An Undocumented Method of Filtering and Translating Structural SGML to HTML using Citec Multidoc Pro Style Sheets

Dr. Daniel Paul O’Donnell
University of Lethbridge, Department of English
daniel.odonnell@uleth.ca
Fax: +1 (403) 382-7191 | Voice: +1 (403) 329-2377
Matthew Van Egmond
Research Assistant
University of Lethbridge

DOI (added 2018): 10.5281/zenodo.1193818

Abstract: This practice note describes an undocumented, inexpensive, and easy-to-use process by which structurally encoded SGML documents can be translated to HTML using Citec Multidoc Pro SGML Browser Style Sheets. The method is simple to implement yet versatile enough to allow developers to process even quite complicated documents on short notice. It should be of particular interest to smaller projects where funding and encoders’ time is at a premium.

Introduction

Formatting structurally encoded SGML (Standard General Markup Language) documents for display can be a time consuming and expensive business. The very emphasis on structure over layout that makes these languages so versatile for computer processing ensures that they are also relatively difficult to format for display to human readers. Either a translation script is required to convert the structurally encoded mark-up to one of the more popular display languages or the document in question must be viewed with dedicated SGML software. Document distribution is, as a result, almost never spontaneous. It can, indeed, represent a significant investment as staff and resources that could perhaps better be devoted to content development are used to perform basic formatting tasks.

The results of this time and expense, moreover, can be aesthetically disappointing. Smaller markets ensure that all but the most expensive SGML applications lack the sophisticated display capabilities common in commercial HTML (Hypertext Markup Language) browsers. Writing such functions into a customized translation script can increase development time and costs; leaving them out can cause technically unsophisticated users to question the utility of the project as a whole. From the point of view of funding agencies, supervisors, and the general public, these difficulties in display and distribution can make structural-encoding projects seem very eccentric indeed: compared to their HTML-based cousins, structural projects cost more money, take more time to complete, and often require special equipment—all to produce an output that, to the uninitiated at least, can often seem less sophisticated than the average high school senior’s home page.

This practice note describes an inexpensive and easy-to-use solution to this problem. It uses Citec Multidoc Pro SGML Browse Style Sheets in an undocumented way to create templates for assigning HTML mark-up to SGML elements. The method is simple to implement yet versatile enough to allow developers to process even quite complicated documents. Its only requirements are the Citec browser and a relatively good knowledge of HTML coding practices.

The remainder of this note is divided into three sections. The first provides some background information on the Multidoc Style Sheet Language and its associated editor; the second illustrates how these can be used to accomplish some common SGML to HTML translation tasks; the third discusses implementation problems including limitations on the procedure’s use and suggestions for streamlining and standardizing certain translation operations.

The Multidoc Style Sheet Language and Editor

Multidoc Style Sheets are SGML documents that map element names from an SGML source document to display instructions used by the Multidoc Browser. Each element in the source document is assigned a single `<STYLE>` element in the style sheet. These `<STYLE>` elements in turn contain nested information about the source element’s display: visibility, typeface, paragraph structure, background color, etc.

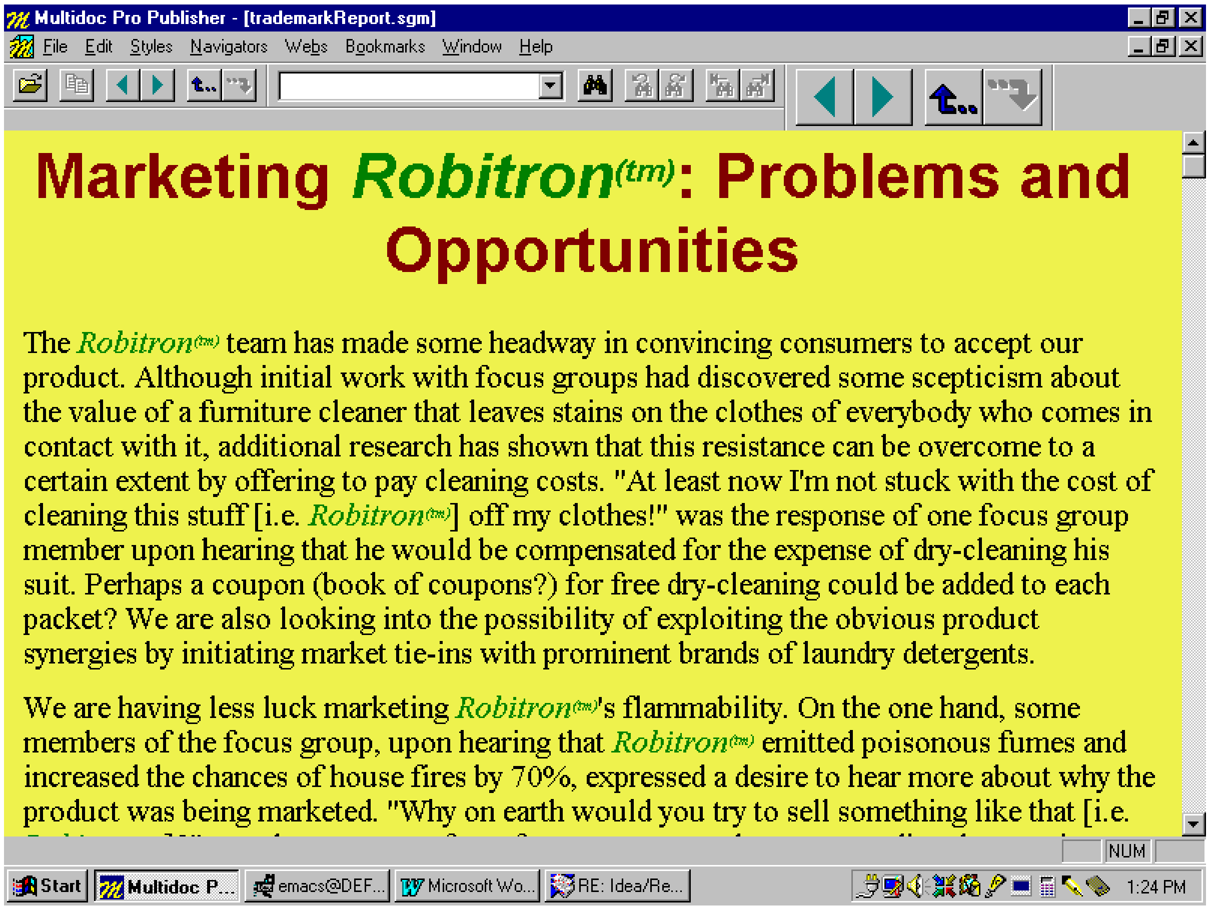
Display qualities can be assigned explicitly in the style sheet or left to be inherited from surrounding elements in the source. Qualities not explicitly assigned are inherited. This allows designers to assign a few key features to a given element without necessarily having to specify every single aspect of its display or even be able to predict every context in which it might appear. In the following fragment from a hypothetical company marketing report, for example, the SGML element `<TRADEMARK>` has been given three specific qualities: a green face, italic slant, and the appended text string "(tm)". All other aspects of the element’s appearance are left to the surrounding context. A `<TRADEMARK>` in the document’s title, for example, might therefore be expected to have a different font face, size, weight, and paragraph alignment than one found in a main body paragraph:

Figure A: Fragment from a Multidoc Style Sheet (emphasis added)

```
<!DOCTYPE STYLESHEET PUBLIC "-//SYNEX INFORMATION AB//DTD
STYLESHEET V2.00//EN">
<STYLESHEET> <em>
<STYLE TAG="TRADEMARK">
<FONT-SLANT V="ITALIC">
<FONT-COLOR V="GREEN">
<Z-TEXT V='(tm)'>
</STYLE></em>
```

```
<STYLE TAG="TITLE">
<FONT-FAMILY V="ARIEL">
<FONT-WEIGHT V="BOLD">
<FONT-SIZE V=32>
<FONT-COLOR V="MAROON">
<BREAK-BEFORE>
<BREAK-AFTER>
</STYLE>
<STYLE TAG="P">
<FONT-FAMILY V="TIMES">
<FONT-WEIGHT V="MEDIUM">
<FONT-SIZE V=16>
<BREAK-BEFORE>
<BREAK-AFTER>
</STYLE>
```

Figure B: Document Formatted for Display in Multidoc Pro SGML Browser Using Style Sheet in Figure A


Multidoc Pro Publisher - [trademarkReport.sgm]
File Edit Styles Navigators Webs Bookmarks Window Help
NUM
Start Multidoc P... emacs@DEF... Microsoft Wo... RE: Idea/Re...
1:24 PM

Marketing *Robitron*(tm): Problems and Opportunities

The *Robitron*(tm) team has made some headway in convincing consumers to accept our product. Although initial work with focus groups had discovered some scepticism about the value of a furniture cleaner that leaves stains on the clothes of everybody who comes in contact with it, additional research has shown that this resistance can be overcome to a certain extent by offering to pay cleaning costs. "At least now I'm not stuck with the cost of cleaning this stuff [i.e. *Robitron*(tm)] off my clothes!" was the response of one focus group member upon hearing that he would be compensated for the expense of dry-cleaning his suit. Perhaps a coupon (book of coupons?) for free dry-cleaning could be added to each packet? We are also looking into the possibility of exploiting the obvious product synergies by initiating market tie-ins with prominent brands of laundry detergents.

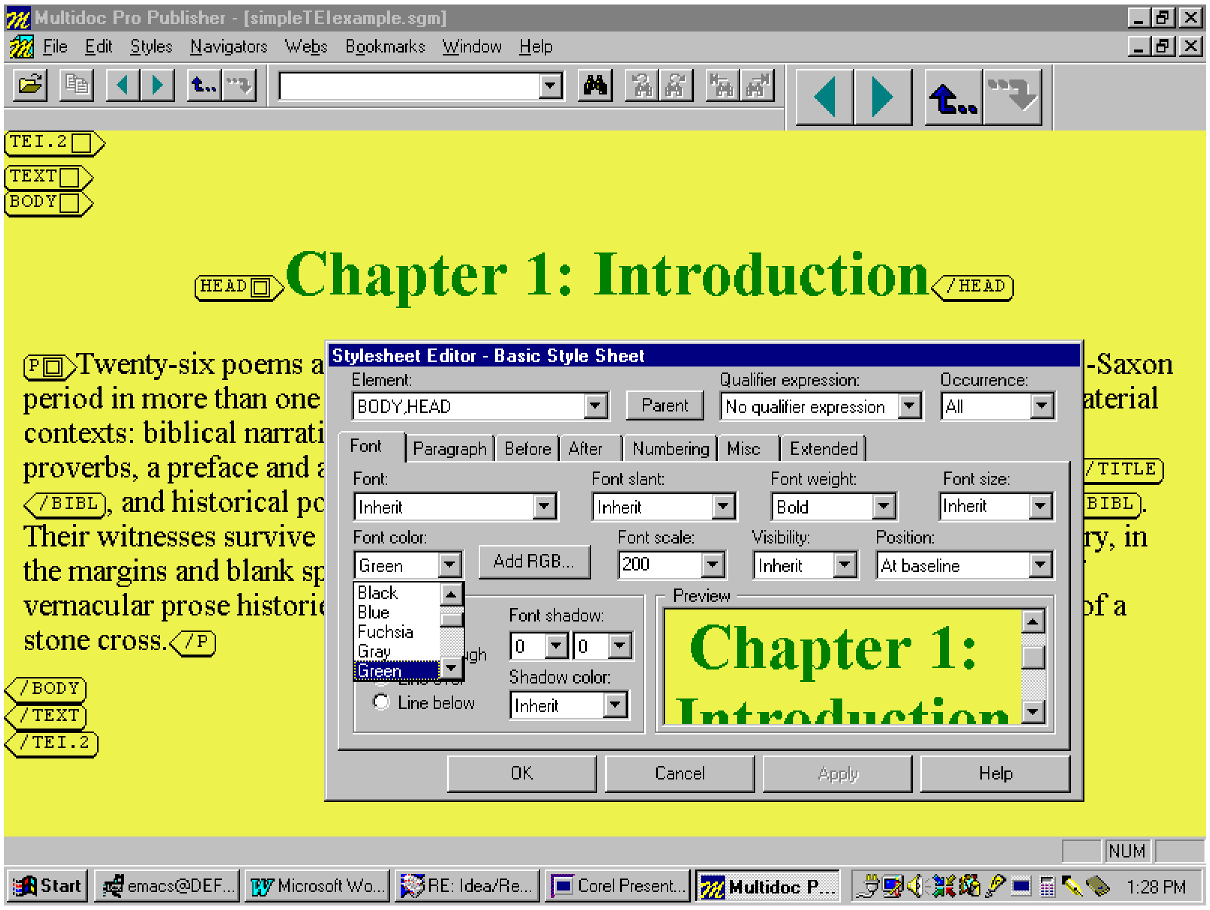
We are having less luck marketing *Robitron*(tm)'s flammability. On the one hand, some members of the focus group, upon hearing that *Robitron*(tm) emitted poisonous fumes and increased the chances of house fires by 70%, expressed a desire to hear more about why the product was being marketed. "Why on earth would you try to sell something like that [i.e.

Editing Multidoc Style Sheets

Because they are SGML documents in their own right, Multidoc Style Sheets can be composed or edited by hand using a simple text editor or word processor. This can be useful in ensuring consistency in the rendition of individual elements across multiple style sheets or in developing new sheets from existing ones.

An awareness of SGML is not necessary, however, for most style sheet operations. The developer’s version of the software, Multidoc Pro, comes packaged with a wizard-like editing utility that allows developers to create style sheets graphically, without any knowledge of their underlying structure. Using this utility, styles are assigned and edited by ‘pointing and clicking’ at specific instances of individual SGML elements in a source document (The style sheet editor will not work unless the developer calls a specific document instance into the browser’s memory; once edited, however, style sheets are independent SGML documents and will work with any SGML document showing similar coding). A relatively intuitive editor interface then presents developers with a catalogue of display features arranged under a logical series of tabs covering such aspects as font colour, paragraph alignment, the assignment of preceding and following text-strings, etc. A small preview window allows developers to see how the element will appear after changes made in the editor take effect (Figure C):

Figure C: Using the Multidoc Style Sheet Editor to Change the Appearance of an SGML Element


Multidoc Pro Publisher - [simpleTEIexample.sgm]
File Edit Styles Navigators Webs Bookmarks Window Help
TEI.2
TEXT
BODY
HEAD
/HEAD
P
/BIBL
/P
/BODY
/TEXT
/TEI.2
/TITLE
BIBL
Stylesheet Editor - Basic Style Sheet
Font Paragraph Before After Numbering Misc Extended
Add RGB...
Parent
Black
Blue
Fuchsia
Gray
Green
Line below
Preview
OK Cancel Apply Help
NUM
Start emacs@DEF... Microsoft Wo... RE: Idea/Re... Corel Present... Multidoc P...
1:28 PM

Chapter 1: Introduction

Twenty-six poems a… -Saxon period in more than one … aterial contexts: biblical narrati… proverbs, a preface and a… and historical po… Their witnesses survive … ry, in the margins and blank sp… vernacular prose historie… of a stone cross.

