4 Languages and Character Sets

Since the first publication of this chapter, many of its recommendations have been rendered obsolete or obsolescent by the development of ISO/IEC 10646 and the adoption of Unicode as the underlying character set for all XML documents. The chapter has undergone considerable revision to reflect these changes, but further substantial change is likely in the next release of these Guidelines.

Computer systems vary greatly in the sets of characters they make available for use in electronic documents; this variety enables users with widely different needs to find computer systems suitable to their purposes, but it also complicates the interchange of documents among systems; hence the need for a chapter on this topic in these Guidelines.

In the absence of any generally agreed universal character set, creators of electronic documents have constantly faced such problems as:

1. selecting which character set to use in creating, processing, or storing the electronic text
2. preparing documents for interchange so that the characters within them are not corrupted in transit
3. encoding characters which are not provided by the character set available on the computer system in use at one or other end of the interchange
4. indicating shifts from one character set to another, e.g. from the Latin alphabet to Greek and back, or to a special symbol character set and back

Over the last two decades of the twentieth century, it became increasingly clear that a unified character set, able to accomodate all languages and scripts used in the world, would be not only desirable, but also feasible. With the formation of the Unicode Consortium and the development of the Unicode Standard, later synchronized with the work of the International Organization for Standardization (ISO) which lead to the definition of ISO 10646, it became possible to speak of a *universal character set*, and thus to solve all of the problems listed above for the vast majority of users.

With the availability of a Universal Character Set, users no longer need to select different character sets for different languages or applications; when documents are prepared for interchange, the usage of the universal character set ensures that characters will not be corrupted in transit, and that the same characters will be available at either end of the interchange; and lastly, there is no need to shift between different character sets within a single document.

For some of the materials which users of these Guidelines are likely to want to process (notably, ancient texts using little-known writing systems) the simplest ways of using the Universal Character Set may not be applicable. Nevertheless, it provides a more reliable and comprehensive mechanism for processing materials using such writing systems than any hitherto available.

In this chapter we first describe informally the model of character encoding which underlies modern computing systems. We next describe how present-day encoding systems can benefit from the advent of Unicode, particularly in the XML context, and make recommendations for its use by the TEI community. We also discuss issues relating to continued use of SGML legacy data.

4.1 A simple character encoding model

4.1.1 Some definitions

The term *character* has several meanings which it is necessary to keep distinct for a proper understanding of the issues which arise in representing texts in digital form. At the risk of insulting the intelligence of the expert reader, we will attempt to disentangle some of those meanings here.[45]

We use the term *abstract character* to refer to the atomic component of some writing system, independent of how it is realized in some written form, and also of how it is stored in some digital form. For example, we might use the letter A to stand for the same abstract character (capital letter roman A), whatever font

[45] This informal introduction is derived partly from an excellent tutorial on character code issues written by Jukka Korpela, available from `http://www.cs.tut.fi/~jkorpela/chars.html`, which includes a useful list of pointers to other introductory tutorial material. Definitive information on the topics discussed here is available from the Unicode Consortium's website at `http://www.unicode.org/`.

is used to render it, and whatever pattern of bits is used to represent it in digital storage. A character is an inherently abstract notion: it corresponds with 'the smallest unit that carries semantic value' within some writing system.

A given computer system will usually be configured to support a fixed number of abstract characters, which we call its *character repertoire*.

We use the term *glyph* to refer to the particular written form used for an abstract character when it is rendered on screen or paper, and the term *font* for a particular set of glyphs. The same character may be represented by many different glyphs; less obviously, the same glyph, may in certain circumstances correspond with different abstract characters, or be used with different interpretations, as when, for example, the Greek capital letter omega is also used to represent the unit of electrical resistance (ohm).

When dealing with historical material, it may be a matter of some debate as to whether some glyph should be regarded as a variant of the same abstract character or as a different character. For example, in early printed texts the lowercase roman letter S may appear in an elongated form. Should this be treated as a distinct abstract character (itself with variant glyphs) or simply as a variant glyph for the abstract character represented also by the non-elongated form? Some relevant questions to ask are:

- Does the elongated form of the s ever contrast with the non-elongated form in the same document with a different meaning? If so, it should be handled as a separate character.
- Could the two forms be interchanged without affecting the meaning of a document? If so, they should be regarded as variant glyphs.
- Does one of the forms appear in a predictable context (for example only in a particular position in a word)? If so, it might be a variant glyph, rather than a distinct character.

