

Project P923-PF

Multilingual WEB sites: Best practice, guidelines and architectures

Deliverable 1

Guidelines for building multilingual Web Sites

Volume 2 of 5: Annex A

Web site technologies

Suggested readers:

This document is primarily aimed at anyone who is involved in the process of designing, building or managing WEB sites. It is of immediate relevance to those involved with multilingual WEB sites, but it nevertheless, provides information which will allow monolingual WEB site designers to design sites that are economically upgraded to multilingual sites.

EDIN 0008-0923

Project P923

For full publication

September 2000

EURESCOM PARTICIPANTS in Project P923-PF are:

- Koninklijke KPN N.V.
- France Télécom
- British Telecommunications plc
- Telecom Italia S.p.A.
- Portugal Telecom S.A.

This document contains material which is the copyright of certain EURESCOM PARTICIPANTS, and may not be reproduced or copied without permission.

All PARTICIPANTS have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the PARTICIPANTS nor EURESCOM warrant that the information contained in the report is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

This document has been approved by EURESCOM Board of Governors for distribution to all EURESCOM Shareholders.

Executive Summary

At this time, there is no procedure or tools for designing and managing web sites which takes into account the possibility of working in a multilingual environment. On the other side, there are a lot of tools already present on the market, which may be anyway exploited, for necessity or because they nevertheless are useful.

There are three main sections which will be taken into account: "Web middleware", Web Editors and Content management tools. The first section is composed by all basic blocks in the web architecture: they are of course necessary to build a site, but they are generally only loosely related with multilinguality. On the other side, sometimes the underlying concepts are useful: JavaScript, Dynamic HTML and CGI are basically well suited to create a multilingual system. Web Editors are what a web designer uses to create the site: they might give much benefit if they let the user keep in parallel all versions of her site, but this does not seem yet to be the case. The so-called Development Systems are much more integrated than a simple web editor, how powerful it can be: at the moment, even if they are not thought for multilinguality, they can however be adapted for it without too much hassle.

A last section follows with some comments on other issues arising when developing a multilanguage system; finally there are two appendices. The first one gives some information about the terminology used while speaking about characters, fonts and so on; the second is an article which explains the difficulties invalid in multilingual text processing.

List of Authors

Maurizio Codogno (CSELT/Telecom Italia, IT)

Malek Boualem, Jérôme Vinesse (CNET, France Telecom, FR)

Table of Contents

Executive Summary	1
List of Authors	2
Table of Contents	3
Abbreviations	4
A.1 Web Middleware	5
A.2 HTML.....	6
A.3 CGI and SSL.....	8
A.4 JavaScript	10
A.5 Dynamic HTML and ASP.....	11
A.6 XML	13
A.7 PHP.....	15
A.8 Character encoding and fonts technologies	16
A.9 Web Editors.....	18
A.9.1 FrontPage	18
A.9.2 HotDog Professional 5.5	19
A.9.3 FileMaker Pro 5 Unlimited	20
A.9.4 Adobe PageMill	21
A.10 Development systems.....	22
A.10.1 Visual InterDev.....	22
A.10.2 ColdFusion	23
A.11 Other issues	24
A.2 Metadata Standards	25
A.13 References	27
Appendix 1: Charset terminology glossary	28
Appendix 2: Multilingual text processing.....	30
1 Introduction.....	30
2 Multilingual text processing difficulties	30
3 Multilingual text input (Keyboard)	31
3.1 Keyboards.....	31
3.2 Interpreting characters and rendering glyphs (context analysis).....	31
3.3 Ideogram phonetic input.....	31
4 Multilingual text coding	33
4.1 Character coding	33
4.2 Writing systems coding.....	33
4.3 Language coding	33
5 Multilingual text editing.....	35
5.1 Bi-directional texts	35
5.2 Insertion / deletion of characters in bi-directional texts	35
5.3 Multilingual text justification.....	35
6 Multilingual text printing	37
7 Multilingual data exchange	38

