

The Future of the TEI WSD

David J. Birnbaum

University of Pittsburgh

djbpitt+@pitt.edu

Euro Summer School:
Electronic Publishing for Cultural
Heritage Studies

Sofia, 28 September 2002

Outline

- The Siren Call of the Font Change
- The Problem: Gaiji
- The TEI P3 WSD
- XML and the Death of `sdata` and `subdoc`
- Why You May Not Need a WSD
- XML Alternatives to the TEI P3 WSD

The Siren Call of the Font Change

- System- and application-specific
- Mixture of levels
 - Character set is informational units
 - Font is typographic properties
 - Early Cyrillic characters may be in Times Roman or Courier fonts, and not just “black letter”

Gaiji

- ‘External character’
- Text item not present in base character set

The TEI WSD

- Text Encoding Initiative (TEI)
- Writing System Declaration (WSD)
 - Encoded as an “auxiliary SGML Document” (subdoc)
 - Used for two different purposes
 - Documentation
 - Processing
- Each text element is ...
 - Encoded as an *entity* in a document (e.g., &aos;)
 - Described in a <form> element in a WSD ...

The `<form>` Element

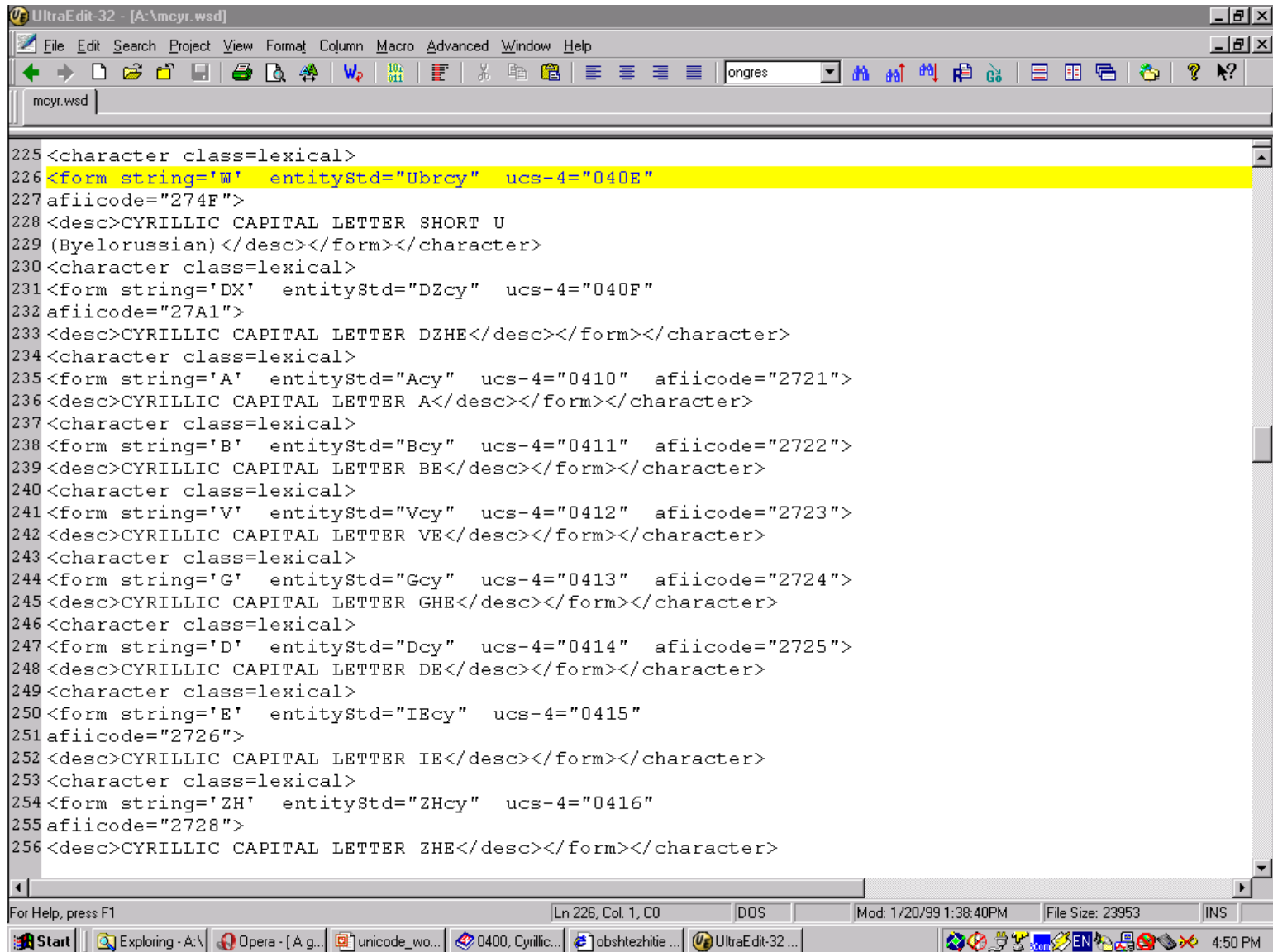
- Attributes of the `<form>` element
 - `string`: byte string
 - `codedCharSet`: base character set for string
 - `entityStd`: entity name for character
 - **`entityLoc`**: entity name for character
 - **`ucs-4`**: Unicode-related 32-bit identifier
- No glyph identifier
 - **`afiicode`** attribute removed from P4
 - RIP Association for Font Information Interchange (AFII)