Because historical materials are often fragmentary and idiosyncratic in their appearance, the decision as to whether a sign is a character or variant may be a matter of debate. It is also a decision which changes over time: it might be argued, for example, that the letters U and V, which are now regarded as distinct, were at one time variant glyphs for the same abstract character in the Latin writing system. However such issues are handled, they should be carried through consistently and adequately documented.

We use the term *coded character set* (strictly) to mean the set of numeric values associated with a given character repertoire when it is represented in digital form. For example, in some character code, the abstract character A might be represented by the number 31, B by the number 32, and so on. Each of these mappings (from abstract character to number) is sometimes called a *code point*. A number of other phrases are sometimes used in place of 'coded character set', including *character code* or *character set*, and the same phrase is also often used as a synonym for both font and repertoire, as we have defined them here. Our usage follows common practice within both ISO and W3C.[46]

Note also that a given abstract character may be represented by more than one code point, or by a sequence of code points. For example, the single abstract character ä corresponds with code point 244 in the Universal Character Set, but also with the sequence of code points 31 and 104, which stand for the letter a and the diaresis symbol respectively.[47]

Finally, we use the term *encoded character* to mean simply the numerical value associated with that abstract character in a given coded character set.[48]

The character encoding (i.e. the representation used for its encoded characters) applicable to an XML document is stated in its encoding declaration (2.10.1 *SGML and XML declarations*) and is UTF8 or UTF16 by default. These Guidelines do not recommend usage of other character encodings for XML documents. The encoding applicable to an SGML document is defined (along with other matters) in the SGML declaration prefixed to it and is almost entirely arbitrary.

[46] The terminology used is summarized in ISO/IEC 2022 (1994)

[47] Abstract characters such as the diaresis or umlaut symbol, which are combined with others to form new characters are technically known as *composing* or *combining* characters.

[48] The way that a numerical value is actually represented as a sequence of bits in computer storage may vary: for example, the number 31 might be represented using 16 or 32 or even 64 bits, with different left-to-right ordering, or with different byte-groupings, on different hardware.

4.1.2 Characters and glyphs

Both the Unicode standard and ISO/IEC 10646 give fundamentally similar definitions for the terms *character* and *glyph* along the lines we have informally introduced above. The ISO definitions are as follows:

character A member of a set of elements used for the organisation, control, or representation of data. (ISO/IEC 10646-1: 1993, 31)

glyph A recognizable abstract graphic symbol which is independent of any specific design. (ISO/IEC 9541-1: 1991, 3.5)

Here are the Unicode definitions:

character (1) The smallest component of written language that has semantic value; refers to the abstract meaning and/or shape, rather than a specific shape (see also *glyph*), though in code tables some form of visual representation is essential for the reader's understanding. (2) Synonym for abstract character. (See Definition D3 in Section 3.3, *Characters and Coded Representations*) (3) The basic unit of encoding for the Unicode character encoding. (4) The English name for the ideographic written elements of Chinese origin.

glyph (1) An abstract form that represents one or more glyph images. (2) A synonym for glyph image. In displaying Unicode character data, one or more glyphs may be selected to depict a particular character. These glyphs are selected by a rendering engine during composition and layout processing. (See also *character*.)

glyph image The actual, concrete image of a glyph representation having been rasterized or otherwise imaged onto some display surface.

Although representative glyphs do appear in Unicode character tables as an aid to the reader, it is important to note that these are not normative; the Unicode Standard defines characters, as defined above, and not glyphs.

These may be compared with the following related, partially overlapping definitions, used in the field of linguistic theory (See for example R. R. K. Hartmann and F.C. Stork: *Dictionary of language and linguistics*, Applied Science Publishers Ltd., London, 1976)

grapheme A minimally distinctive unit of a particular writing system. The different variants, e.g., the cursive and printed shapes of letters M, m, cursivated m, M, etc. in an alphabetic writings system are all allographs of the grapheme /m/.

allograph One of a group of variants of a grapheme or written sign in a particular writing system. It usually refers to different shapes of letters and punctuation marks, e.g., lower case, capital, cursive, printed, strokes.