Abbreviations

ASP	Active Server Pages
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
DHTML	Dynamic HTML
DOM	Document Object Model
DTD	Document Type Description
GIF	Graphical Interchange Format
HTML	HyperText Markup Language
IE	Internet Explorer
JPEG	Joint Photographics Experts Group
OLE	Object Linking and Embedding
PHP	Parsed Header Processing
RDF	Resource Description Format
RFC	Request For Comments
SGML	Standard Generalized Markup Language
SQL	Structured Query Language
SSI	Server Side Include
SSL	Secure Socket Layer
WYSIWYG	What You See Is What You Get
XML	eXtensible Markup Language

A.1 Web Middleware

There are many basic blocks necessary to build a Web site, and of course they are necessary for monolingual websites as well as for multilingual ones. In this section, those blocks are covered in some detail, in order to understand what relation they have with the main goal of the Project and whether they may be adopted "as is" or it is necessary to make some modification.

HTML is of course the "most basic" of the blocks, and no Web system can do without it. On the other side, being at such a low logical level means that it is general enough not to give problems while developing a multilingual Web site, at least for Western languages. Support for other language is also possible, but requires a bit more of care, and it would be difficult at the moment to mix in the same page data e.g. in German and Turkish, lest we not even think at Chinese.

The main limitation with HTML is that the pages written in that language are static. This is adequate for descriptions, but people often want to offer the user more interaction. There are some tools which have been *de facto* standardised to expand HTML with interactivity: CGI receives input from the user and performs operations involving that data, SSI has been the first system which let the developer add dynamic data within a HTML page, JavaScript adds the capability to modify the appearance of a page after some action from the user, like moving the mouse over a particular region of the screen, Dynamic HTML and ASP further evolved this paradigm. All of this components are transparent with respect to multilinguality, but they could be make use of that information, to show for example help in the language chosen by the user even if the main information is monolingual.

XML, or more properly the whole suite of specification under the XML umbrella, is a further step onwards. Its importance derives from the fact that it is designed to be extensible: in practice, it is possible to add meaning to a keyword by means of an ancillary file, like a style sheet. The logical step to work out several substyles for the various languages which all share a common general layout is immediate: unfortunately not all the components are quite ready at the moment, but the technology promises to be very useful in helping people to develop a common look and feel for their site while allowing the peculiarities of the renditions in the local languages.

Scripting languages like PHP are the logical link towards the issues developed in the next section: they give the developer the capability to interact easily with external databases, and to add constructs typical of programming languages like template inclusion and conditional statements to HTML. All of these features are really useful: they can be exploited, besides what hinted at in the previous paragraph, also to leave together all the information in a single document, relying to the processing from the system to present only the intended one to the user. This is really useful in the case of short excerpts of text like those found in captions.

The last part of this section gives some information about character encoding and font technologies. They are not really important for multilingual sites based only on Western languages, but should nevertheless be addressed, since they may affect the output of the pages, and especially impair further extensions of the site.

A.2 HTML

The most important thing in setting up a Web site is of course HTML (HyperText Markup Language), since this is the language in which every document is eventually rendered. HTML was born in 1990 as a simple way to add tags to a document in order to give it a logical structure. The situation remained rather quite until 1995 (version 2.0 of the language, RFC 1866): some tags were added to allow exchange of data among client and server, using forms, but there was no interaction problem among the various versions of browsers in the market. The situation abruptly changed with the advent of Netscape, the first commercial browser. In order to gain market share, Netscape started adding "features" to the language, so that users could write "nice" pages. The result was a change in the architecture of the language: while the old tags usually represented *content*, the new ones were mainly concerned with *representation*: you had a tag to centre text instead of a generic attribute for alignment, and you have another tag to mean that from that point on you want text displayed with a special font or colour. In this way, pages appeared nicer to people who wrote them, but not necessarily to other people, whose displays maybe did not render correctly the tags or had a different window resolution. When Microsoft Internet Explorer appeared, the race towards adding "features" became frantic, and it was not uncommon to find web sites which had a version offered to people using Netscape and another version offered to people using Internet Explorer: usually the few ones with other browsers were given a really basic page. Two ambitious projects, HTML+ (see http://www.w3.org/MarkUp/HTMLPlus/htmlplus_1.html) and HTML 3.0 (see <http://www.w3.org/MarkUp/html3/CoverPage.html>) were never finalised, due to lack of interest by the browser vendors, and the situation began to worsen, until the World Wide Web Consortium (<http://www.w3.org/>) started to work towards a unified model, since it feared that the fragmentation of the market could stop development. In the beginning of 1997, a new standard was adopted, namely HTML 3.2 (<http://www.w3.org/TR/REC-html32.html>) which comprised all the extensions to HTML 2.0 found in the browser, were they useful (for example, tables) or not (such as BLINK tag)