TEI WSD (P3 Example)

```
<character class="lexical">  
<form    entityLoc="aos"  
        ucs-4="0430"  
        afiicode="10993"  
</character>
```

(See *Computer Standards and Interfaces* 18 [1996]: 201–252)

A Real WSD



```
UltraEdit-32 - [A:\mcyr.wsd]
File Edit Search Project View Format Column Macro Advanced Window Help
mcyr.wsd
225 <character class=lexical>
226 <form string='W' entityStd="Ubrcy" ucs-4="040E">
227 afiicode="274F">
228 <desc>CYRILLIC CAPITAL LETTER SHORT U
229 (Byelorussian)</desc></form></character>
230 <character class=lexical>
231 <form string='DX' entityStd="DZcy" ucs-4="040F">
232 afiicode="27A1">
233 <desc>CYRILLIC CAPITAL LETTER DZHE</desc></form></character>
234 <character class=lexical>
235 <form string='A' entityStd="Acy" ucs-4="0410" afiicode="2721">
236 <desc>CYRILLIC CAPITAL LETTER A</desc></form></character>
237 <character class=lexical>
238 <form string='B' entityStd="Bcy" ucs-4="0411" afiicode="2722">
239 <desc>CYRILLIC CAPITAL LETTER BE</desc></form></character>
240 <character class=lexical>
241 <form string='V' entityStd="Vcy" ucs-4="0412" afiicode="2723">
242 <desc>CYRILLIC CAPITAL LETTER VE</desc></form></character>
243 <character class=lexical>
244 <form string='G' entityStd="Gcy" ucs-4="0413" afiicode="2724">
245 <desc>CYRILLIC CAPITAL LETTER GHE</desc></form></character>
246 <character class=lexical>
247 <form string='D' entityStd="Dcy" ucs-4="0414" afiicode="2725">
248 <desc>CYRILLIC CAPITAL LETTER DE</desc></form></character>
249 <character class=lexical>
250 <form string='E' entityStd="IEcy" ucs-4="0415">
251 afiicode="2726">
252 <desc>CYRILLIC CAPITAL LETTER IE</desc></form></character>
253 <character class=lexical>
254 <form string='ZH' entityStd="ZHcy" ucs-4="0416">
255 afiicode="2728">
256 <desc>CYRILLIC CAPITAL LETTER ZHE</desc></form></character>
For Help, press F1
Ln 226, Col. 1, CO DOS Mod: 1/20/99 1:38:40PM File Size: 23953 INS
Start Exploring - A:\ Opera - [A g... unicode_wo... 0400, Cyrillic... obshtezhitie... UltraEdit-32... 4:50 PM
```

How sdata Entities Work

1. Begin parsing document
2. Upon reaching WSD declaration, parse WSD and build three-column look-up table (entity, character, glyph) in memory or on disk
3. As each sdata entity is encountered during parsing:
 - a. Throw away regular entity replacement string
 - b. Use entity name associated with output sdata node as pointer into the table that you built at step #2
 - c. Retrieve and insert value from appropriate column

The WSD Solution

- `sdata` entities (e.g., `&aos;`) encode pointers into WSD, which records Unicode values and glyphs
- `sdata` encodings may represent many-to-many relationships among Unicode values and glyphs
 - Example: 1 ǃ are represented by different `sdata` entities, where the WSD `<form>` elements have the same `ucs-4` attribute but different `aficode` attributes
 - Similarity is encoded through shared character pointers
 - Difference is encoded through different glyph pointers

The WSD in Action

- Simple example
 - Encoding: `<p>Cyrillic &aos;</p>`
 - Rendering: Cyrillic а
- Complex example:
 - Encoding: `<p>Cyrillic &juos; &juros;</p>`
 - Rendering: Cyrillic 1 ъ
 - Same `ucs-4` value
(`u+044e` CYRILLIC SMALL LETTER YU)
 - Different `glyph (afii code)` values
 - Conflate or distinguish (lump or split), as needed