While 'glyph' and 'allograph' seem almost synonymous, it should be noted that 'grapheme' is defined with reference to a particular writing system, whereas an 'abstract character' is defined independently of any specific writing system.

The distinction between characters and glyphs is crucial in preparing an encoded text. Users of such texts expect systems to recognize different glyphs as representing the same character when (for example) performing text retrieval or text searching; at the same time, they expect characters to be rendered using appropriate glyphs. When encoding a pre-existing text, the encoder must therefore determine whether a particular letter or symbol is a character or a glyphic variant of one. A coherent model of the relationship between characters and glyphs has been developed within the Unicode Consortium and the ISO working group SC2[49] and will form the base for much future standards work.

The model makes explicit the distinction between two different properties of the components of written language:

- their content, i.e. its meaning and phonetic value (represented by a character)

[49] See *ISO/IEC 15285:1998 Information technology — An operational model for characters and glyphs*.

- their graphical appearance (represented by a glyph)

When searching for information, a system generally operates on the content aspects of characters, usually with little attention paid to their appearance of characters. A layout or formatting procedure on the other hand, has little to do with the content, but needs to be concerned with the exact appearance of characters. Of course, many operations require attention to both kinds of feature (hyphenation for example), but in general the kind of text encoding described in these Guidelines tends to focus on content rather than appearance (see further 6.3 *Highlighting and Quotation*).

When the purpose of an encoding is to represent information about which glyphs were used in some instance of the document being treated, one might choose to do so at either or both of two levels:

- on the level of character encoding, e.g. with appropriate Unicode code points.
- on the markup level, with appropriate elements and/or attributes.

It should be noted that using 'appropriate Unicode code points' to represent glyph information requires that such choices be documented in the TEI Header or WSD. Such documentation does not guarantee proper display of the desired glyph but at least makes the intention of the encoder discoverable.

At present, neither the Unicode Standard nor these Guidelines offer detailed specifications for the encoding of glyph variations. Some discussion of related matters is given in 18 *Transcription of Primary Sources*, and the writing system declaration (25 *Writing System Declaration*) offers some features for the definition of variant glyphs, but further work is needed in both these areas before detailed recommendations can be made.

4.1.3 Characters and their encoding

Over the years, many different ways of encoding abstract characters have been proposed by national standard bodies and vendors of information processing systems, often derived from the sometimes limited character repertoires available on specific kinds of hardware.[50]

Because of this variety, at the time these Guidelines were first published (1994) no single character set could plausibly be recommended for use in TEI-encoded documents. It was felt at that time that users would have to use whatever character sets were available to them, subject to the character set restrictions imposed by the SGML declaration. For texts subject to 'blind' interchange (that is, interchange between parties who do not or cannot make explicit agreements over the character set to be used in interchange), users would have to rely on a small subset of the available character repertoires which could be reliably transported across networks.[51] At that time, also, the character encoding used by an SGML document could be redefined by the SGML declaration: this was necessary, since it would otherwise have been impossible to interchange SGML documents using different encodings.

With the advent of XML, and the definition of Unicode, the job of those wishing to encode arbitrary character repertoires becomes very much simpler. The Unicode standard[52] defines the Universal Character Set. Its primary goal is to provide an unambiguous encoding for the content of plain text, ultimately covering all languages. Currently in its third major version, Unicode provides coverage for most of the world's writing systems. It also contains additional characters for interoperability with older character

[50] For a historical survey, see Charles E. Mackenzie *Coded character sets: history and development* (Addison-Wesley, 1980); see also Tom Jennings' *Annotated history of character codes* at `http://www.wps.com/texts/codes/`.