Element: BODY,HEAD
Qualifier expression: No qualifier expression
Occurrence: All
Font: Inherit
Font slant: Inherit
Font weight: Bold
Font size: Inherit
Font color: Green
Font scale: 200
Visibility: Inherit
Position: At baseline
Font shadow: 0 0
Shadow color: Inherit

The style sheet editor interface makes layout design a visual and relatively intuitive process. Display features like text and background color, typeface, and paragraph justification can be adjusted by eye on screen, and style sheet designers can lay out entire SGML documents using skills they have acquired using any commercial WYSIWYG (What You See Is What You Get) word processor. While advanced layout operations may require a more detailed knowledge of SGML or project-specific conventions, the graphical nature of the interface means that most aspects of style sheet design can be left to project members with relatively little knowledge of SGML—professional document designers or junior staff-members as the project's aesthetic requirements and budget allow.

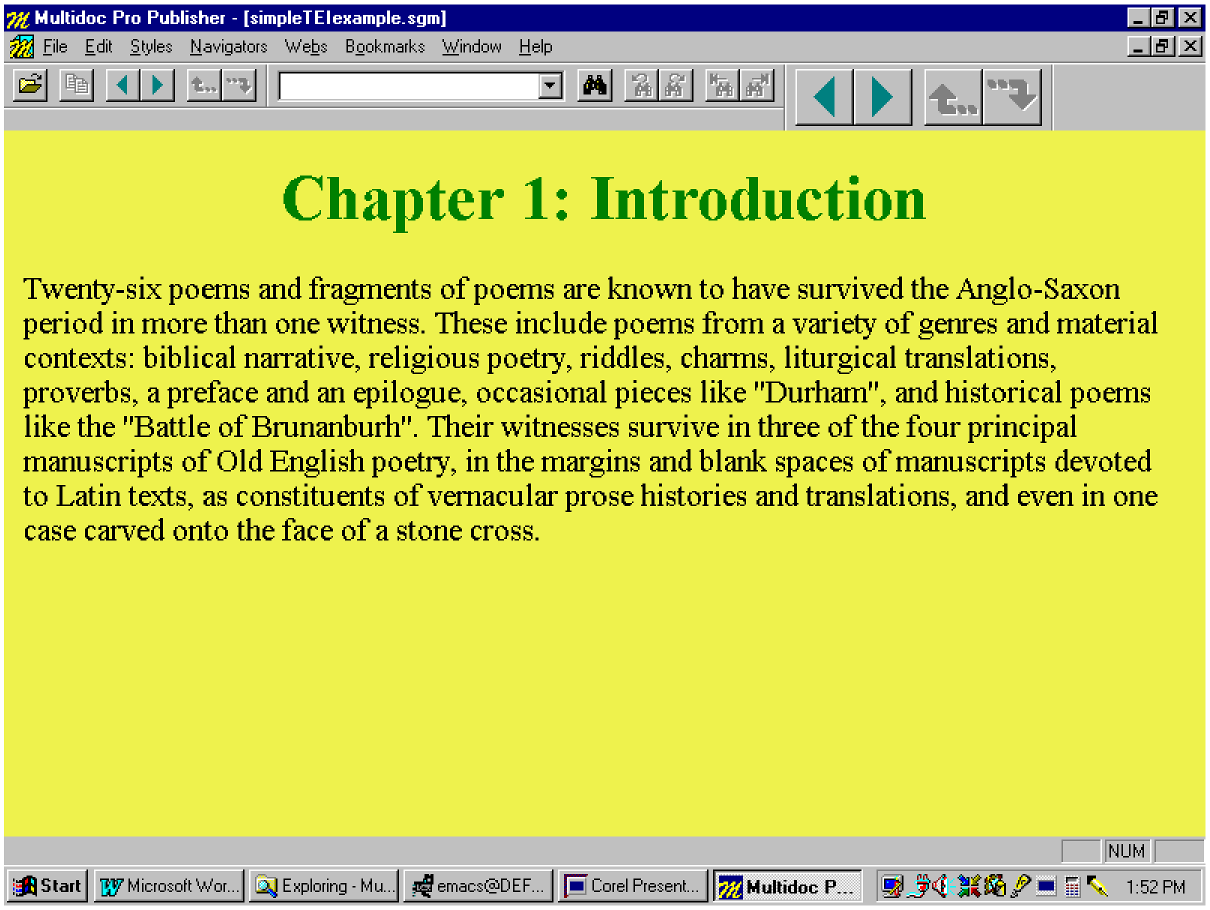
Using Multidoc Style Sheets to Translate SGML to HTML

The actual process by which Multidoc Style Sheets are used to translate SGML to HTML is, conceptually speaking at least, quite straightforward. It can, indeed, be reduced to essentially two steps.

1. Taking advantage of the style sheet language's ability to assign customizable text-strings to individual SGML elements, developers create a style sheet template in which the original document's mark-up is "shadowed" by HTML codes describing how source document elements are to be displayed.
2. Documents formatted with this template are then copied into a text editor or word processor and saved as an HTML file. They are, in most cases, then ready for immediate use with commercial HTML browsers.

How this works in practice can be illustrated by the following basic example. Figure D shows a simple SGML prose document encoded according to the guidelines of the TEI (Text Encoding Initiative) and formatted for display in the Multidoc Browser (Although it is not always necessary to pre-format an SGML document for display before translating it to HTML, doing so does make the entire process significantly easier: developers construct their initial layout using the browser's graphic-oriented and intuitive editor, and simplify their HTML coding by restricting it to providing a description of the already formatted document; this modularity also allows easy modification by other project members or end users):

Figure D: SGML Document as Formatted for Display in Multidoc Browser

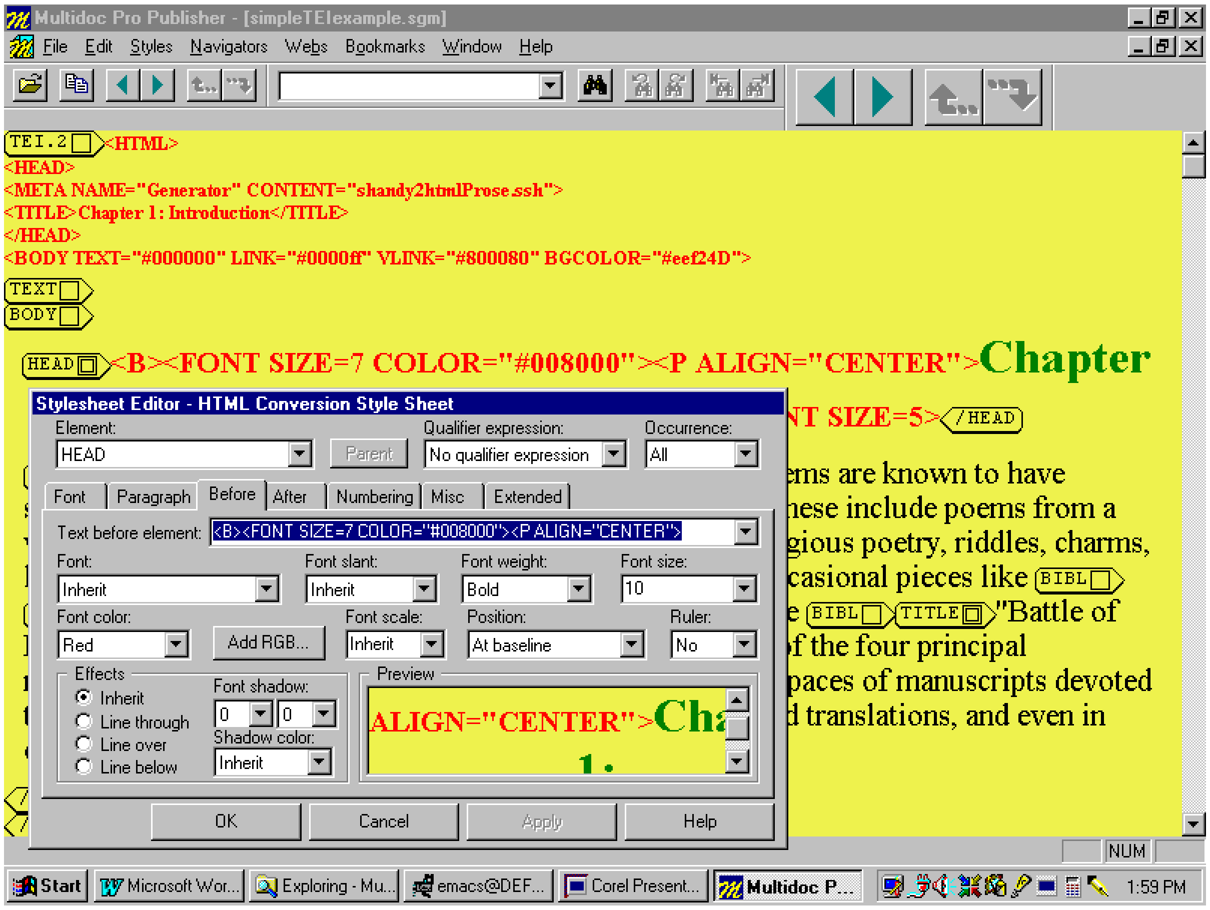

Multidoc Pro Publisher - [simpleTEIexample.sgm]
File Edit Styles Navigators Webs Bookmarks Window Help
NUM
Start
Microsoft Wor...
Exploring - Mu...
emacs@DEF...
Corel Present...
Multidoc P...
1:52 PM

Chapter 1: Introduction

Twenty-six poems and fragments of poems are known to have survived the Anglo-Saxon period in more than one witness. These include poems from a variety of genres and material contexts: biblical narrative, religious poetry, riddles, charms, liturgical translations, proverbs, a preface and an epilogue, occasional pieces like "Durham", and historical poems like the "Battle of Brunanburh". Their witnesses survive in three of the four principal manuscripts of Old English poetry, in the margins and blank spaces of manuscripts devoted to Latin texts, as constituents of vernacular prose histories and translations, and even in one case carved onto the face of a stone cross.

In Figure E the same document is shown as it is used to assign HTML mark-up to the style sheet. HTML tags describing the desired layout are entered as text-strings using the editing interface's "Before" and "After" tabs (In the actual style sheet language, these strings are encoded as attributes to the SGML elements `<A-TEXT>` and `<Z-TEXT>` [see Figure A for an example]. Although all aspects of style sheet design discussed in this practice note can be performed using a text editor, the discussion assumes the editing interface is being used). In the illustration, start tags for the HTML elements `<B>`, `<FONT>` and `<P>` are being added as "Before" text to the SGML element `<HEAD>`. Closing tags for these qualities are usually found under the corresponding "After" tab. In the background, HTML header information can be seen. This has been assigned to the SGML document's root element (`<TEI.2>`).

Figure E: Adding HTML Tagging Using the Style Sheet Editor


Multidoc Pro Publisher - [simpleTEIexample.sgm]
File Edit Styles Navigators Webs Bookmarks Window Help

```
TEI.2 <HTML>
<HEAD>
<META NAME="Generator" CONTENT="shandy2htmlProse.ssh">
<TITLE>Chapter 1: Introduction</TITLE>
</HEAD>
<BODY TEXT="#000000" LINK="#0000ff" VLINK="#800080" BGCOLOR="#eef24D">
TEXT
BODY
HEAD <B><FONT SIZE=7 COLOR="#008000"><P ALIGN="CENTER">Chapter
NT SIZE=5> /HEAD
ems are known to have
hese include poems from a
gious poetry, riddles, charms,
casional pieces like BIBL
e BIBL TITLE "Battle of
f the four principal
paces of manuscripts devoted
d translations, and even in
```

Stylesheet Editor - HTML Conversion Style Sheet

Element: HEAD — Qualifier expression: No qualifier expression — Occurrence: All

Tabs: Font | Paragraph | Before | After | Numbering | Misc | Extended

Text before element: `<B><FONT SIZE=7 COLOR="#008000"><P ALIGN="CENTER">`

Font: Inherit — Font slant: Inherit — Font weight: Bold — Font size: 10

Font color: Red — Add RGB... — Font scale: Inherit — Position: At baseline — Ruler: No

