to the TEI-C should be without term.

Involvement in the consortium is possible in two categories: individual membership, and institutional memebership. Only members have the right to vote on consortium issues and in elections to the Board and the Council, have access to a restricted website with pre-release drafts of Consortium working documents and technical reports, announcements and news, and a database of members, Sponsors, and Subscribers, with contact information, and benefit from discounts on training, consulting, and certification. The Consortium members meet annually at a Members' Meeting where current critical issues in text encoding are discussed, and members of the Council and members of the Board of Directors are elected.

8. Summary

Computers can only deal with explicit data. The function of markup is to represent textual material into digital form through the explicating act of text-encoding. Descriptive markup reveals what the encoder thinks to be implicit or hidden aspects of a text, and is thus an interpretive medium which often documents scholarly research next to structural information about the text. In order for this research to be exchangeable, analyzable, re-usable, and preservable, texts in the field of the humanities should be encoded according to a standard which defines a common vocabulary, grammar, and syntax, whilst leaving the implementation of the standard up to the encoder. A result of communal efforts among computing humanists, the Text Encoding Initiative documents such a standard in the TEI Guidelines. These guidelines are fully adaptable and customisable to one's specific project whilst enhancing this project's compatibility with other projects employing the TEI. Since over two decades,

the TEI has been used extensively in projects from different disciplines, fields, and subjects internationally. The ongoing engagements of a broad user community through the organization of the TEI Consortium consolidates the importance of the text encoding standard and informs its continuous development and maintenance.

9. Further Reading

REFERENCE

The TEI Consortium has its own journal, the *Journal of the Text Encoding Initiative* (jTEI), which provides a good overview of current research into text encoding with the TEI (investigating challenging use cases, proposals for new encoding solutions with TEI, etc.), and research done with TEI-encoded texts. The journal is freely accessible at https://journals.openedition.org/jtei. Notice that the TEI source files for jTEI articles can be downloaded as well.

Burnard, Lou. 2013. "The evolution of the text encoding initiative: From research project to research infrastructure." *Journal of the Text Encoding Initiative* 5. https://journals.openedition.org/jtei/811. 10.4000/jtei.811.

Burnard, Lou, Katherine O'Brien O'Keeffe, John Unsworth, (eds.). 2006. *Electronic Textual Editing*. New York: MLA. Preview: https://tei-c.org/Vault/ETE/Preview/ (Accessed October 2008).

Connell, Sarah, Julia Flanders, Nicole Infanta Keller, Elizabeth Polcha, and William Reed Quinn. 2017. "Learning from the Past: The Women Writers Project and Thirty Years of Humanities Text Encoding." *Magnificat Cultura i Literatura Medievals* 4: 1–19. https://ojs.uv.es/index.php/MCLM/article/view/10074. 10.7203/MCLM.4.10074.

Cummings, James. 2008. "The Text Encoding Initiative and the Study of Literature." In *A Companion to Digital Literary Studies*, edited by Ray Siemens, and Susan Schreibman. Oxford: Blackwell. 451–476. https://www.digitalhumanities.org/companion/view?docId=blackwell/9781405148641/9781405148641.xml&doc.view=print&chunk.id=ss1-6-6 (Accessed October 2008).

———. 2019. "A world of difference: Myths and misconceptions about the TEI." *Digital Scholarship in the Humanities* 34 (Supplement_1): i58-i79. 10.1093/llc/fqy071.

Flanders, Julia, and Fotis Jannidis (eds.). 2018. *The Shape of Data in Digital Humanities: Modeling Texts and Text-based Resources*. Abingdon, New York: Routledge.

Ide, Nancy, and Jean Véronis (eds.). 1995 *Text Encoding Initiative: Background and Context*. Dordrecht: Kluwer Academic Publishers. Reprinted from Computers and the Humanities 1995, 29.

Morgenstern, Anja, and Agnes Amminger. 2019. "Biography as Compilation: How to Encode Georg Nikolaus Nissen's *Biographie W. A. Mozart's* (1828) in TEI P5." *Journal of the Text Encoding Initiative* 11. https://journals.openedition.org/jtei/2725. 10.4000/jtei.2725.

Mylonas, Elli, and Allen Renear (eds.). 1999. *Special Issue: Selected Papers from TEI 10: Celebrating the Tenth Anniversary of the Text Encoding Initiative*. *Computers and the Humanities* 33 (1–2).

Pierazzo, Elena. 2015. "Textual scholarship and text encoding." In *A New Companion to Digital Humanities*, edited by Susan Schreibman, Ray Siemens, and John Unsworth: 307–321. 10.1002/9781118680605.ch21.

Romary, Laurent. 2016. "The Text Encoding Initiative: 30 years of accumulated wisdom and its potential for a bright future." Paper presented at the conference Language Technologies & Digital Humanities 2016, Sep 2016, Ljubljana, Slovenia. https://hal.inria.fr/hal-01374597

Schreibman, Susan, and Sebastian Rahtz (eds.). 2009. *Special Issue: TEI at 20. LLC. The Journal of Digital Scholarship in the Humanities* 24.

Terras, Melissa, Ron Van den Branden, Edward Vanhoutte. 2009. "Teaching TEI: The Need for TEI by Example." *Literary and Linguistic Computing* 24 (3): 297–306. 10.1093/llc/fqp018.