[51] This subset comprised only the following characters taken from the international reference version (IRV) of ISO 646

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
" % & ' ( ) * + , - . / : ; < = > ? _
```

The 1994 edition of these Guidelines recommended that (for interchange purposes) other characters should be represented with entity references, or with transliterations, documented in an accompanying Writing System Declaration.

[52] This section gives only a very short overview of those parts of the Unicode standard relevant to the current discussion. For further and more precise information, the reader should consult the Unicode Consortium website or the book *The Unicode Standard Version 3.0*. This was the current major edition available in print at the time of writing, and is used in references. Two minor revisions have been made to this, which are documented at the web site of the Unicode Consortium; the version number of the online edition is 3.1.1.

encodings, and characters with control-like functions included primarily for reasons of providing unambiguous interpretation of plain text. Unicode provides specifications for use of all of these characters.

Users of these Guidelines are strongly recommended to make themselves familiar with the general principles of Unicode, as spelled out in *The Unicode Standard Version 3.0*, Chapter 2. Additionally, there is an excellent document co-published by the World Wide Web Consortium and the Unicode Consortium, *Unicode in XML and other Markup Languages*.[53] which gives some general considerations, and specific usage suggestions.

We have stated several times above (2.7.3 *Character references*) that all XML documents use the same character encoding: this is the Universal Character Set defined by ISO/IEC 10646, which is effectively the same as the character encoding defined by the Unicode consortium.[54] Strictly speaking, these are both character encodings and character repertoires, since each of them defines both a set of abstract characters and the code points corresponding with them.

It should be stressed again that Unicode represents 'abstract characters' independent of specific glyph forms: the glyph examples given in Unicode code charts are not normative, being chosen simply to indicate which character is intended. There is no guarantee that a given character encoded with a given Unicode codepoint will be rendered in a similar way on different computer systems, because information about which glyph from which glyph collection is to be used is simply not encoded, and is therefore not available for any rendering process.

Although it may be tempting to specify a glyph guided by its particular appearance in a font, users should be aware that font repertoires may vary; what one system displays may not necessarily appear the same to someone using a different font. Hence, just relying on character encoding and the font may not be sufficient and a comment in the WSD or TEI header should be included, specifying the exact nature of the intended glyph.

The code space for Unicode characters allows for around one million characters. These characters are organized as 17 'planes', each of which holds up to 65 536 codepoints. Only the first of these planes, the 'Basic Multilingual Plane' (BMP) can be addressed (or represented) using a single 16 bit integer; for all other planes two 16 bit integers are required. Unicode 3.1 has assigned codepoints beyond the BMP for the first time, but most characters can be addressed using just the BMP.

4.1.4 Character semantics

In addition to the Universal Character Set itself, the Unicode Consortium maintains a database of additional character semantics. This includes names for each character codepoint and normative properties for it. This database is an important reference in determining which Unicode codepoint to use to encode a certain character. In addition to the printed documentation and lists made available by the Unicode consortium, it may also be accessed by a number of search systems over the web (e.g. `http://www.eki.ee/letter/`). Examples of character properties included in the database include *case*, *numeric value*, *directionality*, and its status as a 'compatibility character'.[55]

The existence of compatability and composed characters means that, with the best will in the world, there will still remain a number of characters that have more than one encoding in Unicode. In addition to characters composed from a base character and some diacritical marks, Unicode also contains quite a number of 'precomposed characters'. Most of these were already encoded in earlier standards that served as sources for Unicode; in principle, no further precomposed characters are to be added. A sequence of a base character and one or more diacritical marks is supposed to be equivalent to the corresponding precomposed character; yet both are present. In the same way, different allographic forms of the same underlying ideograph may sometimes be regarded as distinct Unicode characters.

[53] Written by Martin Dürst and Asmus Freytag, this document is available from `http://www.unicode.org/unicode/reports/tr20/tr20-5.html` .

[54] The character encoding standards defined by the ISO as 'ISO/IEC 10646-1:2000' and the Unicode Consortium as the 'Unicode Standard' are identical for most practical purposes: in all instances where we refer to either of the two standards below, the other is also meant to be included.

[55] A *compatability character* is a character included for compatibility with existing standards, even though it can be represented by other characters or combinations of characters already encoded in the Unicode Standard. See *Unicode in XML and other Markup Languages* , Section 4 for additional information.

It is important to treat such variation in a consistent and normalized way. Where, for a particular project, these multiple character representations are regarded as equivalent, data integrity requires that the project standardize on one form and document its decision. The Unicode Consortium provides four standard normalization forms, [56] of which the Normalization Form C (NFC) seems to be most appropriate for text encoding projects. The World Wide Web Consortium has produced a document entitled *Character Model for the World Wide Web 1.0*, [57] which among other things outlines some principles of normalization. In general, normalization to the shortest possible Unicode encoding is recommended.