TEI, XML, and the WSD

- XML excludes subdoc
 - Alternatives are available: ndata entities, TEI header, in-line encoding
- XML excludes sdata
 - XML parsing produces a tree (document object model, DOM)
 - SGML sdata entities represented by special DOM node type after parsing
 - XML lacks sdata, and resolves all regular entities to character data during parsing (impossible to “throw away regular entity replacement string”)

The Problem

- The WSD encodes the fact that two items are simultaneously the same and different
- If the ability to encode that fact is to be retained under XML, it needs to be provided without subdoc and without sdata

Alternatives to subdoc 1

- ndata
- Non-parsed external data entity
- Consistent with TEI P3 subdoc approach
- Used in TEI P4

```
<!NOTATION wsd PUBLIC '-//TEI P3-  
1994//NOTATION Writing System Declaration//EN'>  
<!ENTITY myWSD SYSTEM "myWSD.xml" NDATA  
wsd>
```

Linking the WSD to the Main Document

```
<TEI.2>  
  <teiHeader>  
    <!-- ... -->  
    <language id="ChSl" wsd="myWSD">  
    <!-- ... -->  
  </teiHeader>  
  <text lang="ChSl">  
    <!-- ... -->  
  </text>  
</TEI.2>
```

Alternatives to subdoc 2

- TEI header
 - May be implemented as additional tag set (“pizza topping”)
 - Consistent with encoding of ISO character entities in TEI as part of regular document (DTD)
- In-line
 - Requires multiple encodings of identical information

sdata and beyond

- sdata
 - SGML only
- XML Character-Level Markup
 - Cumbersome
 - Doesn't work in attribute values
- Regular Characters (transliteration)
 - Ambiguous
- Unicode Variation Selector
- Unicode Private Use

Alternatives to sdata

- The output of parsing XML is nodes and character data (text)
- Entities are resolved early in parsing
 - An entity cannot be distinguished from its replacement text after parsing
- Gaiji must be represented through characters

Transliteration

- Published Early Cyrillic transliteration systems
 - *Polata* 1981
 - *Polata* 1987
 - Grünberg 1995
 - Cleminson 1997
 - Lazov 2000
- Requirements
 - Unambiguous
 - Reversible
 - Separation of content and markup
 - Relatively stateless

Character Representation of Gaiji

- Maintaining state: need to backtrack
- Perniciously (or cryptically) stateful
 - Regular characters (traditional transliteration)
- Mildly stateful
 - Regular characters with regular character delimiters
 - Regular characters with private use character delimiters
- Stateless
 - Private use characters

Why You May Not Need a WSD

- XML In-Line Markup
- Unicode Variation Selector
- Unicode Private Use
- Transliteration

XML In-Line Markup

- `a<gaiji glyph="z">b</gaiji>c`
- Christian Wittern, Kyoto, Buddhist texts
- Limitations
 - Fatal: Not available in attribute values
 - Pernicious: Requires multiple encoding of identical information

Unicode Variation Selector 1

- Introduced in Unicode Version 3.2
- Postposed combining character
- Specify glyphic variants of a common character (common semantics) in plain text
- Affects only appearance
- *Uses must be codified, similarly to new characters*
- Ignorable
- Commercial support?

Unicode Variation Selector 2

- Currently encoded for Mongolian and mathematics
 - Mongolian variation selectors
 - Sixteen generic variation selectors:
 - from: `u+fe00 VARIATION SELECTOR-1`
 - to: `u+fe0f VARIATION SELECTOR-16`
- Cyrillic proposal under preparation by Everson, Birnbaum, and Cleminson (based on Birnbaum 1996)

Unicode Private Use

- 6,400 basic (16-bit) characters reserved for “private use”
 - u+e000 to u+f8ff
- 131,068 surrogate pairs reserved for “private use”
- Guaranteed never to be assigned
- Commercial support?
- Prior agreement required
- May replace `sdata` in XML-compatible WSD alternative

Prior Agreement Required

- “Successful interchange requires agreement between sender and receiver regarding interpretation of private-use codes.”
- “These codes can be freely used for characters of any purpose, but successful interchange requires agreement between sender and receiver on their interpretation.”

Two Approaches to Private Use

1. Assert that text elements not (yet) present in Unicode are characters
 - PUA characters are intended to be rendered directly
 - Splitter Heaven
2. Replace old `sdata` entities as pointers into WSD
 - PUA characters are not intended to be rendered directly
 - PUA characters survive parsing, and are processed afterwards (e.g., by XSLT stylesheet) by mapping to some value in `<form>` element

Conclusions

- WSD in P3 form is not available under XML
- Gaiji may be represented by private use characters, variation selectors (or transliteration)
- Writing systems may be documented in TEI header, `ndata` entity (or in line)