INTENSIONAL HTML

BILL WADGE and TANER YILDIRIM
Department of Computer Science, University of Victoria
Victoria, B.C. V8W 3P6, Canada
E-mail: wwadge@cs.uvic.ca, taner@gulf.uvic.ca

ABSTRACT

Intensional HTML is a high-level Web authoring language that makes practical (using standard client and server software) specification of pages and sites that exist in many different versions or variants. Each page of IHTML defines an intension - an indexed family of actual (extensional) HTML pages which varies over a multi-dimensional author-specified version space. The version space is partially ordered by a refinement/specialization ordering. Authors can create multiple labeled versions of the IHTML source for a given page. Requests from clients specify both a page and a version, and the server software selects the appropriate source page and uses it to generate the requested actual HTML page. Authors do not, however, have to provide separate source for each version. If the server software cannot find a source page with the exact version requested, it uses the page whose label most closely approximates the requested version. In other words, it treats the refinement ordering as a (reverse) inheritance ordering. Thus different versions can share source, and authors can write generic, multi-version code.

1. Introduction

1.1. The Versioning Phenomenon

Almost all artifacts produced by human kind - whether ships or shirts, cars or carnations - are produced in groups of versions or variants. Each member of the group is specially adapted to a particular use. Yet they all share a family resemblance, and as a result are easier to produce and use than a collection of unrelated individuals.

This phenomenon is especially noticeable with computer software. Partly, this is because software is so easy to copy and modify. Also, the rapid advance of computer technology creates new needs and opportunities, which are usually more easily addressed by modifying existing products, rather than creating new ones.

The creation of the World Wide Web has produced another variant of the software version problem. As the Web grows, site designers come under increasing pressure to provide multi-version sites, for a number of reasons. Firstly, the Web is international, and a truly international site must be available in many different languages. The bandwidth available to users varies greatly, so that some appreciate high quality graphics while others prefer purely text pages. Different browsers have different capabilities, for example, in terms of tables and frames. Some sites offer more material to paying subscribers while others may want certain information hidden from outsiders. Some sites would naturally offer different information (e.g. weather reports) to people in different parts of the world. Finally, site designers might want to offer sites that are customizable to take personal preferences (background colors/patterns, text and hyperlink colors) into account.

1.2. The Problem with Cloning

The problems encountered in producing and managing software versions arise with Web documents as well.

The easiest way to produce a variant of a web site is to make a copy of the HTML source and modify it. Unfortunately, the cloning (copying/modifying) approach to version creation can, in the long run, produce severe difficulties maintaining the resulting families of versions. The problems arise when changes are required in parts of the original that were copied unchanged into the new versions. The same changes have to be made many times, in the sources for all the versions which used the original code.

The inevitable result of the copy-and-modify approach is a large family of clones which, as a group,
almost impossible to change in any uniform way.

The obvious solution is to ensure that the different members of the family share the same code (and not copies thereof) so that necessary changes are made only once and need not be propagated. To use Ted Nelson's terminology, we must arrange that different versions transclude the source they have in common.

This is easier said than done; in fact software version management is one of the most difficult problems in software engineering. Indeed, the success of the object-oriented approach is due in part to class inheritance, which allows code to be shared (reused) on a large scale.

HTML certainly allows two different pages to link to a third, but this is a very crude form of sharing. Links are essentially pointers, and the problem with sharing by pointers is that you also share everything the shared object itself has pointers to.

Consider the problem of supporting English and French versions of a simple slide show. The slide show consists of a sequence of pages of text and/or graphics, each linked to the next page in the sequence. Obviously, we have to create separate English and French versions of pages with text on them. But we also have to make clones of any pages that have only images, even though the French and English versions will appear identical on the screen. The problem is that the English version of the page in question must be linked to the rest of the English version of the show, while the French version of the slide must be linked to the rest of the French version. Two separate source files are required which differ only in the hyperlinks to the next slides.

2. The IHTML Solution

In this paper we present IHTML (Intensional HTML), an HTML-based authoring language which incorporates an OO (inheritance based) approach to hypertext versioning. IHTML allows authors to define, with a single source file, a whole indexed family of HTML variants based on the file in question. These variants are generated on demand, then discarded after use. In a sense, IHTML automates the cloning process, and eliminates the maintenance problem by ensuring that the clones are short lived.