Unsworth, John. 2011. "Computational work with very large text collections. Interoperability, Sustainability, and the TEI." *Journal of the Text Encoding Initiative* 1. https://journals.openedition.org/jtei/215. 10.4000/jtei.215.

10. What's Next?

You have reached the end of this tutorial module providing an introduction to the TEI and text encoding for the humanities. You can now either

- proceed with other TEI by Example modules
- take an interactive test. This comes in the form of a set of multiple choice questions, each providing a number of possible answers. Throughout the quiz, your score is recorded and feedback is offered about right and wrong choices. Can you score 100%. Test it here.

BIBLIOGRAPHY

Barnard, David T., Cheryl A. Fraser, and George M. Logan. 1988. "Generalized Markup for Literary Texts." *Literary and Linguistic Computing* 3 (1): 26–31. 10.1093/llc/3.1.26.

Barnard, David T., Ron Hayter, Maria Karababa, George M. Logan, and John McFadden 1988. "SGML-Based Markup for Literary Texts: Two Problems and Some Solutions." *Computers and the Humanities* 22 (4): 265–276.

Berkowitz, Luci, Karl A. Squitier, and William H. A. Johnson. 1986. *Thesaurus Linguae Graecae, Canon of Greek Authors and Works.* New York/Oxford: Oxford University Press.

Bray, Tim, Jean Paoli, and C. M. Sperberg-McQueen. *Extensible Markup Language (XML) 1.0.* W3C Recommendation 10-February-1998. https://www.w3.org/TR/1998/REC-xml-19980210 (accessed September 2008).

Burnard, Lou 1988. "Report of Workshop on Text Encoding Guidelines." *Literary and Linguistic Computing* 3 (2): 131–133. 10.1093/llc/3.2.131.

Burnard, Lou, and C. M. Sperberg-McQueen. 2006. "TEI Lite: Encoding for Interchange: an introduction to the TEI Revised for TEI P5 release." February 2006 https://tei-c.org/release/doc/tei-p5-exemplars/html/tei_lite.doc.html.

DeRose, Steven J. 1999. "XML and the TEI." *Computers and the Humanities* 33 (1–2): 11–30.

Goldfarb, Charles F. 1990. *The SGML Handbook*. Oxford: Clarendon Press.

Hockey, Susan 1980. *Oxford Concordance Program Users' Manual*. Oxford: Oxford University Computing Service.

Ide, Nancy M., and C. M. Sperberg-McQueen. 1988. "Development of a Standard for Encoding Literary and Linguistic Materials." In *Cologne Computer Conference 1988. Uses of the Computer in the Humanities and Social Sciences. Volume of Abstracts.* Cologne, Germany, Sept 7–10 1988, p. E.6-3-4.

———. 1995. "The TEI: History, Goals, and Future." *Computers and the Humanities* 29 (1): 5–15.

Kay, Martin 1967. "Standards for Encoding Data in a Natural Language." *Computers and the Humanities*, 1 (5): 170–177.

Lancashire, Ian, John Bradley, Willard McCarty, Michael Stairs, and Terence Russon Woolridge. 1996 *Using TACT with Electronic Texts*. New York: Modern Language Association of America.

Russel, D. B. 1967. *COCOA: A Word Count and Concordance Generator for Atlas*. Chilton: Atlas Computer Laboratory.

Sperberg-McQueen, C. M. 1991. "Text in the Electronic Age: Textual Study and Text Encoding with examples from Medieval Texts." *Literary and Linguistic Computing* 6 (1): 34–46. 10.1093/llc/6.1.34.

Sperberg-McQueen, C. M., and Lou Burnard (eds.). 1990. *TEI P1: Guidelines for the Encoding and Interchange of Machine Readable Texts*. Chicago/Oxford: ACH-ALLC-ACL Text Encoding Initiative. https://tei-c.org/Vault/Vault-GL.html (accessed October 2008).

———. 1993. *TEI P2 Guidelines for the Encoding and Interchange of Machine Readable Texts* Draft P2 (published serially 1992–1993); Draft Version 2 of April 1993: 19 chapters. https://tei-c.org/Vault/Vault-GL.html (accessed October 2008).

———. 1994. *Guidelines for Electronic Text Encoding and Interchange. TEI P3.* Oxford, Providence, Charlottesville, Bergen: Text Encoding Initiative.

———. 1999. *Guidelines for Electronic Text Encoding and Interchange. TEI P3. Revised reprint.* Oxford, Providence, Charlottesville, Bergen: Text Encoding Initiative.

———. 2002. *TEI P4: Guidelines for Electronic Text Encoding and Interchange. XML-compatible edition.* XML conversion by Syd Bauman, Lou Burnard, Steven DeRose, and Sebastian Rahtz. Oxford, Providence, Charlottesville, Bergen: Text Encoding Initiative Consortium. https://tei-c.org/Vault/P4/doc/html/ (accessed October 2008).

TEI Consortium. 2007. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Oxford, Providence, Charlottesville, Nancy: TEI Consortium. https://tei-c.org/Vault/P5/1.0.0/doc/tei-p5-doc/en/html/ (accessed October 2008).