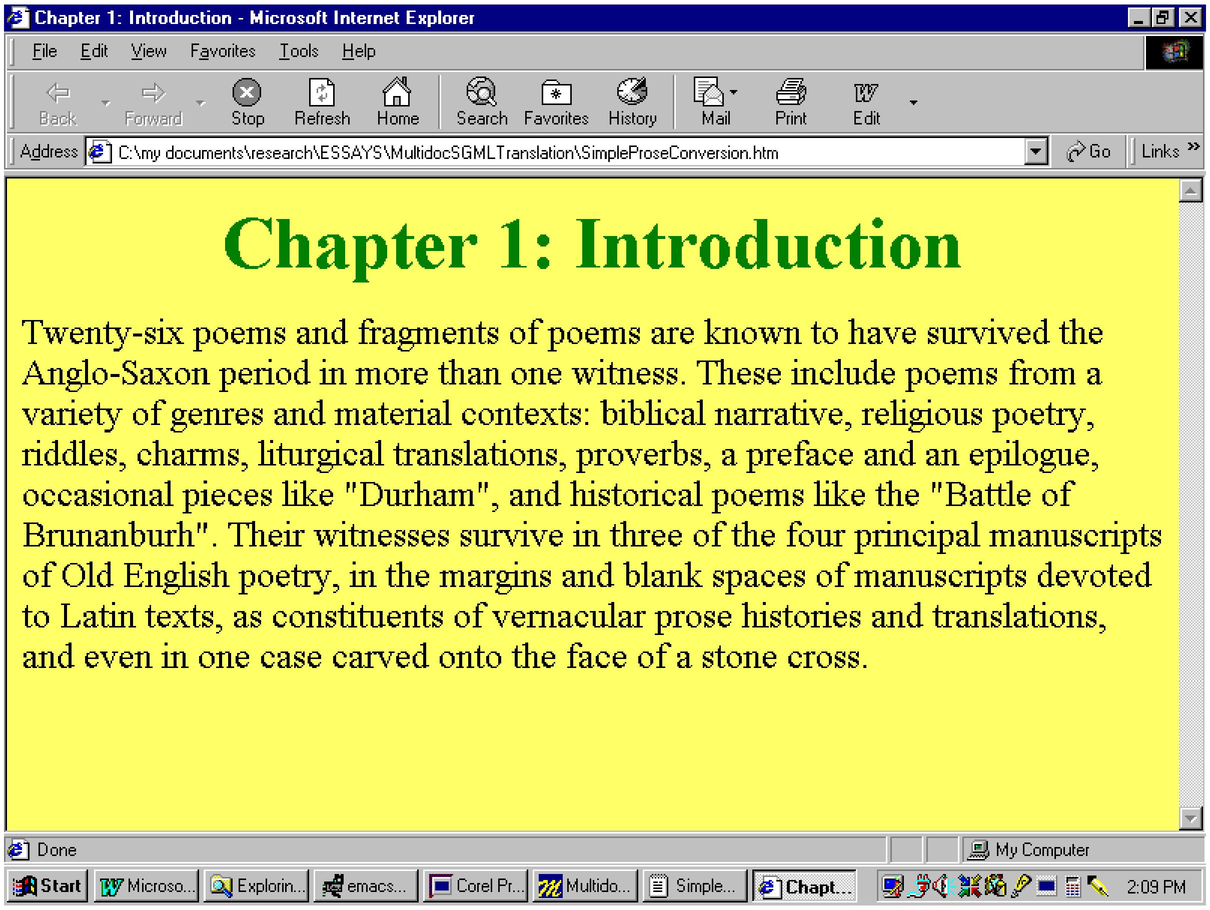
Effects: (●) Inherit ( ) Line through ( ) Line over ( ) Line below

Font shadow: 0 0 — Shadow color: Inherit

Preview: ALIGN="CENTER">Cha 1.

Once the necessary HTML mark-up has been added, the original SGML tagging is suppressed and the resulting display—a legal HTML document—copied to a standard text editor or word processor for saving and, if necessary, post-processing. Figure G shows the document as it is displayed in a commercial HTML browser:

Figure G: Output of Figure E as Displayed in Commercial HTML Browser


Chapter 1: Introduction - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address C:\my documents\research\ESSAYS\MultidocSGMLTranslation\SimpleProseConversion.htm
Go Links
Done
My Computer
Start Microso... Explorin... emacs... Corel Pr... Multido... Simple... Chapt...
2:09 PM

Chapter 1: Introduction

Twenty-six poems and fragments of poems are known to have survived the Anglo-Saxon period in more than one witness. These include poems from a variety of genres and material contexts: biblical narrative, religious poetry, riddles, charms, liturgical translations, proverbs, a preface and an epilogue, occasional pieces like "Durham", and historical poems like the "Battle of Brunanburh". Their witnesses survive in three of the four principal manuscripts of Old English poetry, in the margins and blank spaces of manuscripts devoted to Latin texts, as constituents of vernacular prose histories and translations, and even in one case carved onto the face of a stone cross.

More Complex Translations

The example discussed in Figure D through Figure G is extremely simple. Each SGML element has exactly one appropriate HTML mark-up. There are no structural incompatibilities between the SGML source and HTML output. The SGML document is a simple prose text with no links, bookmarks, or tables. The HTML output introduces no features not found in the original SGML. The entire translation, indeed, could perhaps just as easily have been accomplished using the "Search and Replace" function on a standard word processor.

Most SGML to HTML translations are more complex, however. There may be structural incompatibilities between the SGML source and HTML output. There may be attribute information to preserve or translate. Individual SGML elements may require different HTML formatting in different contexts or when they contain different types of content. And developers will almost certainly want to add display, navigation, and scripting effects to the HTML output.

It is in these contexts that the Multidoc Style Sheet Language comes into its own as an SGML to HTML filtering and translation application. Using a small number of key variable expressions, the language provides developers with a powerful set of tools for accomplishing sophisticated transformations. With these tools and sufficient ingenuity, developers ought to be able to accomplish most translation tasks. The following section provides some examples of common translation problems, arranged in order of relative difficulty.

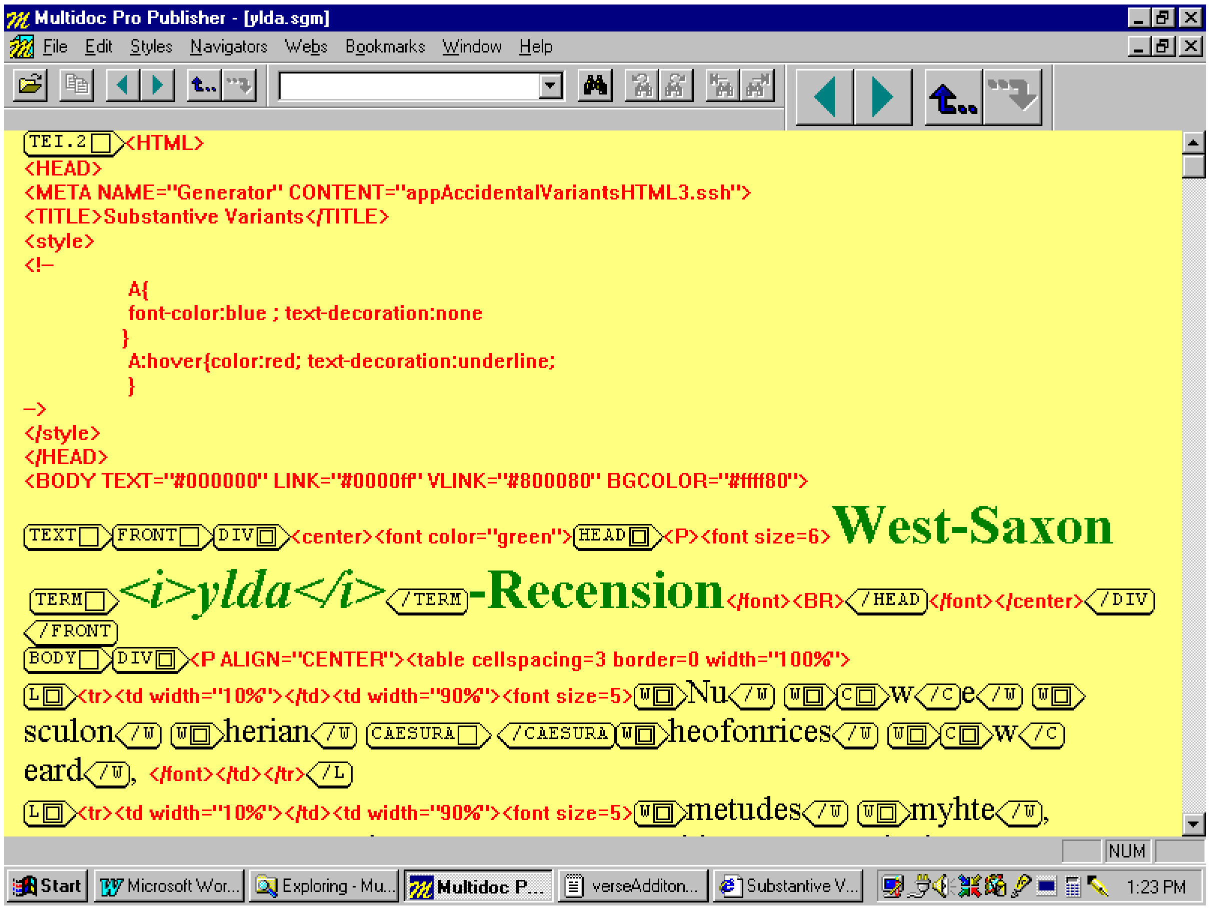
Adding HTML Effects and Scripting Not Found in the SGML Source

A frequent goal in translating SGML documents to HTML involves the introduction of coding and scripts for effects or layout not present in the original SGML document. Such material can appear in the HTML header or document body. In the document header, it might involve the addition of descriptive keywords and titles, scripts for dynamic events, or style information controlling the appearance of specific HTML tags. In the document body, it might include the creation of HTML tables for positioning SGML content, or script and attribute information necessary for the correct functioning of specific interactive effects.

In practice, this additional material can have a spectacular effect on display, ease of use, and the mechanics of navigation. In theory, however, such coding is essentially identical to the descriptive HTML mark-up illustrated in Figure D through Figure G. Like this descriptive coding, scripts and other additional material are assigned to specific elements in the source document using the style sheet editor’s "Before" and "After" tabs (the escape codes `\n` and `\tab` can be used to indicate line breaks and tabs in longer sequences). The main difference lies in the sometimes advanced knowledge of HTML such sequences demand, and, occasionally, the ingenuity required for ensuring their proper integration into the basic HTML template.

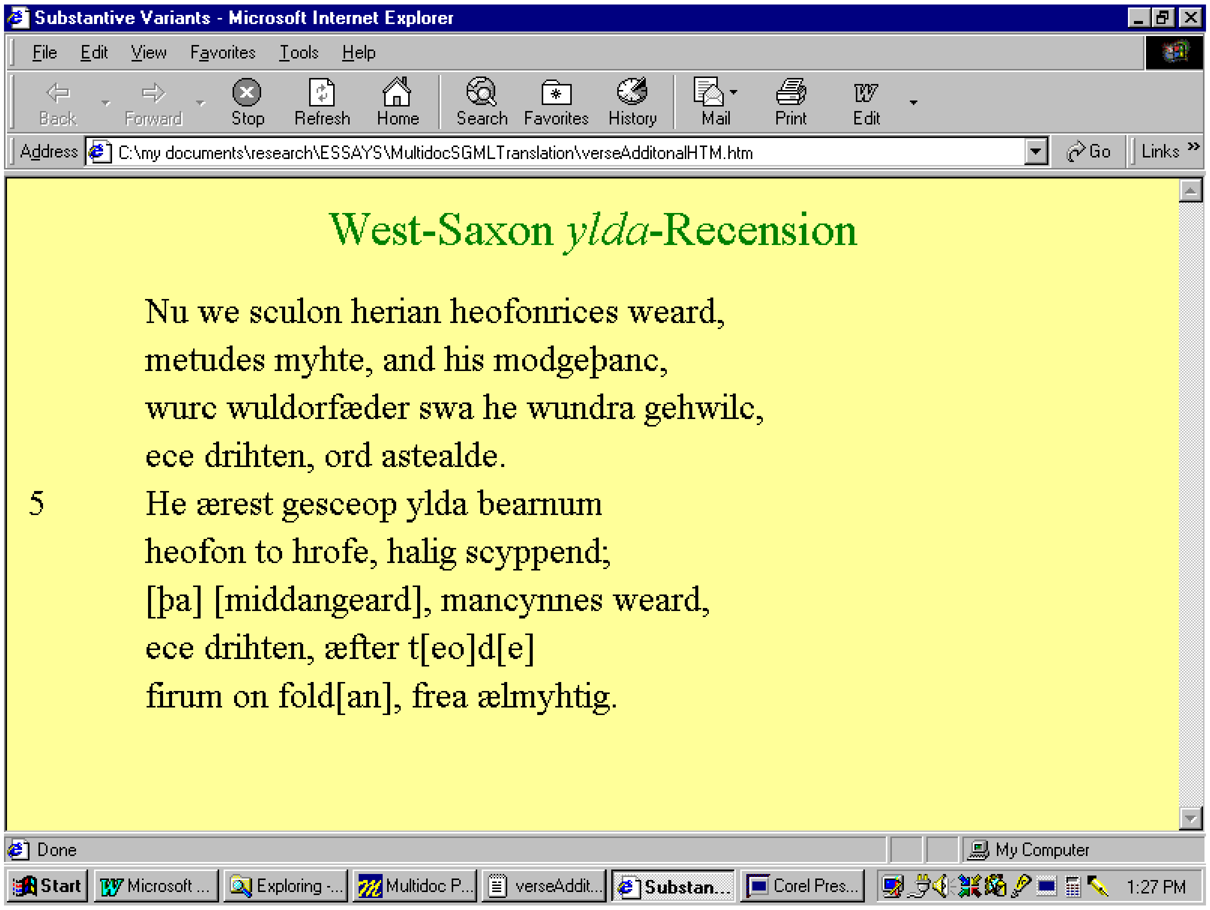
The following example from the *Electronic Cædmon’s Hymn* shows two uses of such additional material (Figure H): a CSS (Cascading Style Sheet) style fragment has been added to the HTML header to control the appearance of links in the output document and the main body has been reformatted as an HTML table to ensure correct spatial alignment (Figure I shows a text formatted with this style sheet as displayed by a commercial HTML browser). In both cases, the additional coding has been introduced into the style sheet using the "Before" and "After" tabs. Information about HTML style has been assigned (with other header information) to SGML root element (`<TEI.2>`). Tags for constructing the HTML table are entered before SGML `<DIV>` and `<L>` elements as appropriate.

Figure H: Adding <STYLE> Information and HTML Table Formatting to SGML Style Sheet



```
TEI.2 <HTML>
<HEAD>
<META NAME="Generator" CONTENT="appAccidentalVariantsHTML3.ssh">
<TITLE>Substantive Variants</TITLE>
<style>
<!–
        A{
        font-color:blue ; text-decoration:none
       }
        A:hover{color:red; text-decoration:underline;
        }
–>
</style>
</HEAD>
<BODY TEXT="#000000" LINK="#0000ff" VLINK="#800080" BGCOLOR="#ffff80">
TEXT FRONT DIV <center><font color="green"> HEAD <P><font size=6> West-Saxon
TERM <i>ylda</i> /TERM -Recension</font><BR> /HEAD </font></center> /DIV
/FRONT
BODY DIV <P ALIGN="CENTER"><table cellspacing=3 border=0 width="100%">
L <tr><td width="10%"></td><td width="90%"><font size=5> W Nu /W  W C w /C e /W  W
sculon /W  W herian /W  CAESURA  /CAESURA W heofonrices /W  W C w /C
eard /W,  </font></td></tr> /L
L <tr><td width="10%"></td><td width="90%"><font size=5> W metudes /W  W myhte /W,
```

Figure I: Output of Figure H as Displayed in Commercial HTML Browser


Substantive Variants - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address
Go
Links
Done
My Computer
Start
Microsoft ...
Exploring -...
Multidoc P...
verseAddit...
Substan...
Corel Pres...
1:27 PM

C:\my documents\research\ESSAYS\MultidocSGMLTranslation\verseAdditonalHTM.htm

West-Saxon *ylda*-Recension

Nu we sculon herian heofonrices weard,
metudes myhte, and his modgeþanc,
wurc wuldorfæder swa he wundra gehwilc,
ece drihten, ord astealde.
5 He ærest gesceop ylda bearnum
heofon to hrofe, halig scyppend;
[þa] [middangeard], mancynnes weard,
ece drihten, æfter t[eo]d[e]
firum on fold[an], frea ælmyhtig.