The last official version of HTML is 4.0, published in Spring 1998 and with a proposed "bug-fix" version, HTML 4.01, which should be adopted in a few months (see <http://www.w3.org/TR/html40/>). This version of HTML extends the language with mechanisms for style sheets, scripting, frames, embedding objects, improved support for right to left and mixed direction text, richer tables, and enhancements to forms, offering improved accessibility for people with disabilities.

Among the most important extensions from the point of view of the web designer, there is a new model for tables, in which it is possible to express the relative widths of columns; mechanisms are provided to include scripts in an HTML document which are independent of the scripting language; it is also possible to give a structure to a group of documents, via the LINK tag, to help people to find the relationships among the separate parts of a logical unique document.

In HTML 4.0 there is a standard mechanism for embedding generic media objects and applications in HTML documents. The OBJECT element (together with its more specific ancestor elements IMG and APPLET) provides a mechanism for including images, video, sound, mathematics, specialised applications, and other objects in a document. It also allows authors to specify a hierarchy of alternate renderings for user agents that don't support a specific rendering.

The problem of content versus representation was addressed with the introduction of the concept of style sheets. Style sheets simplify HTML markup and largely relieve HTML of the responsibilities of presentation, since they give both authors and users control over the presentation of documents – font information, alignment, colours, etc. Style information can be specified for individual elements or groups of elements, and may be specified in an HTML document or in external style sheets. Styles have therefore a limited capability for showing different rendering of the same document.

HTML 4.0 has moreover been designed with the help of experts in the field of internationalisation, so that documents may be written in every language and be transported easily around the world. This has been accomplished by incorporating RFC2070, which deals with the internationalisation of HTML. One important step has been the adoption of the ISO/IEC:10646 standard (see [1]) as the document character set for HTML. This is the world's most inclusive standard dealing with issues of the representation of international characters, text direction, punctuation, and other world language issues. HTML 4.0 offers greater support for diverse human languages within a document. This allows for more effective indexing of documents for search engines, higher-quality typography, better text-to-speech conversion, better hyphenation, etc.

It's important to keep in mind that HTML 4 has been created with a strong focus of accessibility and better authoring. The first thing means that most tags have now a way to specify an alternate rendering: as for authoring, you can choose among three different variants, each one with its own DTD - Document Type Definition - which is the model that sets out the rules and regulations for using HTML in a succinct and definitive manner. The variants are HTML 4.0 Transitional, which takes advantage of HTML 4.0 features including style sheets but nonetheless makes small adjustments to your mark-up for the benefit of those viewing your pages with older browsers which can't understand style sheets, like using BODY with bgcolor, text and link attributes; HTML 4.0 Strict, for people who want really clean structural mark-up, free of any tags associated with layout; and HTML 4.0 Frameset, to use HTML Frames to partition the browser window into two or more frames.

Even if browsers are often forgiving of errors in the HTML syntax, it is better to validate the document, that is to check its syntax against the rules implied by the DTD. In this way, people may be sure that every browser shows (more or less...) the same things. W3C offers a validation service at <http://validator.w3.org/>, so that content providers can use it to validate their Web pages against the HTML 4.0 Recommendation, thereby ensuring the maximum possible audience for their Web pages. In addition, it can be used to check conformance against previous versions of HTML, including the W3C Recommendation for HTML 3.2 and the IETF HTML 2.0 standard.