4.1.5 Characters from the Private Usage Area

Although Unicode has already assigned more than 94 000 characters to unique codepoints, there is always the possibility that characters needed are not defined in Unicode. Some of these may be presentation forms or alternate writing styles of East Asian characters that do not qualify to be included in Unicode. For such characters, Unicode provides a 'Private Usage Area', which is reserved for use by vendors, private groups, and individuals. There are 6400 codepoints in this area in the BMP and 131 068 in other planes.

By definition, the codepoints in this range are of use only internally. Their use in TEI documents intended for interchange is therefore *strongly discouraged*. Where encoders need to interchange non-Unicode characters, they should do so by other mechanisms, for example by named character entity references, supported by appropriate documentation, for example in a WSD.

For local processing, on the other hand, use of characters from this area might prove convenient, since, if the corresponding font resources are available, users can see the characters more easily on their screens and analytical software might not be able to process entity references in the same way as characters. In any case, before preparing a TEI document for interchange, all occurrences of characters from the Private Usage Area should be removed.

As a concrete example, supposing that we wished to render the elongated s (or any other glyph variant) in a distinct way in our local processing environment, we might conveniently assign some codepoint from the PUA for this purpose (say, U+E000). This would then make it possible to create a font that displays the desired character at this codepoint. However, since this assignment would only be valid on the local site, any texts containing such codepoints would not be suitable for interchange until they have been re-encoded, for example by substituting a character entity reference (such as `&long-s;`) for each occurrence of U+E000. When received at some other site, the character entity reference could again be resolved to a code point from the PUA. If (as is possible) the receiving site has already assigned some other use for the codepoint U+E000, it can simply choose some other hitherto unused codepoint (say, U+E080) to represent the same variant glyph. (On character entity references, see 4.2.1 *Character input and entity references* below).

4.2 Entry and display of characters

So far we have largely been concerned with the advantages of using a single character encoding for the storage and representation of data. At the time of writing (2002), Unicode-aware software is becoming increasingly common, as the ideas of Unicode find ready acceptance in the networked world. Nevertheless, there will continue for some time to be a need for guidance on how to input and display Unicode data using non-Unicode-aware systems, just as there will always need to be a way of entering and displaying non-Unicode characters. The methods developed to cope with the pre-Unicode world of multiple character sets can helpfully be re-applied to address these problems.

4.2.1 Character input and entity references

Data characters can be included in an XML document directly, by typing them in from a keyboard, or indirectly by representing them by means of entity references. When characters are typed in directly, the software used can be configured in a variety of ways to simplify the task of entering characters not immediately visible on the keyboard (for example, by using special keyboard shortcuts, *escape sequences*,

[56] See Unicode Technical Report 15 at `http://www.unicode.org/unicode/reports/tr15/`.
[57] Available at `http://www.w3.org/TR/charmod`: see section 4.2 Definitions for W3C Text Normalization.

onscreen virtual keyboards, etc.); these are not described in any detail here, as they are so environment-dependent. For example, if the character Ä is not directly available from the keyboard, and if one is using a machine running Windows, one might enter it by holding down the ALT key and typing the digits 0196 on the numeric keyboard. Alternatively, on a system such as Gnu Emacs, one might define a sequence of keystrokes such as `A"` to have the same effect.

When characters are represented using *entity references* (described in more detail in section 2.7 *Entities*), the reference may be given as a numeric entity reference, using either decimal or hexadecimal notation, or it may be given as a standardized name. For example, the character Ä might be represented by any of the three following entity references `&#196; &#x00C4; &Auml;`. The first two represent the character required by means of its code point value (196 in decimal, C4 in hexadecimal) in the Unicode character code: they are thus entirely self-sufficient, and can be processed directly by any Unicode-aware system; they are not however as attractive for human beings, who generally find names more memorable than numbers. In the third case above, the name `Auml` is taken from a widely-used *entity set* called ISO Latin 1. An entity set is simply a list of entity declarations in which parts of some character repertoire are defined as entities with memorable names or mnemonics, with values taken from an appropriate character encoding. For example, the XML version of the iso-lat1 entity set includes the following declaration `<!ENTITY Auml "&#x00C4;">`. [58]

If entering the following text,

> Trotz dieser langen Tradition sekundäranalytischer Ansätze wird man die Bilanz tatsächlich durchgeführter sekundäranalytischer Arbeiten aber in wesentlichen Punkten als unbefriedigend empfinden müssen.

in a system which does not allow for direct representation of a-umlaut or u-umlaut, one could transcribe this sentence thus:

```
Trotz dieser langen Tradition sekund&auml;ranalytischer
 Ans&auml;tze wird man die Bilanz tats&auml;chlich
 durchgef&uuml;hrter sekund&auml;ranalytischer Arbeiten
 aber in wesentlichen Punkten als unbefriedigend
 empfinden m&uuml;ssen.
```

Before an entity can be referred to, it must be declared. Standard public entity names can be declared en masse, by including within the DTD subset of the document a reference to the standard public entity which declares them. The German document quoted above, for example, might have the following lines, or their equivalent, in its DTD subset:

```
<!ENTITY % ISOLat1
        PUBLIC "ISO 8879-1986//ENTITIES Added Latin 1//EN"
        "http://www.tei-c.org/XML_Entities/iso-lat1.ent">
 %ISOLat1;
```

Such mechanisms are obviously unnecessary for the treatment of characters such as a-umlaut or u-umlaut which are readily available in Unicode. However, they may also be used for the representation of other characters which are not yet standardized but which an encoder wishes to distinguish. Such characters can be represented within a document by any arbitrary entity name, which the encoder can then associate with different expansions depending on the software system or application involved.

For example, in transcribing a manuscript, it might be desirable to distinguish among three distinct forms of the letter ‘r’. In the transcript, each of these forms will be encoded by an entity reference, for example: `&r1;`, `&r2;`, and `&r3;`. Entity declarations must then be provided within the DTD subset of the document to define these entities and specify a substitute string.

One possible set of declarations would be as follows:

```
<!ENTITY r  'r[1]'>
<!-- most common form of 'r'  -->
 <!ENTITY r2 'r[2]'>
<!-- secondary form of 'r'    -->
```

[58] The most widely used such entity set is to be found in Annex D to ISO 8879; it is also reproduced or summarized in most SGML textbooks, notably Charles F. Goldfarb, *The SGML Handbook* (Oxford: Clarendon Press, 1990). Entity sets appropriate for use with both SGML and XML are available from the TEI website.

```
 <!ENTITY r3 'r[3]'>
<!-- third form of 'r'        -->
```

The expansions shown above will simply flag each occurrence with a number in brackets to indicate which form of 'r' appears in the manuscript.

More realistically, we may be able to associate each of the variant forms found with some predefined Unicode character. For example, assuming that r1 represents an r with a dot above it, r2 represents an r with a tail, and r3 represents an r with a fish-hook, we could simply use the appropriate Unicode values as follows:

```
<!ENTITY r1  '&#x1e59;'>    <!-- r with dot above  -->
 <!ENTITY r2 '&#x027d;'>     <!-- r with tail       -->
 <!ENTITY r3 '&#x027e;'>     <!-- r with fish-hook  -->
```

Obviously, this will only have the desired effect if a font containing the required glyphs is available when the document is rendered. If rendering the glyphs requires some special processing action, it may be preferable to use a processing instruction (2.9.1 *Processing instructions*) as the replacement value for the entity, as in the following example:

```
<!ENTITY r2  '<?tex \specialR?>'>
     <!-- when processing with TeX, use the \specialR command -->
```

If, on the other hand, the three kinds of r are being distinguished for some other kind of reason, perhaps to count their relative frequencies, with no particular concern as to how they are rendered, it might be preferable to convey the distinction by tagging them explicitly. In such a case, the replacement text for the three entities might look like the following:

```
<!ENTITY r  '<c type="1">r</c>'>
<!-- most common form of 'r'  -->
 <!ENTITY r2 '<c type="2">r</c>'>
<!-- secondary form of 'r'    -->
 <!ENTITY r3 '<c type="3">r</c>'>
<!-- third form of 'r'        -->
```

Finally, if the intention is for all three forms to be treated alike, we might supply declarations like the following:

```
<!ENTITY r  'r'>
<!-- most common form of 'r'  -->
 <!ENTITY r2 'r'>
<!-- secondary form of 'r'    -->
 <!ENTITY r3 'r'>
<!-- third form of 'r'        -->
```

When locally defined entities are used for the representation of characters in the text, for example to record presentational variants as in this example, a writing system declaration (25 *Writing System Declaration*) should be used to document their meaning.

4.2.2 Transliteration schemes

For lengthy transcriptions in scripts not supported by the document character set, entity references may prove unwieldy. In such cases, it is also possible to *transliterate* the material. In a transliteration scheme, glyphs properly associated with one abstract character are systematically redefined with another: for example, a glyph which 'looks like' an A (but is not in fact an A) might be transliterated as one. To avoid information loss, a *reversible* transliteration scheme (i.e. one in which it is possible to reconstruct the original writing from the transliteration) should be preferred. [59]

For example, using the Beta code transcription developed for ancient Greek by the Thesaurus Linguæ Græcæ,[60] one would transcribe the start of the *Iliad* of Homer thus:

[59] Many reversible transliteration schemes were defined in pre-Unicode days, see for example *ALA-LC Romanization Tables: Transliteration Schemes for Non-Roman Scripts*, approved by the Library of Congress and the American Library Association, tables compiled and edited by Randall K. Barry (Washington: Library of Congress, 1991).

[60] Thesaurus Linguæ Græcæ, *Beta Manual* (Irvine: TLG, [1988]). See also Luci Berkowitz and Karl A. Squitier, *Thesaurus Linguæ Græcæ Canon of Greek Authors and Works* 2nd edition (Oxford: Oxford University Press, 1986).

```
<l>*MH=NIN A)/EIDE QEA\ *PHLHI+A/DEW *)AXILH=OS
 OU)LOME/NHN, H(\ MURI/' *)AXAIOI=S A)/LGE' E)/QHKE,</l>
```

In an XML context, there is no particular reason to use a transliteration scheme, other than convenience of text preparation or the handling of legacy systems. If used, the transliteration scheme should be documented using a Writing System Declaration (25 *Writing System Declaration*). However, texts encoded using transliteration schemes are inherently non-standard, and the use of such schemes is thus deprecated except where circumstances permit no alternative.

4.3 Code shifting

Linguists use the term *code shifting* for the practice, common in many languages, of switching from one human language (such as French) to another (such as Kreol) within the speech of a single speaker. Many written documents also contain material from more than one language: loan words, quotations from foreign languages, etc. Since languages use a variety of *writing systems*, which in turn use a variety of *character repertoires*, shifts in language frequently go hand in hand with shifts in character repertoire and writing system. With the use of Unicode, a change in writing system or human language will not generally require any change in character encoding system; nevertheless, since language change is frequently of importance in meaningful processing of a document, the encoding scheme defined here provides a global attribute lang to make it possible to mark language shifts explicitly.

Some languages use more than one writing system. For example, some Slavic languages may be written either in the Latin or in the Cyrillic alphabet; some Turkic languages in Cyrillic, Latin, or Arabic script. In such cases, each writing system must be treated separately, as a separate 'language'. Each distinct value of the lang attribute, therefore, represents both a single natural language and a single writing system.[61]

Each value used for the lang attribute corresponds with the identifier of a <language> element defined in the <langUsage> element of the header of the TEI document concerned (see 5.4.2 *Language Usage*). This <language> element may additionally reference a writing system declaration, using its wsd attribute. The values may be taken from the two- or three-letter standard language codes defined by ISO-639:1988 or ISO 639-2:1998 respectively[62] or, if there is no applicable language code in the ISO 639 family of standards, from other appropriate lists of language identifiers.[63]

Like any global attribute, the lang attribute may be used on any element in the document. To mark a technical term, for example, as being in a particular language, one may simply specify the appropriate language on the <term> element (for which see 6.3.4 *Terms, Glosses, and Cited Words*):