Most scripts and additional HTML tagging added in the translation process will, naturally, be concerned with appearance and navigation. But such material can also have a structural function. The *Electronic Cædmon's Hymn*, for example, uses DHTML (Dynamic Hyper-Text Markup Language) event scripting to overcome an important structural incompatibility in the way notes are handled by the TEI and standard HTML.

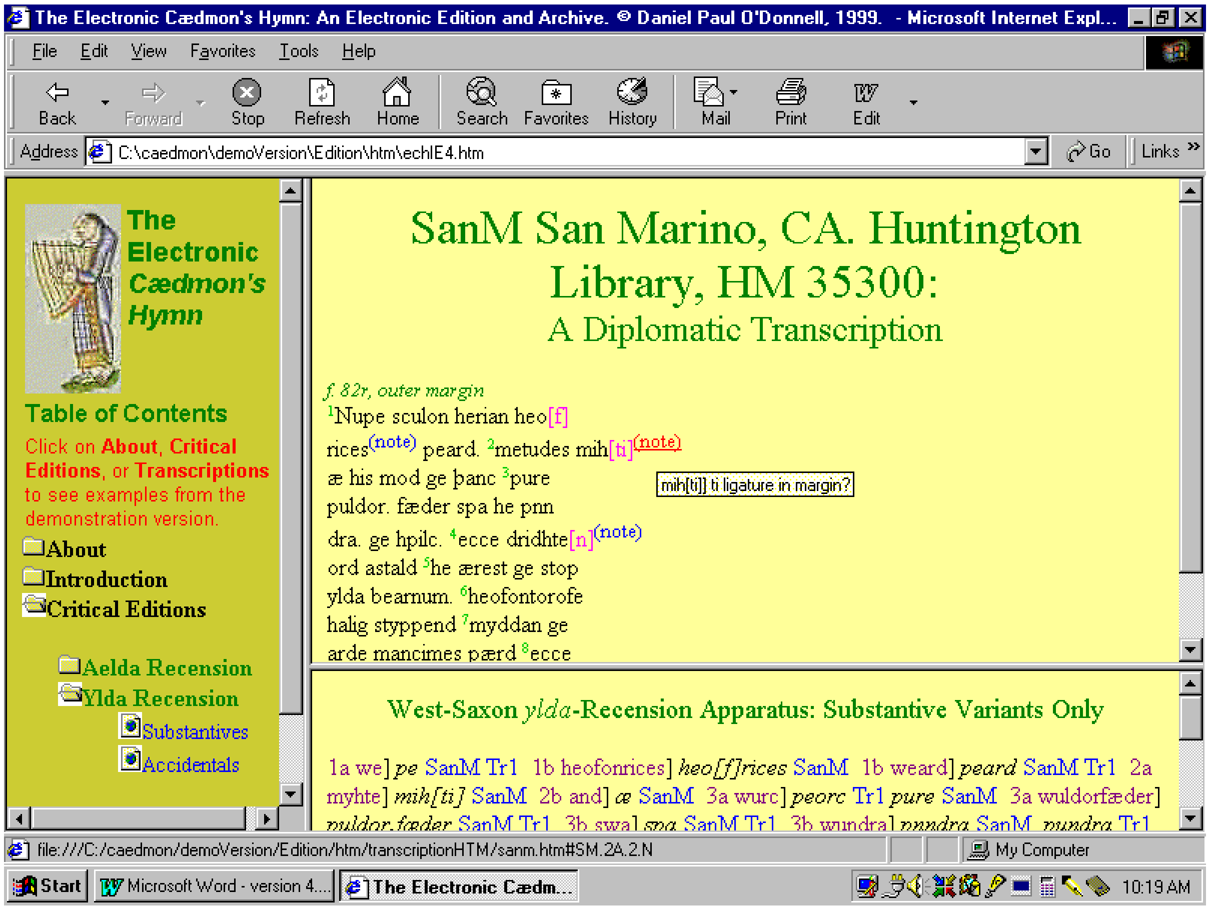
The TEI uses the element `<NOTE>` to encode "any additional comment found in the text, [and] marked in some way as being out of the main textual stream" ([ACH/ACL/ALLC Text Encoding Initiative, P3], §6.8.1). The content of this element can range from a single word to multiple paragraphs. Notes can be positioned at the point of reference, elsewhere in the same document, or in a separate SGML document. The TEI guidelines, however, recommend placing them at the point of reference ([ACH/ACL/ALLC Text Encoding Initiative, P3], §6.8.1). In the Multidoc Browser, such notes are commonly replaced with an icon that indicates the presence of an annotation without otherwise interrupting the flow of the text. Users access the note, which then appears in a separate window, by clicking on the icon.

HTML has no equivalent to the TEI `<NOTE>`, and no separate mark-up for distinguishing annotation from the surrounding "main textual stream." In practice, therefore, most HTML texts collect annotations in a distinct document or document section, marking their location with hyperlinks at the point of reference.

This structural incompatibility means that TEI notes encoded in the preferred position cannot, except in the simplest of cases, be translated directly into HTML: `<NOTE>` elements must either be removed from their original location and replaced with pointers, or they must be distinguished in some way from the flow of the surrounding text. The first approach, which may be preferable in the case of more complicated annotation, is beyond the relatively limited extraction capabilities of the Multidoc Style Sheet Language. It will almost certainly require some pre- or post-processing. The second approach, which is appropriate for simple or short notes, can be accomplished in a variety of ways depending on the nature of the annotation's contents and context. In some cases—bibliographic references in running prose, for example—enclosing the annotation in parenthesis at the point of reference may be a sufficient solution. Other types of annotation, such as grammatical glosses or context sensitive help information, however, may prove more intrusive than useful if they are placed in the document's main running text.

The *Electronic Cædmon's Hymn* solves this structural problem by converting all glosses, simple annotations, and tips to 'mouseOver' events. Users access the annotation by placing the mouse over the glossed term or an unambiguous 'icon' in the text. This in turn causes a small "Tool Tip" window containing the note or context-sensitive tip to appear—associating notes with their referents in a relatively unobtrusive fashion and eliminating in the process the need to extract SGML content from the document (Figure J).

Figure J: Translating an SGML `<NOTE>` as a mouseOver Event (Top Right Frame)


The Electronic Cædmon's Hymn: An Electronic Edition and Archive. © Daniel Paul O'Donnell, 1999. - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address C:\caedmon\demoVersion\Edition\htm\echlE4.htm
Go Links

The Electronic Cædmon's Hymn

Table of Contents

Click on **About**, **Critical Editions**, or **Transcriptions** to see examples from the demonstration version.

About
Introduction
Critical Editions
Aelda Recension
Ylda Recension
Substantives
Accidentals

SanM San Marino, CA. Huntington Library, HM 35300:
A Diplomatic Transcription

f. 82r, outer margin
1Nupe sculon herian heo[f]
rices(note) peard. 2metudes mih[ti](note)
æ his mod ge þanc 3pure
puldor. fæder spa he pnn
dra. ge hpilc. 4ecce dridhte[n](note)
ord astald 5he ærest ge stop
ylda bearnum. 6heofontorofe
halig styppend 7myddan ge
arde mancimes pærd 8ecce

mih[ti]] ti ligature in margin?

West-Saxon *ylda*-Recension Apparatus: Substantive Variants Only

1a we] *pe* SanM Tr1 1b heofonrices] *heo[f]rices* SanM 1b weard] *peard* SanM Tr1 2a myhte] *mih[ti]* SanM 2b and] *æ* SanM 3a wurc] *peorc* Tr1 *pure* SanM 3a wuldorfæder] *puldor.fæder* SanM Tr1 3b swa] *spa* SanM Tr1 3b wundra] *pundra* SanM *pundra* Tr1

file:///C:/caedmon/demoVersion/Edition/htm/transcriptionHTM/sanm.htm#SM.2A.2.N

My Computer
Start Microsoft Word - version 4... The Electronic Cædm... 10:19 AM

The effect is accomplished by assigning the following "Before" and "After" strings to the appropriate TEI `<NOTE>` elements (Figure K). If only some `<NOTE>` elements in the original document are to receive this presentation, a conditional expression may be required (see "Conditional Expressions," below). 'Tool Tips' also require some scripting. This is attached, with other HTML header information, to the SGML root.

Figure K: "Before" and "After" Text for Incorporating SGML `<NOTE>` in mouseOver "Tool Tip"

Before Text:

```
<A onMouseOver="showtip(this,event,'
```

After Text:

```
')" onMouseOut="hidetip()"><SUP>(note)</SUP></A>
```

Extracting and Manipulating SGML Attribute References

The examples discussed above are complicated only in the knowledge of HTML they require. The HTML tags are in all cases still entered as simple text-strings under the style sheet editor's "Before" and "After" tabs. A more complicated, but still very common, task in translating SGML documents to HTML involves the extraction, preservation, or manipulation of ATTREFs (attribute references) in the source and output documents. Such operations might typically include:

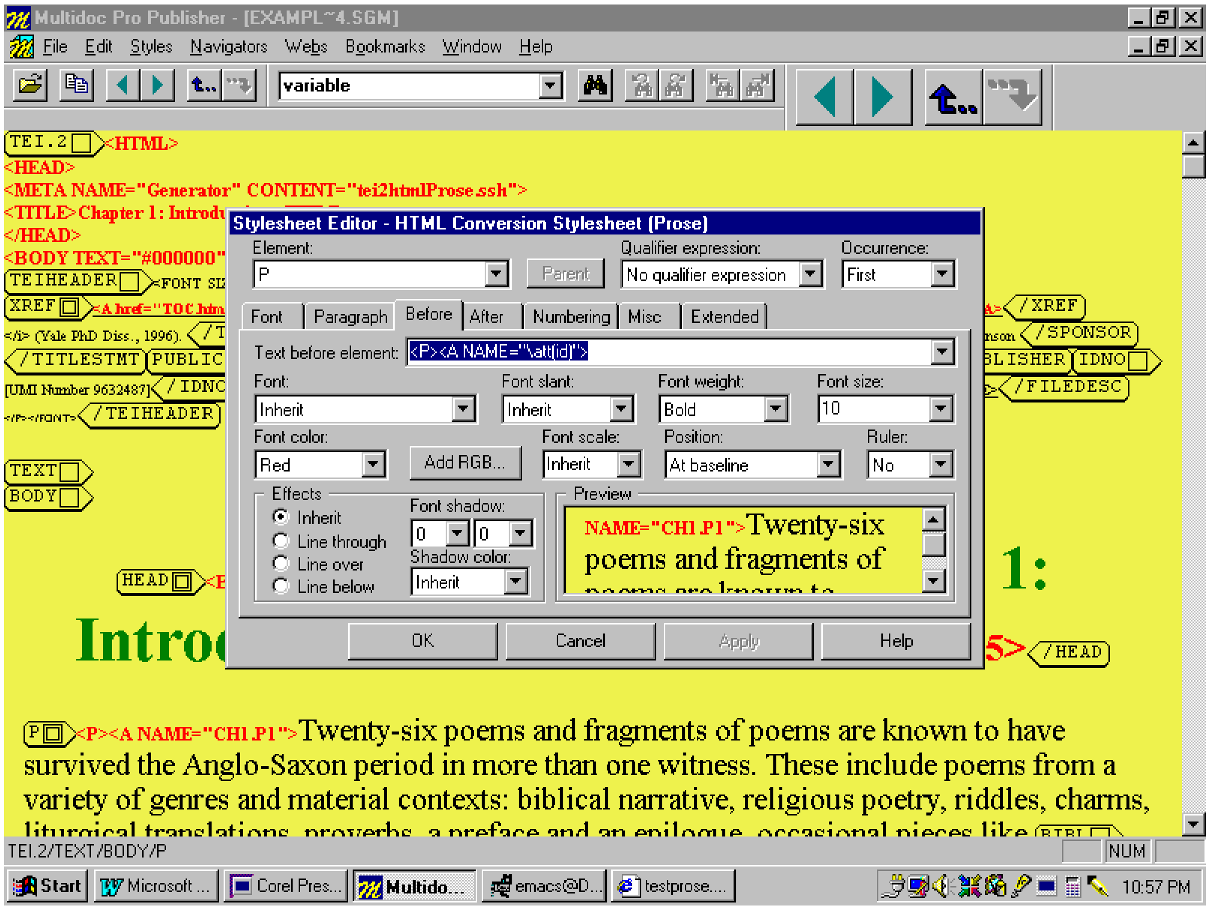
- reusing an SGML ID or NAME reference in the HTML output
- extracting number values or glossary identifications from SGML elements for use as HTML content
- assigning different HTML layouts to a single SGML element on the basis of some significant attribute value

These manipulations can take a variety of forms, depending on the specific syntax of the source and output languages and the goals of the translation. Regardless of the details of the case in question, however, all manipulations make use of the same basic variables, `\att(attribute_name)` for basic attribute extraction, `\ifatt(attribute_name = attribute_value)` for conditional expressions.