Note that the development of HTML has now been stopped, since W3C believes in the future of XML. An interim version, XHTML 1.0, has been released and will be explained in the following sections.

A comprehensive explanation of HTML can be found at <http://utopia.knoware.nl/users/schluter/doc/tags/index.html>.

A.3 CGI and SSI

HTML is a “static” language: this means that the page stored in the hard disk of the server is exactly what the user sees in his browser. She may click on links, but this does not actually mean that the original page is changed: a set of instruction is sent to the server, or to another different server, and a new page is loaded. Soon after the first explosion of the web, however, developers found it useful to add features to the server, so that it could cope with those limitations. The first results were CGI and SSI.

The Common Gateway Interface (CGI) bears this name because it is a standard for interfacing external applications with information servers, such as HTTP or Web servers: the standard was jointly created by CERN and NCSA when the latter started developing Mosaic. As it was said above, plain HTML documents that the Web daemon retrieves are static, which means they exist in a constant state: text files that don't change. A CGI program, on the other hand, is executed in real-time, so that it can output dynamic information. A simple example is the compilation of a form in a web page. Suppose that we have a Unix database which we want to “hook up” to the World Wide Web. When a user clicks on the “submit” button, she sends to the server some data which codes the query she wants to perform. This data has to be processed somehow: we need to create a CGI program that the Web daemon will execute to transmit information to the database engine, and receive the results back again and display them to the client. This is an example of a gateway, and this is where CGI, currently version 1.1, got its origins.

The idea is simple, but its implementation may be difficult. Actually, a CGI program is an executable, so it may do virtually everything. In practice, it has better to be quick, since otherwise the user just stares at her monitor wondering what happens. Moreover, since a CGI program is executable, it is basically the equivalent of letting the world run a program on your system, which isn't the safest thing to do. Therefore, there are some security precautions that need to be implemented when it comes to using CGI programs. The most important of those are that the Web server – and thus the CGI program - should be run as a user with no special permissions, and that CGI programs need usually to reside in a special directory, usually `/cgi-bin/`, to be controlled better.

A CGI program can be written in any language that allows it to be executed on the system, such as any Unix shell, Visual Basic, C/C++, TCL, Perl, or even COBOL if you feel at ease with it and you have a compiler handy. There is not a “best” language for CGI programs: C is has higher performance, but more is difficult to debug, modify, and maintain , so many people prefer to write CGI scripts instead of programs. Of course, if the program is written in a compiled language, the source need not to reside under `/cgi-bin/`, while shell or Perl scripts must. The program or script may access several environment variables which contains relevant information, like name and parameters it was called with, or the name of the server.

CGI is on the way towards standardisation under the IETF: for the time being, it is possible to find its specification at <http://hoohoo.ncsa.uiuc.edu/cgi/>.

Server Side Includes (SSI) is another way which the server may use to process information. The first implementation of SSI was made in the NCSA server, in 1994. A Server Side Include is a command that the web server processes before serving to the browser. It can be built into any web page, but having the server parse documents is a double edged sword. It can be costly for heavily loaded servers to perform parsing of files while sending them. Further, it can be considered a security risk to have

average users executing arbitrary commands as the server's user, if the generic execute command is allowed. Usually the solution consists in having only some directory in which executable pages are allowed, and maybe setting up a different extensions for parseable files, like .shtml. There is nothing which forbids the use of the standard extension .html, but performance issues are to be taken into account.

In order to build in a directive for the server to process, a special syntax is necessary. The commands appear as comments, between `<!--` and `-->`. This means that a server which does not recognise them passes them to the client which discards them. So, the general form of a SSI directive is

```
<!--#command tag1="value1" tag2="value2" -->
```

There are not many commands which can be specified, but they are really powerful, since it is possible to execute an arbitrary command through `/bin/sh` or include another file, albeit only in the subtree relative to the current document to avoid security leaks.