IHTML is intensional because IHTML source has both intensional and extensional meanings. The intension is the whole indexed family of HTML pages; the extensions are the different individual HTML pages.

The main feature of IHTML is that authors can provide multiple sources for the same page, each source labeled with a different version. The IHTML server software accepts requests for particular versions of particular pages, and generates the actual HTML from the appropriate IHTML source files.

For example, in the case of the slide show, the author could name the pages slide1, slide2, slide3, ... and provide for each of these pages two source files, one English and one French.

IHTML authors do not, however, have to provide separate source files for every possible version. The IHTML index space ("version space") is partially ordered by a refinement relation, and the source for a more refined version is by default inherited from the less refined (more generic versions).

When the IHTML server software receives a request for a particular version of a particular page (or part thereof), it looks for a source file labeled with the requested version. If there is no such source, it looks for a source file whose label most closely approximates the requested version (if there is no such source file, or no best source file, it reports an error). More refined versions can therefore by default transclude source from more generic ones, and a relatively small number of source files can define a very large family of pages.

For example, suppose that the fifth slide is purely graphical. The author can provide a single source file, labeled as the standard (so-called vanilla) version. When a request comes for, say, the French version of page slide5, the server-side software first looks for a source file for that page labeled as the French version. When it finds none, uses the more general standard version. Requests for the English version of slide5 are similarly referred to the single standard source.

2.1. Intensional Elements

The intensional elements of IHTML are links, includes and images. Intensional elements in HTML
source look exactly like the corresponding elements of HTML. IHTML links are interpreted, however, as denoting a whole family of links, each connecting a given version of the page they appear in to the corresponding version of the page they link to. Similarly, IHTML image elements represent a family of image elements, each including the corresponding version of the image they point to. The corresponding versions can be more generic, even vanilla, versions when the target page or the image does not exist in the 'current' version, i.e. the version requested of the file in which the element appears.

For example, suppose that the generic IHTML source for slide5 contains the link \texttt{<a href=page5>...</a> and the image element \texttt{<img src=image5>}. The link is interpreted as meaning that the English version of page5 is linked to the English version of page6, and that the French version of page5 is linked to the French version of page6. When the server-side software generates the French version of page5 from the generic source, it makes the generic link into a link to the French version of page6 (this might be the only change made). In case of the image element, the server-side software uses the French version of image5 to generate French version of page5 and English version of image5 to generate the English version of the same page.

IHTML includes represent a family of include directives, each incorporating (by copying) the contents of the corresponding version of the source file they point to. When processing an include directive, the server-side software looks for the version of the named source file whose version label most closely approximates the current version.

For example, each page of the slide show might have \texttt{<include file=header> at the top and <include file=footer> at the bottom and the English version of page4 will include the English version of the header and the footer.}

The include facility is very important for IHTML because it allows authors to break the source components into pieces smaller than a whole page. This allows authors to isolate the parts of a page that actually vary, and write more generic source for the parts (such as headers and footers) that do not.

2.2. The IHTML Version Space

The families of pages specified by IHTML are indexed by (subspaces of) an algebraically defined version space. In the terminology of intensional logic, the elements of this space are possible worlds; each individual possible world (version) determines a particular extension, i.e. an actual HTML page.

The elements of this version space are expressions built up from identifiers using the operations + and %.

The % operator is the subversion operator: \( V\%w \) is (by definition) a refinement of \( V \). For example, \texttt{Mac68} is a subversion of Mac.

The + operator is the version join operator - the least upper bound in the refinement ordering. Intuitively, version \( V + W \) is the most general version which incorporates the modifications/refinements of both versions \( V \) and \( W \). For example, the \texttt{Mac68+french} version might be the version which is designed for 68K Macs and uses French as its interface language.