Basic Extraction

Basic extraction of SGML attribute values is accomplished using the `\att(attribute_name)` variable. Entered under the "Before" or "After" tab, this retrieves and outputs the value of the specified SGML attribute as a plain text string. In Figure M, for example, the variable `\att(id)` is being used to extract the value of the SGML attribute `ID` to supply the `NAME` attribute on an HTML bookmark:

Figure M: Extracting SGML ATTREFs Using the `\att(attribute_name)` Variable

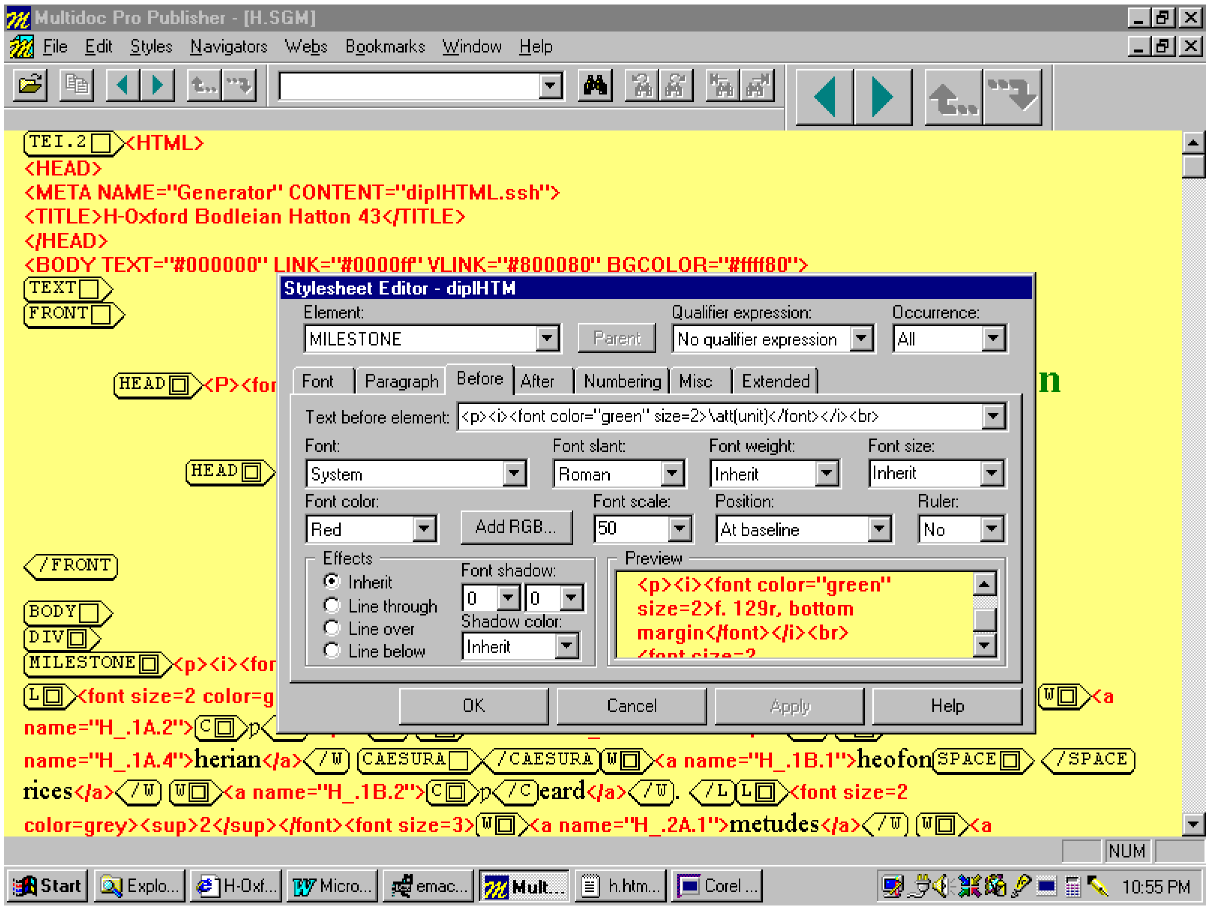

Multidoc Pro Publisher - [EXAMPL~4.SGM]
File Edit Styles Navigators Webs Bookmarks Window Help
variable
Stylesheet Editor - HTML Conversion Stylesheet (Prose)
Font Paragraph Before After Numbering Misc Extended
OK Cancel Apply Help
TEI.2/TEXT/BODY/P
NUM
Start Microsoft ... Corel Pres... Multido... emacs@D... testprose....
10:57 PM

TEI.2 <HTML>
<HEAD>
<META NAME="Generator" CONTENT="tei2htmlProse.ssh">
<TITLE>Chapter 1: Introdu
</HEAD>
<BODY TEXT="#000000"
TEIHEADER <FONT SI
XREF <A href="TOC.htm
</i> (Yale PhD Diss., 1996). /T
/TITLESTMT PUBLIC
[UMI Number 9632487] /IDNC
</P></FONT> /TEIHEADER
TEXT
BODY
HEAD <E
Intro
A> /XREF
nson /SPONSOR
BLISHER IDNO
/FILEDESC
1:
5> /HEAD
P <P><A NAME="CH1.P1">Twenty-six poems and fragments of poems are known to have survived the Anglo-Saxon period in more than one witness. These include poems from a variety of genres and material contexts: biblical narrative, religious poetry, riddles, charms, liturgical translations, proverbs, a preface and an epilogue, occasional pieces like

Element: P
Qualifier expression: No qualifier expression
Occurrence: First
Text before element: <P><A NAME="\att(id)">
Font: Inherit
Font slant: Inherit
Font weight: Bold
Font size: 10
Font color: Red
Font scale: Inherit
Position: At baseline
Ruler: No
Effects: [✓ Inherit] Line through Line over Line below
Font shadow: 0 0
Shadow color: Inherit
Preview: NAME="CH1.P1">Twenty-six poems and fragments of poems are known to

The same variable can be used to extract SGML attribute values for use as HTML text. In the example presented in Figure N and Figure O, this has been done twice: Initial information about the location of the text in question ("f. 129r, Bottom Margin"), has been extracted from the `UNIT` attribute of the TEI element `<MILESTONE>`; small green superscript numbers in the main body of the text represent references to a canonical line-numbering system have been extracted from the `N` attribute of the TEI element `<L>`:

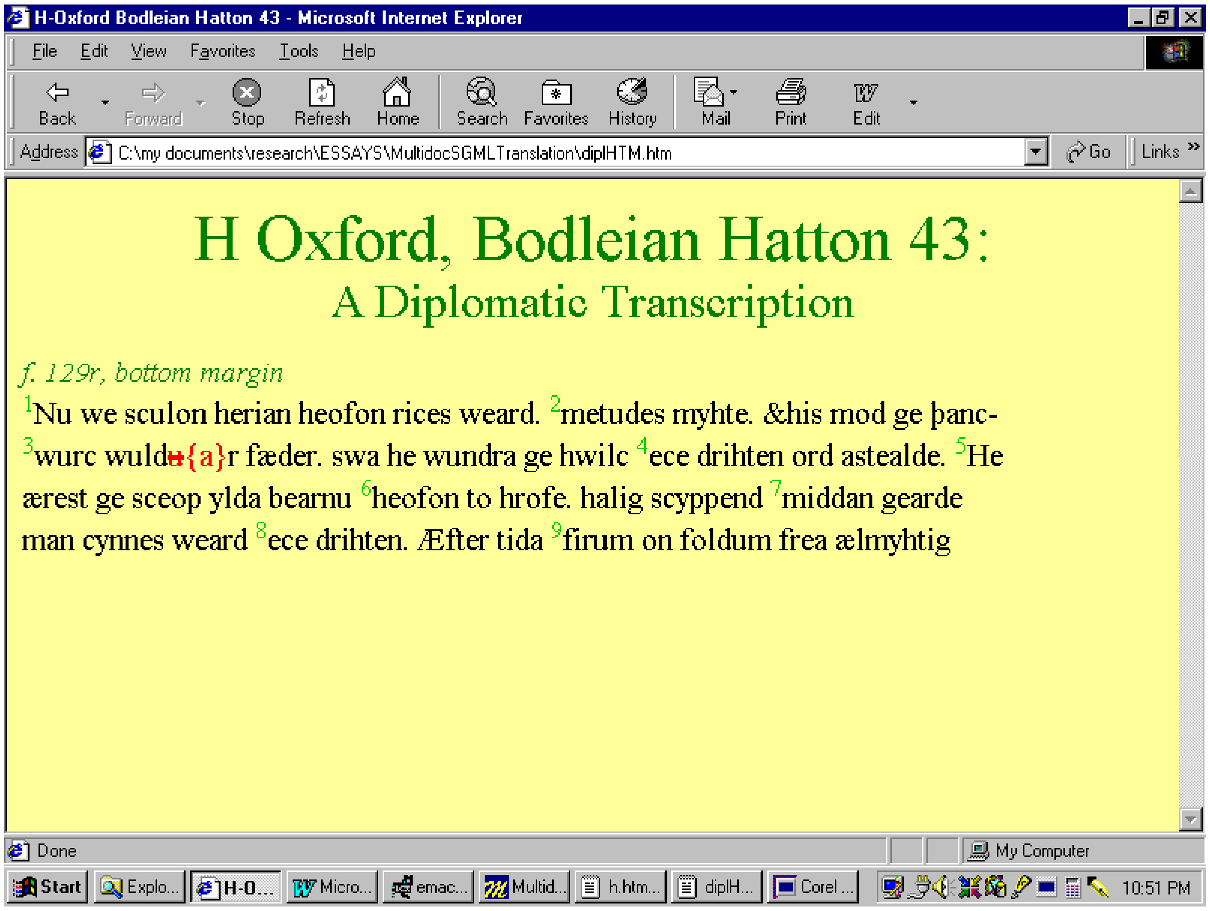
Figure N: Creating HTML Content From SGML ATTREFs Using the `\att(attribute_name)` Variable


Multidoc Pro Publisher - [H.SGM]
File Edit Styles Navigators Webs Bookmarks Window Help
Stylesheet Editor - diplHTM
Parent
Font Paragraph Before After Numbering Misc Extended
OK Cancel Apply Help
NUM
Start Explo... H-Oxf... Micro... emac... Mult... h.htm... Corel ...
10:55 PM

```
TEI.2 <HTML>
<HEAD>
<META NAME="Generator" CONTENT="diplHTML.ssh">
<TITLE>H-Oxford Bodleian Hatton 43</TITLE>
</HEAD>
<BODY TEXT="#000000" LINK="#0000ff" VLINK="#800080" BGCOLOR="#ffff80">
TEXT
FRONT
    HEAD <P><for
        HEAD
/FRONT
BODY
DIV
MILESTONE <p><i><for
L <font size=2 color=g                                              W <a
name="H_.1A.2"> C p
name="H_.1A.4">herian</a> /W CAESURA /CAESURA W <a name="H_.1B.1">heofon SPACE /SPACE
rices</a> /W W <a name="H_.1B.2"> C p /C eard</a> /W . /L L <font size=2
color=grey><sup>2</sup></font><font size=3> W <a name="H_.2A.1">metudes</a> /W W <a
```

Element: MILESTONE
Qualifier expression: No qualifier expression
Occurrence: All
Text before element: <p><i><font color="green" size=2>\att(unit)</font></i>
Font: System
Font slant: Roman
Font weight: Inherit
Font size: Inherit
Font color: Red
Add RGB...
Font scale: 50
Position: At baseline
Ruler: No
Effects: [✓ Inherit] Line through Line over Line below
Font shadow: 0 0
Shadow color: Inherit
Preview:
<p><i><font color="green" size=2>f. 129r, bottom margin</font></i>
<font size=2

Figure O: Output of Figure N as Displayed in Commercial HTML Browser


H-Oxford Bodleian Hatton 43 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address
Go Links
Done
My Computer
Start Explo... H-O... Micro... emac... Multid... h.htm... diplH... Corel ...
10:51 PM

C:\my documents\research\ESSAYS\MultidocSGMLTranslation\diplHTM.htm

H Oxford, Bodleian Hatton 43:

A Diplomatic Transcription

f. 129r, bottom margin

^{1}Nu we sculon herian heofon rices weard. ^{2}metudes myhte. &his mod ge þanc-
^{3}wurc wuld~~u~~{a}r fæder. swa he wundra ge hwilc ^{4}ece drihten ord astealde. ^{5}He
ærest ge sceop ylda bearnu ^{6}heofon to hrofe. halig scyppend ^{7}middan gearde
man cynnes weard ^{8}ece drihten. Æfter tida ^{9}firum on foldum frea ælmyhtig

Using ATTREFs in Conditional Expressions

SGML attribute values can also be used to assign context sensitive HTML mark-up to individual elements. The following fragment from a hypothetical TEI-encoded edition of the eighteenth-century novel *Tristram Shandy*, for example, distinguishes between two types of annotation: 'authorial' notes written for the original text by the novel's author, and 'editorial' notes recording the observations and commentary of a subsequent, hypothetical and relatively dim, modern editor (Figure P):

Figure P: Fragment from a Hypothetical, TEI-Encoded Edition of Tristram Shandy

```
<TEI.2>
<!-- TEI header material has been omitted -->
<TEXT>
<BODY>
<DIV ID="5.25" TYPE="chapter">
<HEAD ID="5.25.t">Chapter Twenty-Five
</HEAD>
<P ID="5.25.1">'Tis a point settled,--and I mention it for the
comfort of Confucius
<PTR ID="5.25.1.ptr.1" TYPE="authorial" TARGET="5.25.1.an.1">, who
is apt to get entangled in telling a plain story--that provided he
keeps along the line of his story,--he may go backwards and
forwards as he will,--'tis still held to be no digression.
<PTR ID="5.25.1.ptr.2" TYPE="editorial" TARGET="5.25.1.en.1">
</P>
<P ID="5.25.2">This being premised, I take the benefit of the
<EMPH>act of going backwards
</EMPH> myself.
</P>
</DIV>
<!-- Elsewhere in the Document -->
<DIV TYPE="notes">
<NOTE ID="5.25.1.an.1" N="1" TYPE="authorial"
TARGET="5.25.1.ptr.1">Mr. Shandy is supposed to mean ***** ***
***, Esq.; Member for ******,--and not the Chinese Legislator.
</NOTE>
<NOTE ID="5.25.1.en.1" TYPE="editorial" TARGET="5.25.1.ptr.2">This
passage is almost certainly a reference to a contemporary event.
The "Confucius" mentioned in Sterne's authorial note has not been
identified.
</NOTE>
</DIV>
</BODY>
</TEXT>
</TEI.2>
```