Documentation for SSI can be found at <http://hoohoo.ncsa.uiuc.edu/docs-1.5/tutorials/includes.html>.

A.4 JavaScript

After the processing from the server side, the next logical step was to move processing from the server to the client, easing so the former. The first attempt in this direction is the language Java, developed by Sun Microsystems. A web page may include the `APPLET` tag, which instructs the server to load a Java program, “applet”, and run it locally. However, Java is powerful but difficult to use, so Netscape devised a simpler way to have local computation. Since at the time of its creation the very word “Java” was really hip, they decided to call it JavaScript, adding confusion since there is no actual resemblance with Java. Indeed, JavaScript is basically a scripting language developed to add interactivity and power to web documents. A comparison with Java may be useful:

Java is (a) compiled into executable applets, (b) downloaded separately and viewed inside documents, and (c) strongly typed, that is all variables and variable types must be declared;

JavaScript is (a) interpreted by browser like HTML, not compiled, (b) embedded into HTML documents, and (c) loosely typed, that is not all variables have to be declared.

Although JavaScript is not as powerful as Java, two things make it widely accepted ; simplicity and speed. JavaScript is much easier to write and learn than Java, which requires the client to import packages and a bunch of other stuff, and a compilation of the code. On the other hand, JavaScript doesn't need to be downloaded as a separate executable file, and it simply exists within the HTML document.

Unfortunately there is no “official” standardisation of JavaScript. It was created by Netscape, so it is most compatible with Netscape. Internet Explorer 4.x supports 99% of what JavaScript has to offer, although IE 3.x is not quite as adorable. By the way, Microsoft came out with its own scripting language, VBScript (based on Visual Basic).

The syntax for using JavaScript is rather simple: the commands – remember that it’s a scripting language! – are embedded into HTML documents using the `SCRIPT` tag. For the sake of clarity, it is better to specify the type of scripting language used, `<SCRIPT LANGUAGE="JavaScript">` or even its version, `<SCRIPT LANGUAGE="JavaScript1.1">`. Since some old browser does not know what to do with the `SCRIPT` tag and would happily display the code, it is also customary to “hide” the script, putting it between comments `<!--` and `//-->`.

JavaScript, like many programming languages, relies on objects, functions, and event handlers to create workable programs. In a typical function, like

```
document.write("Hello, world!")
```

we can see an object (“document”, the output file) and a method (“write”, what we should do with the object). An object may also have properties (static attributes of the object, like `document.lastModified` which carries the date in which the document was changed). It is also possible to declare functions, which may have parameters and return values as usual.

Further information about JavaScript may be found at Netscape, <http://developer.netscape.com/docs/manuals/javascript.html>.

A.5 Dynamic HTML and ASP

The quest for interactivity in web pages reaches its peak with technologies like Dynamic HTML and ASP, both endorsed by Microsoft, which try to have a unified approach to the problem. Unfortunately, even if Microsoft assures that it is working with the World Wide Web Consortium (W3C) to help ensure interoperability and support for users on multiple systems with different browsers, the result is still not quite satisfactory.

The idea behind Dynamic HTML, or DHTML in brief, started from the consideration that Making simple updates, such as changing the colour of text after a Web page loads, traditionally has meant reloading the page. These limitations have slowed the user experience and have impeded interactivity on the Web. In the pages authored with Dynamic HTML, every element in the page is truly dynamic. After the page has been loaded, content providers can change any element of the page – text or graphics – without a round-trip to the server.