These ideas are formalized by the following axioms:

\[
V \subseteq V\%V', \quad V\%W \equiv V', \quad (V\%V')\%W' \equiv V\% (V'\%W')
\]

\[
V \subseteq V + V', \quad (V_1 \subseteq V_1') \quad (V_2 \subseteq V_2') \quad \frac{V_1 + V_2 \subseteq V_1' + V_2'}{V + V' \equiv V},
\]

\[
V + V' \equiv V' + V, \quad (V + V')' \equiv V + (V' + V'), \quad V\%V' + V\%V'' \equiv V\% (V' + V'').
\]

where \( \subseteq \) is the refinement operator and \( \equiv \) represents the most general version - vanilla version.

The elements of this version space are equivalence classes of expressions together with the coarsest
order which satisfies the axioms. This space is similar to Prolog's set of Herbrand terms - a convenient collection of abstract symbolic objects to which we can attach meanings.

The IHTML version space extends this version space in one important way: it allows explicit dimensions.

For example, we interpreted the term french in the above expression as referring to the interface language. What if we were producing information about cooking and also needed to specify the cuisine? In the IHTML space, we can use arbitrary identifiers as dimensional 'multipliers' and form sums that specify coordinates for each of the given dimensions. This enlarged space includes expressions such as

\[
\text{platform:MacK58 + language:french + cuisine:chinese}
\]

The extra rules are:

\[
\begin{align*}
D:\varepsilon & = \varepsilon \\
D: (V + V') & = D: V + D: V' \\
D: V \subseteq E: V' & \iff D = \varepsilon \text{ and } V \subseteq V'.
\end{align*}
\]

It should be clear now how to compare two dimension sums. In general

\[
D_0: V_0 + D_1: V_1 + \ldots \subseteq E_0: W_0 + E_1: W_1 + \ldots
\]

if and only if each \( D_i \) is equal to \( E_j \) for some \( j \) and in each such case \( V_i \subseteq W_j \). (We assume the \( D \)'s and \( E \)'s are distinct, ordered in dictionary order and that no \( V_i \) is \( \varepsilon \).)

\[2.3 \text{ Transversion Elements}\]

The second distinguishing feature of IHTML is the ability to define what we call transversion elements. These elements are intensional elements where the corresponding versions of their target sources are different from the current version.

A transversion link has the same format as an intensional link, except that the element may contain assignments to dimension identifiers. The link is interpreted as leading from a given version of the source page to the modified version of the target page - the modifications resulting from altering the coordinates of the given dimensions as specified.

For example, the author of the slide show might include, in the English versions of the source of the title page, a link of the form \(<a \: href=\text{page1} \: language=\text{french}>\) In the (say) language:english+background:blue version of the title page, this will be interpreted as a link to the language:french+background:blue version of the title page.

Notice that following this link will take the reader to the French version of the whole slide show. The reason is that the French version of page1 is linked to the French version of page2, and so on. The English and French versions coexist as sort of parallel universes, and the transversion links let the reader move from one of these universes to the other, without necessarily filling in forms or composing complex URLs. At the same time, they give the author full control over way in which different versions of the site are interconnected.

Transversion includes and image elements are also distinguished from their intensional correspondents by dimension identifier modifiers. For example, \(<\text{include file=footer language=english}>\) and \(<\text{img src=welcome.gif language=english}>\) will include the footer and the welcome image in a version like the current version except that the language component is english.

Transversion includes and image elements, however, differ from transversion links in the scope of the modifications to the current version. These elements include the modified version of their target source, however the current version of the site remains unchanged. However, activating a transversion link changes
the current version of the whole site.

IHTML also allows links to conventional (unversioned) HTML webware. We call these extensional (or external) elements and they are like HTML elements except that href and src are replaced with xhref and xsrc, respectively. Extensional includes are not implemented, since it only makes sense to include server-side files. Furthermore, unversioned files can be included by intensional includes if the only existing file is stored as the vanilla version of that file. (One could think of an extensional element as a transversion element in which all the coordinates are set to e.)

3. Implementation

It might seem from what has been said that IHTML requires its own version of the server and client software. In the long run, this might indeed be the best way to implement IHTML. However, in the meantime, we can use standard techniques to allow existing clients to browse multi-version IHTML sites published by existing servers.

The requests from the client consist of an URL and a version. Satisfying such a request involves searching for the appropriate IHTML source file and then transforming the generic source into the particular HTML corresponding to the version in question.