In translating this fragment for HTML display, the designer decides to use different symbols for hyperlinks referring to each type of note: a small red, superscript and bold *A* for notes by the original author, and a similarly formatted *E* for notes by the modern editor. Since the original document uses a single SGML element (`<PTR>`) for internal links to both types of textual notes, separating these references in the HTML output requires the use of a conditional expression based on the distinguishing ATTREF. When the SGML attribute `TYPE` has the value "authorial," the style sheet adds an "A" 'icon'. When it has the value "editorial", an "E". This content-sensitive mark-up is expressed by a conditional statement added to the SGML element's "After" tab (Figure Q; the condition is emphasized):

Figure Q: "Before" and "After" Text for Conditional HTML Markup of SGML `<PTR>` Element in Figure P (Conditional Expression Emphasized)

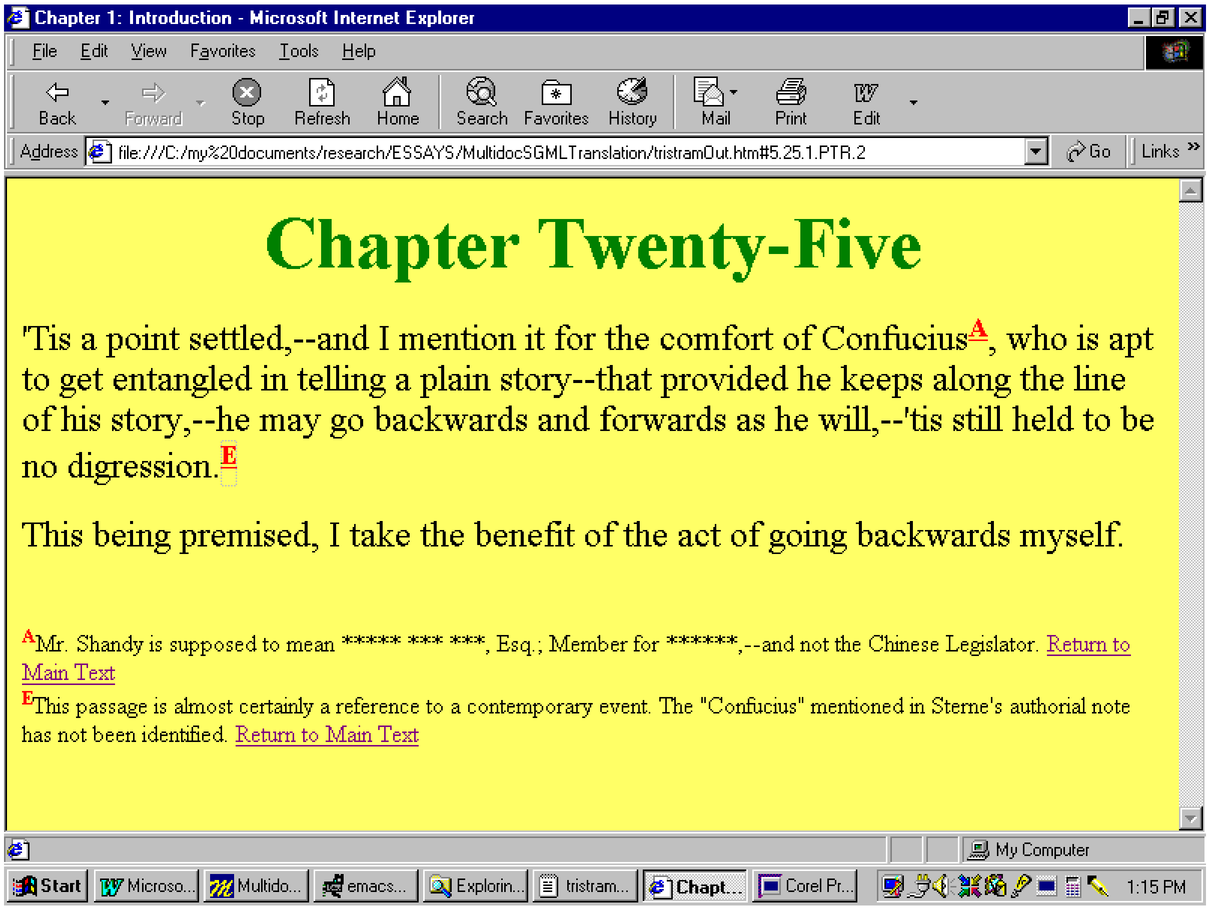
Before Text:

```
<A NAME="\att(id)" HREF="#\att(target)">
```

After Text:

```
<B><FONT SIZE=5
 COLOR="RED"><SUP><em>\ifatt(type="editorial")E</SUP></B></FONT></A>\e
 <pre>
```

Figure R: Conditionally Formatted Links as Displayed in Commercial HTML Browser


Chapter 1: Introduction - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address file:///C:/my%20documents/research/ESSAYS/MultidocSGMLTranslation/tristramOut.htm#5.25.1.PTR.2
Go Links
My Computer
Start Microso... Multido... emacs... Explorin... tristram... Chapt... Corel Pr... 1:15 PM

Chapter Twenty-Five

'Tis a point settled,--and I mention it for the comfort of Confucius[A], who is apt to get entangled in telling a plain story--that provided he keeps along the line of his story,--he may go backwards and forwards as he will,--'tis still held to be no digression.[E]

This being premised, I take the benefit of the act of going backwards myself.

[A]Mr. Shandy is supposed to mean ***** *** ***, Esq.; Member for ******,--and not the Chinese Legislator. Return to Main Text

[E]This passage is almost certainly a reference to a contemporary event. The "Confucius" mentioned in Sterne's authorial note has not been identified. Return to Main Text

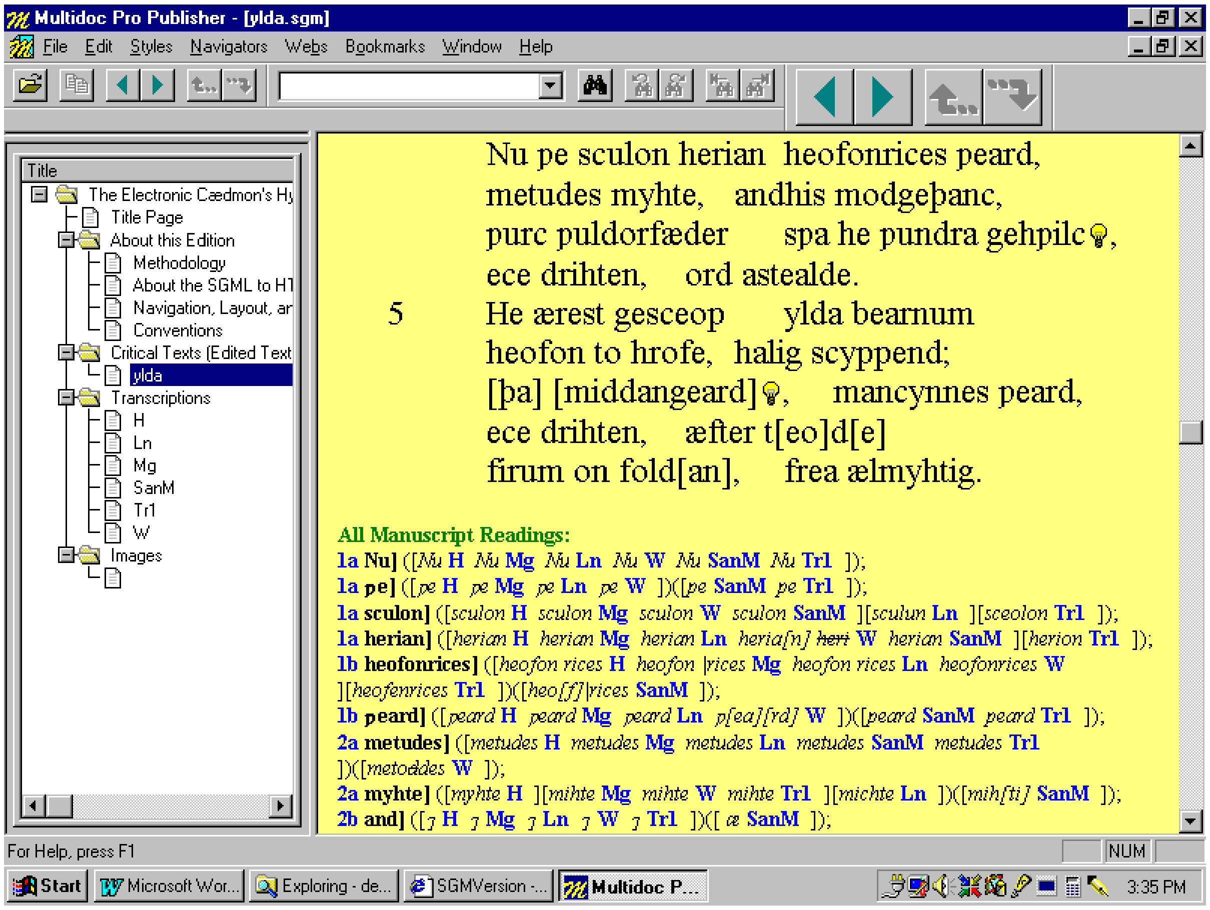
Pre-Filtering SGML Documents Using the "Extended" Tab

The Multidoc Style Sheet language allows for the use of only a limited set of conditional expressions, pertaining exclusively to attribute and environment variables, in the style sheet editor's "Before" and "After" tabs. A more complete set, including conditions that test for an element's location in the SGML document structure or arbitrary ATTREFs in neighboring, parent, or child elements is permitted under the style sheet editor's "Extended" tab. While the primary purpose of this function is to allow for complex SGML filtering operations, the expanded conditional language does have a role to play in SGML to HTML translation—particularly in its ability to produce multiple HTML views of a single SGML document.

An example of this can again be found in the *Electronic Cædmon's Hymn*. This edition constructs its textual apparatus from a hierarchically organized list of textual variants. These lists contain every single reading in the more than twenty medieval manuscripts known to have contained the poem, and run to tens of pages of typescript despite the fact that the poem itself is only nine lines long. In order to help readers make sense of this information, a number of pre-selected views have been created to filter the variant lists according to various standard criteria: in one view, users can see all manuscript forms arranged line by line; in another, a list of dialectal and orthographic variants; in a third, only those forms which appear in some way to have a significant affect on sense or meter.

Each of these views has a slightly different layout. In the "All Manuscript Readings" view, users are presented with a straight line-by-line listing of readings from the surviving manuscripts followed by their manuscript sigla (Figure S).

Figure S: "All Manuscript Readings" Apparatus View in Multidoc Browser

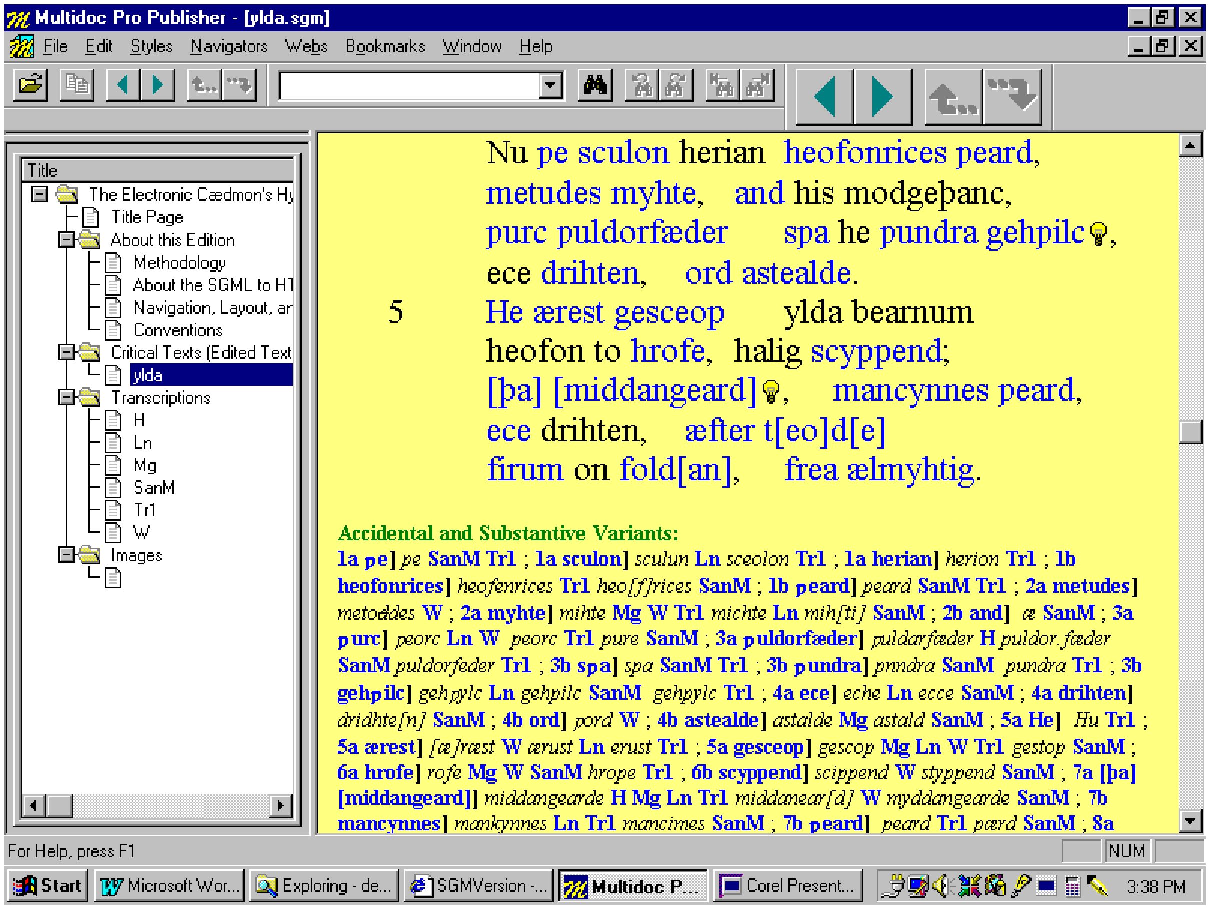


Nu pe sculon herian heofonrices peard,
metudes myhte, andhis modgeþanc,
purc puldorfæder spa he pundra gehpilc,
ece drihten, ord astealde.
5 He ærest gesceop ylda bearnum
heofon to hrofe, halig scyppend;
[þa] [middangeard], mancynnes peard,
ece drihten, æfter t[eo]d[e]
firum on fold[an], frea ælmyhtig.