```
<p lang="en"> ...
But then only will there be good ground of hope for the
further advance of knowledge, when there shall be received
and gathered together into natural history a variety of
experiments, which are of no use in themselves, but simply
serve to discover causes and axioms, which I call
<term lang="la">Experimenta lucifera</term>, experiments of
<term>light</term>, to distinguish them from those which I
call <term lang="la">fructifera</term>, experiments of
<term>fruit</term>.</p>
<p>Now experiments of this kind have one admirable
property and condition:  they never miss or fail. ...</p>
```

When more than one writing system is used for the same human language in a given document, it may be convenient to supply more than one <language> element, each of which will have an identifier derived from ISO 639, extended by an appropriate suffix. For example, a text containing material in

[61] When SGML is in use, the lang attribute also implies a particular coded character set (as defined by the associated WSD); in the XML context however, no change in character encoding is implied by a change in the lang value.

[62] *Codes for the Representation of Names of Languages-Part 2: Alpha-3 Code*, ([Geneva]: International Organization for Standardization, 1998). The list of language codes is also available from the the Library of Congress, which is the registration authority for ISO 639-2: see http://lcweb.loc.gov/standards/iso639-2/langhome.html.

[63] The SIL Ethnologue database at http://www.ethnologue.com/language_code_index.asp is a recommended alternative list of language identifiers.

Old Bulgarian, some parts of which use Cyrillic script, while other parts use Glagolitic script, might define a <langUsage> element like the following:

```
<langUsage>
<language id="OBG-CYR">Old Bulgarian, written in Cyrillic script.</language>
<language id="OBG-GLA">Old Bulgarian, written in Glagolitic script.</language>
<!-- other languages used or referenced in the manuscript description -->
</langUsage>
```

With these declarations in force, the language and writing system appropriate to any section of the text may be marked explicitly:

```
<div lang="OBG-GLA">
<head lang="OBG-CYR">....</head>
<p>...</p>
<!-- ... -->
</div>
```

Note that the language applicable to any element is inherited from its parent by default: in the above example, therefore, each element within the element not specifying otherwise is assumed to be written in Old Bulgarian using Glagolitic script. The <p> element shown above thus uses Glagolitic, while the <head> element uses Cyrillic.

Note that in general glyph distinctions are not specified by these Guidelines; if the glyphs to be used for a given encoded character are not available, application software may choose to display the material in an appropriate transliteration, as entity references, or in some other way.

If a formal writing system declaration is supplied it will be specified by the wsd attribute on the <language> element concerned (see further 4.4 *The Writing System Declaration*).

Any XML document may use an additional attribute xml:lang, the value of which is the identifier of a language from ISO 639 or registered with IANA. According to the XML Recommendation, the scope of this attribute is "considered to apply to all attributes and contents of the element where it is specified, unless overriden with an instance of xml:lang on another element within that content." (XML Recommendation, 2.12). Since the TEI DTD defines a great number of CDATA attributes with predeclared content in English, xml:lang cannot be used by TEI documents as intended in the XML recommendation. The current version of these Guidelines does not recommend use of the xml:lang attribute as a means of indicating language shifts; the TEI global lang attribute should instead be used for this purpose. This recommendation will be reviewed at the next revision of these Guidelines.

4.4 The Writing System Declaration

A Writing System Declaration (WSD) may be used to supply formal documentation for any language and writing system used in a document additional to that implied by its encoded form. It is particularly useful for documentation of non-standard encodings or transliterations. A WSD specifies:

- a formal name for the writing system and language
- a specification for the meaning of each character available in the writing system

The characters available in a writing system may be specified in the WSD for that writing system in one or more of the following ways:

- by reference to an international, national, or TEI-registered coded character set or entity set
- by reference to such a standard followed by formal declaration of all exceptions
- by providing a formal declaration for each character used

Individual characters within a WSD are formally declared, where necessary, by providing the following information:

- the unique code used to represent the character
- special properties such as whether the character is a diacritic mark or not

- brief textual description of the character
- standard or local entity name used for the character in interchange
- other standard identifiers for the character, if available, such as its code in the 'Universal Character Set' of ISO 10646 or Unicode
- optionally, some specification of a suitable graphic rendition for the character in a suitable notation (e.g. graphic image, Metafont program, etc.)

The writing system declaration is one of a set of *auxiliary* documents which provide documentation relevant to the processing of TEI texts. Auxiliary documents are themselves SGML or XML documents, for which document type declarations are provided. The DTD for the Writing System Declaration is discussed in detail in chapter 25 *Writing System Declaration*. Example Writing System Declarations may be obtained as described in chapter 37 *Obtaining TEI WSDs*.