The basic idea is to ensure that all links to an IHTML-specified page go through a cgi script. The call to the script has (at least) two arguments, the generic URL and (a representation of) the particular version requested. The cgi script invokes the server-side software which locates the appropriate source file and produces the HTML.

The cgi script itself ensures that all links in IHTML source are cgi calls. When the script generates HTML from IHTML, it transforms the normal-looking generic IHTML links into cgi calls with the appropriate parameters. The first parameter, the URL, is taken directly from the IHTML source of the link. The second parameter, a version, is normally the same as the version included in the client request. In the case of a transversion element, however, the software modifies this version according to the information in the IHTML source of the element.

For example, if a request comes for the language=french version of slide5, the link `<a href=page6>` becomes

`<a href=cgi-bin/scan.cgi/page6,v>`

and the link `<a href=page6 language=english>` would become

`<a href=cgi-bin/scan.cgi/page6,v,M>`

where, V is the integer representation of the current version and M is the integer representation of the modification (language=english). The server-side software uses coding schemes for version expressions and dimension identifier modifiers. The scheme for version expressions, which is implemented using a variant of the old hash-consing technique, assigns unique small integer codes to version expressions. The dimension identifier modifiers are stored in an array and their locations in the array are used as their integer codes. This keeps the requests and HTML pages short, avoids restrictions on the symbols which can appear in version expressions and dimension modifiers, and also provides a measure of security. Readers cannot jump into an arbitrarily chosen version of the site. Instead, the author, using only transversion links, controls the access of the readers to the different versions of the site.

IHTML source files are stored in directories, each directory containing all the source files for a particular intensional page or image. Most of these pages include a header, a body and a footer, and in most cases the body page is represented as a sub-directory. Each file name has the form pagename.mnn.ihtml or imagename.mnn.[gif|jpg] where pagename and imagename are the generic names of intensional pages and images which are used as the target values of intensional elements. The integer codes embedded in file names (mnn) represent the version labels of the files.

An intensional image like `<img src=image6.jpg>` would become:

`<img src=image/image6/image6.V.jpg>`
where $V'$ is the integer code for the version that is most relevant to the current version and \textit{image} is the directory that contains all the images of the site. Transversion images are transformed in a similar way except the fact that $V'$ would be the most relevant version to the modified current version.

Intensional and transversion includes are handled by invoking the server-side software which finds the most relevant include file and runs the \texttt{cgi} script which in turn processes the IHTML template(s) and replaces the include directive with the HTML contents of the source file(s). Include files can also include other files and there is no limitation on the number of nested includes.

Extensional links and images in IHTML source files are transformed to conventional HTML links and images by replacing \texttt{xhref} and \texttt{xsrc} with \texttt{href} and \texttt{src}, respectively.

The \texttt{cgi} script is implemented in Perl (369 lines) and the server-side software in C (3000 lines). We chose Perl for its portability and powerful string manipulation operators and C for its powerful pointers and data structures.

4. An Intensional Site Example and Comparisons

The second author has produced a multi-version home site using IHTML features available at URL http://lucy.uvic.ca/~tamer/cgi-bin/scan.cgi.

At first sight, it looks like a fairly normal home page. One can follow links to related pages with Mr Yildirim's biographical details, resume, course work, and favorite bands and beers.

However, at the bottom of each of these pages is a link anchored to the phrase \textit{Turkish version of this site}. When we click it, the text on the page changes from English to Turkish.

The words are well chosen: if we proceed to explore the site again, we find the resume, the course work, the bands and so on, but all these pages are in Turkish. We are now in the Turkish version of the whole site, not just of the home page. At the bottom there is an anchored phrase containing the words \textit{ingilisce versiyonu} and, not surprisingly, it leads us back to the English version of the site. Each page also offers us a similar transversion link to the text only version (actually, versions) of the site.