All Manuscript Readings:
1a **Nu]** ([*Nu* **H** *Nu* **Mg** *Nu* **Ln** *Nu* **W** *Nu* **SanM** *Nu* **Trl**]);
1a **pe]** ([*pe* **H** *pe* **Mg** *pe* **Ln** *pe* **W**])([*pe* **SanM** *pe* **Trl**]);
1a **sculon]** ([*sculon* **H** *sculon* **Mg** *sculon* **W** *sculon* **SanM**][*sculun* **Ln**][*sceolon* **Trl**]);
1a **herian]** ([*herian* **H** *herian* **Mg** *herian* **Ln** *heria[n]* ~~*heri*~~ **W** *herian* **SanM**][*herion* **Trl**]);
1b **heofonrices]** ([*heofon rices* **H** *heofon |rices* **Mg** *heofon rices* **Ln** *heofonrices* **W**][*heofenrices* **Trl**])([*heo[f]|rices* **SanM**]);
1b **peard]** ([*peard* **H** *peard* **Mg** *peard* **Ln** *p[ea][rd]* **W**])([*peard* **SanM** *peard* **Trl**]);
2a **metudes]** ([*metudes* **H** *metudes* **Mg** *metudes* **Ln** *metudes* **SanM** *metudes* **Trl**])([*metoddes* **W**]);
2a **myhte]** ([*myhte* **H**][*mihte* **Mg** *mihte* **W** *mihte* **Trl**][*michte* **Ln**])([*mih[ti]* **SanM**]);
2b **and]** ([*7* **H** *7* **Mg** *7* **Ln** *7* **W** *7* **Trl**])([*æ* **SanM**]);

The remaining two views are lemmatized: the readings are sorted and filtered according to the relevant criteria and then formatted in such a way that only the first occurrence of each form is displayed in its entirety. Other witnesses containing the same form are represented by their sigla alone. [Figure T] shows the "Accidental and Substantive Variants" view as it is formatted for display in the Multidoc Browser:

Figure T: "Accidental and Substantive Variants" Apparatus View in Multidoc Browser



Nu pe sculon herian heofonrices peard,
metudes myhte, and his modgeþanc,
purc puldorfæder spa he pundra gehpilc,
ece drihten, ord astealde.
5 He ærest gesceop ylda bearnum
heofon to hrofe, halig scyppend;
[þa] [middangeard], mancynnes peard,
ece drihten, æfter t[eo]d[e]
firum on fold[an], frea ælmyhtig.

Accidental and Substantive Variants:
1a pe] *pe* **SanM Trl** ; **1a sculon]** *sculun* **Ln** *sceolon* **Trl** ; **1a herian]** *herion* **Trl** ; **1b heofonrices]** *heofenrices* **Trl** *heo[f]rices* **SanM** ; **1b peard]** *peard* **SanM Trl** ; **2a metudes]** *metoddes* **W** ; **2a myhte]** *mihte* **Mg W Trl** *michte* **Ln** *mih[ti]* **SanM** ; **2b and]** *æ* **SanM** ; **3a purc]** *peorc* **Ln W** *peorc* **Trl** *pure* **SanM** ; **3a puldorfæder]** *puldarfæder* **H** *puldor.fæder* **SanM** *puldorfeder* **Trl** ; **3b spa]** *spa* **SanM Trl** ; **3b pundra]** *pnndra* **SanM** *pundra* **Trl** ; **3b gehpilc]** *gehpylc* **Ln** *gehpilc* **SanM** *gehpylc* **Trl** ; **4a ece]** *eche* **Ln** *ecce* **SanM** ; **4a drihten]** *dridhte[n]* **SanM** ; **4b ord]** *pord* **W** ; **4b astealde]** *astalde* **Mg** *astald* **SanM** ; **5a He]** *Hu* **Trl** ; **5a ærest]** *[æ]ræst* **W** *ærust* **Ln** *erust* **Trl** ; **5a gesceop]** *gescop* **Mg Ln W Trl** *gestop* **SanM** ; **6a hrofe]** *rofe* **Mg W SanM** *hrope* **Trl** ; **6b scyppend]** *scippend* **W** *styppend* **SanM** ; **7a [þa] [middangeard]]** *middangearde* **H Mg Ln Trl** *middanear[d]* **W** *myddangearde* **SanM** ; **7b mancynnes]** *mankynnes* **Ln Trl** *mancimes* **SanM** ; **7b peard]** *peard* **Trl** *pærd* **SanM** ; **8a**

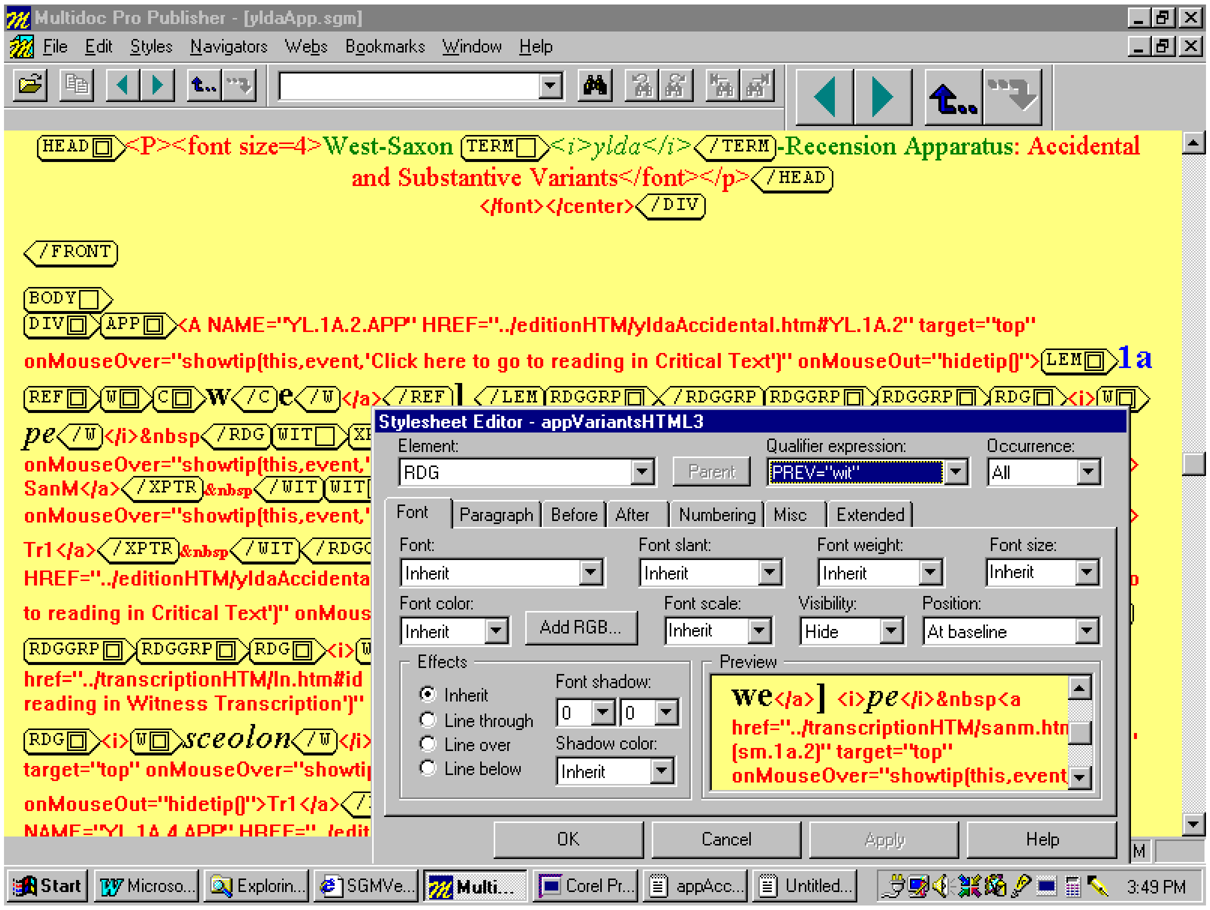
All views are based on the same complete listing of manuscript variants. Their differences are attributable solely to the filtering instructions that allow each style sheet to distinguish among otherwise identical SGML elements on the basis of attribute values or contextual position. The variant lists consist of readings (`<RDG>`) grouped according to various criteria into different reading groups (`<RDGGRP>`). These are in turn assigned to specific readings from the main text (represented by the element `<LEM>`) in an `<APP>` entry (A sample entry is reproduced below, Figure U). The level of abstraction involved in any given list—that is to say whether the apparatus shows all readings, accidental and substantive variants, or substantive variants only—is controlled by a conditional test of the attribute `TYPE` on the SGML element `<RDGGRP>`. The decision to display or suppress individual variants within a selected `<RDGGRP>` is then made on basis of an individual instance's contextual position in the list: all views except for "All Manuscript Readings" display the first reading and sigil in each `<RDGGRP>` and suppress all subsequent forms, representing them instead by their manuscript sigla alone:

Figure U: Simplified Fragment from SGML Apparatus File for the Electronic Cædmon's Hymn (non-essential elements and ATTREFs have been removed)

```
<APP TYPE="substantive" FROM="yl.1a.2">
<LEM>
<W>&wynn;e
</W>
</LEM>
<RGDGRP TYPE="form">
<RGDGRP TYPE="spelling">
<RDG>
<W>pe
</W>
</RDG>
<WIT>SanM
</WIT>
<RDG>
<W>pe
</W>
</RDG>
<WIT>Trl
</WIT>
</RDGGRP>
</RDGGRP>
</APP>
```

Views such as these cannot be translated into HTML without pre-filtration. Because they are based on a test of elements' relative position in the document structure, they require access to the SGML source document's DTD (Document Type Definition). Interestingly, this makes such views actually easier to translate than examples requiring multiple, context-sensitive HTML encodings for a single element. Because the SGML in this example has been pre-filtered before it is translated to HTML, developers are required only to "shadow" the SGML elements they see in HTML. Figure V shows an example of the style sheet used for translating the "Accidental Variant" view to HTML; Figure W, the final HTML output:

Figure V: "Accidental and Substantive Variants" Translation Style Sheet


Multidoc Pro Publisher - [yldaApp.sgm]
File Edit Styles Navigators Webs Bookmarks Window Help
Stylesheet Editor - appVariantsHTML3
OK
Cancel
Apply
Help
Start
3:49 PM

HEAD <P><font size=4>West-Saxon TERM <i>ylda</i> /TERM -Recension Apparatus: Accidental and Substantive Variants</font></p> /HEAD
</font></center> /DIV

/FRONT

BODY
DIV APP <A NAME="YL.1A.2.APP" HREF="../editionHTM/yldaAccidental.htm#YL.1A.2" target="top"
onMouseOver="showtip(this,event,'Click here to go to reading in Critical Text')" onMouseOut="hidetip()"> LEM 1a
REF W C w /C e /W </a> /REF] /LEM RDGGRP /RDGGRP RDGGRP RDGGRP RDG <i> W
pe /W </i> /RDG WIT

Element: RDG | Parent | Qualifier expression: PREV="wit" | Occurrence: All

Font | Paragraph | Before | After | Numbering | Misc | Extended

Font:	Font slant:	Font weight:	Font size:
Inherit	Inherit	Inherit	Inherit

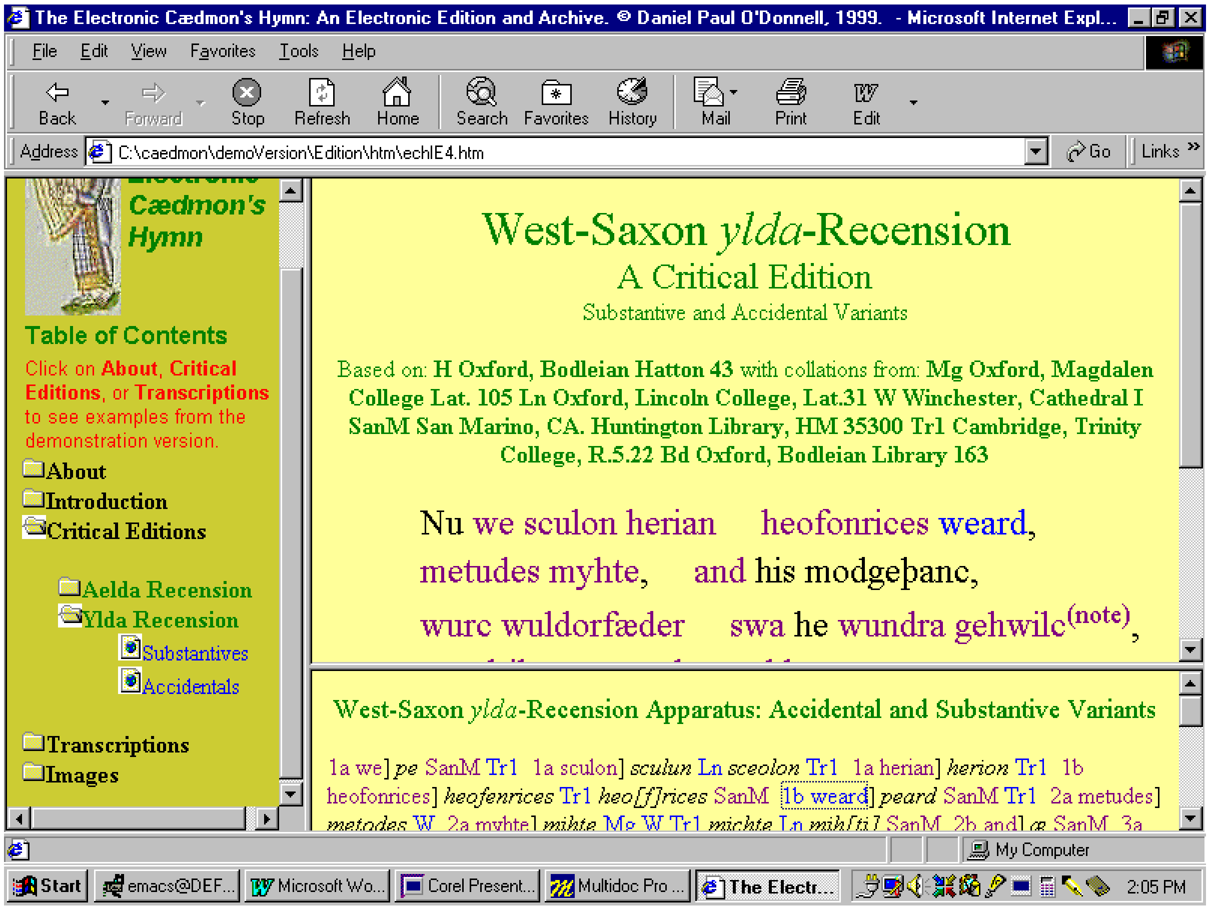
Font color:	Font scale:	Visibility:	Position:
Inherit	Inherit	Hide	At baseline

Effects: [✓ Inherit] Line through Line over Line below

Font shadow: 0 0 — Shadow color: Inherit

Preview:
we</a>] <i>pe</i> <a href="../transcriptionHTM/sanm.htm (sm.1a.2)" target="top" onMouseOver="showtip(this,event

Figure W: "Accidental and Substantive Variants" View in Commercial HTML Browser (Bottom Right Frame)


The Electronic Cædmon's Hymn: An Electronic Edition and Archive. © Daniel Paul O'Donnell, 1999. - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address C:\caedmon\demoVersion\Edition\htm\echlE4.htm
Go Links
My Computer
Start
2:05 PM

Cædmon's Hymn

Table of Contents

Click on About, Critical Editions, or Transcriptions to see examples from the demonstration version.