Finally, all the pages have links to an \textit{options} page. These links take us to pages which contain the options for changing the version of the site. In \texttt{display: graphics} version of this page thirty small anchored images - background colors/patterns are displayed. Following, say, the link anchored to the orange square takes us to the home page of a version of the site in which all the pages (including the options page) have orange backgrounds. The \texttt{display: text} version of the options page contains only six links (instead of images) to change the background color of the site. Similarly, there are twenty-seven options for the color of the text and the color of the hyperlinks of the site. These options are grouped in the same page since it is guaranteed that all the pages of the site exist in all the possible extensions. This is simply, because the IHTML source files have the following body tags:

\begin{verbatim}
<body <include file=background> <include file=text> <include file=link> ... />
\end{verbatim}

These tags include the current version of the background, text and link files which contain assignments to textcolor/\textit{background}, \textit{text} and \textit{link} attributes.

Note that we can view the different versions of the site with a standard browser, just by following links, without filling in forms or otherwise composing complex URLs. In fact, we do not need to know anything about the version algebra.

The site described above uses a tiny five-dimensional subset of the version space. The dimensions are \textit{language}, \textit{display}, \textit{background}, \textit{text}, \textit{link}. The language and display dimensions have two coordinates each; \textit{english}, \textit{turkish} and \textit{text}, \textit{graphics}, respectively. Background dimension has thirty and the last two dimensions have twenty seven coordinates. The site consists of thirteen intensional pages each available in all versions - a total of 1,137,240 virtual HTML pages.

The original one-version HTML program for the site consisted of about 38K bytes divided into 13 different files. The IHTML source consists of 184 files, but most of them, 86, are tiny files - consisting of single lines (like \texttt{background=\textit{image} \texttt{sky.png}}, \texttt{text=":FF00FF"}, \texttt{link=":0000FF"} etc.). There are a total of 107K bytes of IHTML source - and most of the extra is Turkish versions of the original English text. These 107K bytes of IHTML source supports a virtual site which would correspond 3.3M bytes of cloned HTML.
The IHTML version is much more modular. For example, there is exactly one file containing the list of beers, labeled with the standard (vanilla) version. However, some of the movies have alternate Turkish titles, so there are two source files for this information, labeled with integer codes for language:english and language: turkish. Adding extra dimensions (say, for fonts or active, visited link colors) would involve minimal changes, mainly creating labeled include files each with appropriate font or color code.

However, for each activated link, the server-side software must find the most relevant IHTML source file, and the cgi script must parse and transform it into HTML which represents a performance hit that clients must suffer.

5. Conclusions

It should be clear that in IHTML we have incorporated a fundamental concept of OO - inheritance - in order to allow websters to write generic code which applies to many versions of a page. Version inheritance helps IHTML authors reuse code in much the same way that class inheritance allows OO programmers to reuse executable source.

However, dataflow - in particular, dynamic (tagged), demand-driven dataflow is a better paradigm to assign to IHTML. The demands come in from readers following links; each demand is for a particular page together with a particular version (the version is the tag). This demand can (when there are includes) generate demands for other pages, possibly with different version tags. Eventually all the sub-demands are met and the required HTML is returned to the reader. The (intensional) pages correspond to the dataflow program variables, and the IHTML source files play the role of the defining expressions. The crucial difference is that the same variable (page) can have several defining expressions, with the choice of the relevant expression determined by the tag field of the demand.

The dimension identifiers used in IHTML version space are customizable, allowing websters to apply the intensional approach to variation described in this paper to any indexable family of pages even if the pages don’t seem to exist in different versions. This allows a wide variety of possible applications for IHTML.

For example, the slides in the slide show are obviously indexed by the natural numbers. We can therefore consider them to be variants of 'the' slide, and add a page_number dimension to our versions space. This would allow us to have a single generic IHTML source for all the slides, which would have headers, footers, logos, color choices etc. The generic slide page would include a body file, which itself would vary over the page_number dimension. If we have a number of different slide presentations, we could in turn consider them to be versions of 'the' presentation, and write an even more generic page for all our presentations.

A page which changes every day, e.g. that of a newspaper, can clearly be indexed by the set of dates. If we add a page dimension, we can write generic source which specifies parts of the layout (such as mastheads) that are invariant. We can extend our scheme by allowing source pages to be labeled by intervals, with the understanding that the source is valid for requests whose date coordinate lies in the interval. This idea is described in more detail in “The Possible-World Wide Web”, where it is pointed out that it amounts to treating the Web as a kind of reactive system.

6. References