- About
- Introduction
- Critical Editions
  - Aelda Recension
  - Ylda Recension
    - Substantives
    - Accidentals
- Transcriptions
- Images

West-Saxon *ylda*-Recension
A Critical Edition
Substantive and Accidental Variants

Based on: H Oxford, Bodleian Hatton 43 with collations from: Mg Oxford, Magdalen College Lat. 105 Ln Oxford, Lincoln College, Lat.31 W Winchester, Cathedral I SanM San Marino, CA. Huntington Library, HM 35300 Tr1 Cambridge, Trinity College, R.5.22 Bd Oxford, Bodleian Library 163

Nu we sculon herian heofonrices weard,
metudes myhte, and his modgeþanc,
wurc wuldorfæder swa he wundra gehwilc(note),

West-Saxon *ylda*-Recension Apparatus: Accidental and Substantive Variants

1a we] *pe* SanM Tr1 1a sculon] *sculun* Ln *sceolon* Tr1 1a herian] *herion* Tr1 1b heofonrices] *heofenrices* Tr1 *heo[f]rices* SanM 1b weard] *peard* SanM Tr1 2a metudes] *metodes* W 2a myhte] *mihte* Mg W Tr1 *michte* Ln *mih[t]i* SanM 2b and] *æ* SanM 3a

Implementation Problems

The method outlined in this practice note for translating from SGML to HTML is quick, effective, and easy to learn. It provides structural document designers with a tool that allows them to produce accessible versions of their projects on short notice and with a minimal investment. It is, however, not a panacea for all SGML display problems. Like all languages and processes, the method performs some tasks better than others. Of its weaknesses, three in particular deserve special comment:

1. The Multidoc Style Sheet Language is poor at extraction
2. There appears to be no way of processing mathematical conditions
3. The method's requirement that HTML coding be entered by hand brings with it an inherent possibility of error

Of these, the first and third can at least partially be overcome with appropriate pre- or post-processing of the source or output document. The second poses a more serious problem—but one that must be weighed against the procedure's otherwise great versatility.

Extraction

One of the most obvious disadvantages of the Multidoc Style Sheet language lies in its weak extraction capabilities. While developers can use variables like `\att` and `\ifatt` to extract and manipulate certain kinds of SGML information, the amount and types of material which can be manipulated in this fashion is extremely limited. Style sheets cannot, for example, be used to rearrange the order in which SGML elements appear in the document itself. While attribute information can be extracted from SGML attributes, the language does not allow partial or regular expression extraction. This means that even some relatively common operations, such as the translation of TEI-style extended pointer syntax, may require post-processing before the HTML output can be used in a commercial browser. This processing can range from applying a simple search-and-replace routine to the output document to more complicated operations.

Mathematical Conditions

A second problem in using Multidoc Style Sheets for translating SGML to HTML lies in the language's complete inability to apply mathematical filters and conditions. This means, among other things, that it is impossible to automate such common tasks as printing poetic line numbers at arbitrary intervals in larger texts (the line numbers in Figure I and Figure S are extracted using a conditional expression that simply tests whether value of `N` on the SGML element `<L>` is "5"; this would be too unwieldy to use with even a medium-sized poem like *Beowulf* [±3100 lines]). Projects for which such operations are essential will almost certainly need to use a post-processor.

Potential for HTML Coding Errors

A final problem involves the inherent potential for error that exists in entering long or repetitive sequences of HTML coding into the "Before" and "After" tabs. While formatting SGML documents for display in Multidoc is a wizard-based and graphical process, "shadowing" this original mark-up with HTML coding is done by hand—a tedious operation that leaves considerable room for error.

Perhaps the easiest way of solving this problem involves working with the style sheet directly. Because Multidoc Style Sheets are SGML documents, `<STYLE>` elements with complicated HTML coding can be cut and pasted quite easily from one to another. In translating the *Electronic Cædmon's Hymn*—where several different HTML views are provided for each SGML document—we found ourselves relying on this method almost exclusively for enforcing consistency in our standard but relatively complicated HTML document headers.

Another possible solution involves defining particularly complex HTML coding sequences as SGML entities in one of the two Style Sheet Entity Files supplied with the program (`../catalog/Sheet.ent` and `../catalog/Sheet2.ent`—). This method has the advantage of modularity: complex fragments of code—scripts, attribute translations, or long sequences of conditions, for example—can be pre-defined as style sheet entities in the file `Sheet.ent` and `Sheet2.ent` and subsequently entered into the style sheet editor by their entity names. When the source document is displayed, these entities are replaced by the code to which they refer—allowing the users to enter repetitive or long HTML coding fragments over and over again with a minimum number of keystrokes (The editor interface handles entity references in an interesting way: it expands them correctly when they are first entered into the style sheet, but replaces them permanently with the entity value when the style sheet is re-edited; if the sheet is not 1re-edited, the entity name remains in the style sheet).1

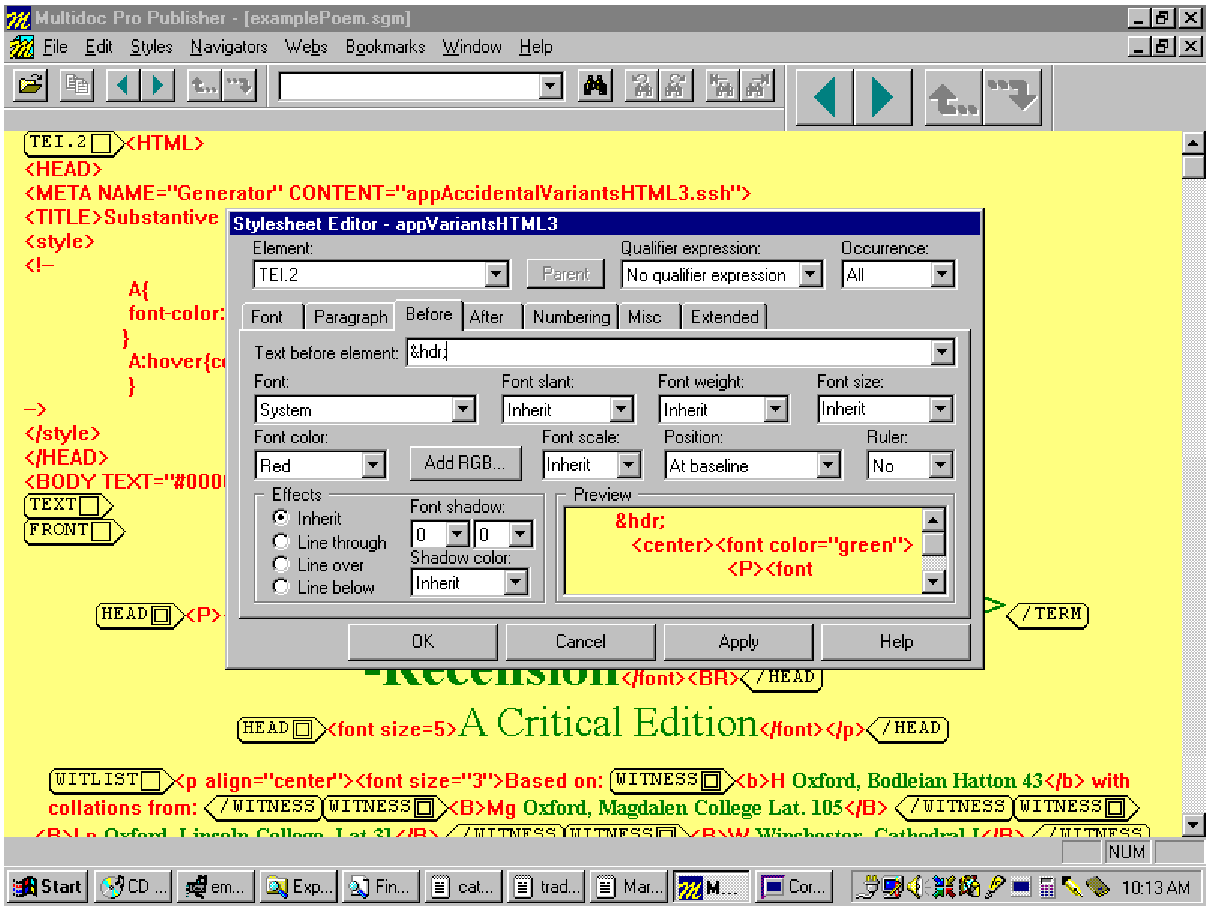
An example of how this can work is given in Figures X through Y. Standardised HTML header information is defined as an entity `&hdr;` in `Style.ent` or `Style2.ent` (Figure X).

Figure X: Defining HTML Header Information as Entity in Multidoc Style Sheet Entity File `Sheet2.ent` (Entity Emphasised)

```
<!-- -//Synex Information AB//DTD Stylesheet v2.00//EN -->
<!ENTITY % ISOlatin1 PUBLIC "ISO 8879-1986//ENTITIES Added Latin
1//EN">
%ISOlatin1;
<!ENTITY q "'" >
<!ENTITY Q '"' >
<!-- The following entities are user defined HTML codings --> <em>
<!ENTITY hdr '<HTML>\n;<HEAD>\n;<META NAME="Generator"
CONTENT="appAccidentalVariantsHTML3.ssh">\n;<TITLE>Substantive
Variants</TITLE>\n;<style>\n;<!--\n;\tab\tab A{\n;\tab\tab
font-color:blue ; text-decoration:none\n;\tab\tab}\n;\tab\tab
A:hover{color:red; text-decoration:underline;\n;\tab\tab
}\n;-->\n;</style>\n;</HEAD>\n;<BODY TEXT="#000000" LINK="#0000ff"
VLINK="#800080" BGCOLOR="#ffff80">' ></em>
```

When the entity name is entered as "Before" text to the SGML root element (in this case `<TEI.2>`) it is automatically replaced in the display by the text defined in the style sheet entity file (Figure Y). When the document is copyed, pasted into a text processor and saved as an HTML file, the entity-defined coding functions just like the hand-entered tagging in the document.

Figure Y: Entering Entity Defining HTML Header Coding in Style Sheet Editor



Multidoc Pro Publisher - [examplePoem.sgm]

```
TEI.2 <HTML>
<HEAD>
<META NAME="Generator" CONTENT="appAccidentalVariantsHTML3.ssh">
<TITLE>Substantive
<style>
<!–
        A{
        font-color:
        }
        A:hover{c
        }
–>
</style>
</HEAD>
<BODY TEXT="#000
TEXT
FRONT
```

Stylesheet Editor - appVariantsHTML3

Element: TEI.2 — Qualifier expression: No qualifier expression — Occurrence: All

Before tab — Text before element: &hdr;

Font: System — Font slant: Inherit — Font weight: Inherit — Font size: Inherit

Font color: Red — Font scale: Inherit — Position: At baseline — Ruler: No

Effects: (●) Inherit — Font shadow: 0, 0 — Shadow color: Inherit

Preview:

```
&hdr;
  <center><font color="green">
      <P><font
```

```
HEAD <P>                                              /TERM
-Recension</font><BR> /HEAD
HEAD <font size=5>A Critical Edition</font></p> /HEAD
WITLIST <p align="center"><font size="3">Based on: WITNESS <b>H Oxford, Bodleian Hatton 43</b> with
collections from: /WITNESS WITNESS <B>Mg Oxford, Magdalen College Lat. 105</B> /WITNESS WITNESS
<B>Ln Oxford, Lincoln College, Lat 31</B> /WITNESS WITNESS <B>W Winchester, Cathedral I</B> /WITNESS
```

Conclusion

Structurally encoded SGML mark-up languages offer document designers great processing flexibility. A single document instance can be reused easily in a variety of different contexts and for a variety of different audiences. But this flexibility comes at the cost of spontaneity. Where designers working directly in popular layout-oriented languages like HTML can distribute their documents without further processing, developers of structural projects must always consider how end users are going to access their material. Currently such access requires either the purchase of specialised display applications or the development of customized translation scripts. Both methods can bring with them significant cost and inconvenience—especially for smaller non-profit projects where funding and developer's time may be equally scarce.

The method proposed in this practice note for SGML to HTML translation offers a partial solution to this problem. While developers still need to translate their structurally encoded SGML documents to HTML in order to display them, this translation is accomplished in a way that considerably reduces the demands on developers' time and expertise. Translators work from an already marked up SGML style sheet that they "shadow" with HTML tags. They use an editing interface that is graphically oriented and allows them to check their work on-screen. Except in the most complex cases, moreover, the skills required are those likely to be possessed by any project member with a good knowledge of HTML and some experience in hand coding. Indeed developers might even find themselves able to harness the aesthetic sophistication of "the average high school senior's web page" by hiring high school seniors to do their SGML to HTML translation!

Notes

1I thank Mårten Stöm of Citec for his help in clarifying this aspect of style sheet entity operations.

Work Cited

Association for Computers and the Humanities (ACH)/Association for Computational Linguistics (ACL)/Association for Literary and Linguistic Computing (ALLC) Text Encoding Initiative, C.M. Sperberg-McQueen, and Lou Burnard. *Guidelines for Electronic Text Encoding and Interchange (P3)* .