The foundation for Dynamic HTML is the Document Object Model (DOM). Its specification can be found at the World Wide Web Consortium, <http://www.w3.org/TR/REC-DOM-Level-1/>. The Document Object Model is a significant innovation: it makes all page elements programmable objects, individually and as part of various collections. Using the object model, people can access and manipulate virtually anything within the document. The HTML elements in the document are available as individual objects, meaning they an element and its attributes can be examined and modified by reading and setting properties and by calling methods. The text is available through properties and methods on the elements. The object model also makes user actions, such as pressing a key and clicking the mouse, available as events. These and other events can be intercepted and processed by creating event handler functions and routines. The event handler receives control each time a given event occurs and can carry out any appropriate action, including using the object model to change the document. The object model is moreover a superset of the JavaScript object model found in Netscape Navigator. This means that portions of the model are compatible with other browsers, even if they do not support Dynamic HTML. Further information about DOM may be found at <http://msdn.microsoft.com/workshop/Author/dom/domoverview.asp>.

The other ingredient for Dynamic HTML is Cascading Style Sheets (CSS), already hinted at when explaining HTML 4. In the case of CSS, "style" means everything that defines how the web pages look. Fonts, spacing, and line height are all part of style and with CSS different attributes may be applied to every tag used in the developed pages.

Interactivity is triggered by action: for example, the snippet of code

```
<h3 onmouseover="this.style.color = 'red';"  
onmouseout="this.style.color = 'black';">Title</h3>
```

shows as the headline will be triggered by two events (the mouse going over it and leaving it), in which case a different style is applied. In this case, the final result is that when the mouse passes over the text, the line becomes presumably red, while it is black by default.

Further information about DHTML may be found at <http://msdn.microsoft.com/workshop/author/dhtml/dhtmlowv.asp>

Another way devised in 1996 by Microsoft to ease interaction of web pages with a local environment is Active Server Pages, or ASP in brief. Simply put, ASP is a mix of HTML, scripts and ASP code that enables you to build dynamic and database-driven web sites. Microsoft describes it as "a server-side scripting environment that you can use to create and run dynamic, interactive, high performance Web server applications." This means that while standard HTML is only a display language, ASP allows you to tailor the information displayed on the page based on user interaction.

Like other tools such as Perl and CGI, server-side ASP is interpreted by the server and the information is returned to the browser to display. Therefore, ASP is cross-browser compatible if used as server-side scripting – in other words, if the script is interpreted on the server, is server-side. (Note: once lynx, the most known text-based browser, was said not to support ASP because it needed cookies. Now this is no more true). There is also client-side scripting, which is interpreted by the browser. Unfortunately, there is no guarantee that every browser understands it, even if it would be really useful since it would take away load from busy servers. The official terminology by Microsoft speaks about an Active Platform. The Active Platform reflects Microsoft's ideas about how a desktop computer and a server computer should communicate. It consists of two parts: the Active Desktop and the Active Server. The Active Desktop refers to the client side, or the user's side, where HTML files are displayed on a web browser. The Active Server refers to the server-side component. This consists of pages that can be interpreted by the server, hence the term Active Server Pages.

ASP runs on Microsoft IIS and Personal Web Server (PWS) on Win95,98 and NT; ASP support is also available in O'Reilly's Website Pro 2.0; moreover, with the addition of commercial software from Chili!Soft, (Chili!Soft ASP, <http://www.chilisoft.com/chiliasp/>), ASP can also be run on a number of other servers, including Netscape FastTrack and Enterprise, and other ISAPI compliant web servers.

How does ASP technology work? Normally, HTML files are "interpreted" on the client side (in a user's web browser). With Microsoft's Active Platform, the server can also interpret files. Interpreting to a server means that before displaying a file in a user's browser, the server will first look at the file's extension. If the file has a standard extension, such as .html or .htm, the server will pass the file to the browser. If the file has an extension of .asp (or .asa), the server will open the file and look for tags that mark ASP code. ASP code is contained inside special tags: <% before the code begins and %> after it ends. Then, the server evaluates the code contained inside these brackets and replaces it with HTML code. This HTML code is generated based on information about the user's setup or other conditions on the client side. The resulting page, which contains some of the file's original markup and some markup that has been generated with ASP, is eventually sent to the browser that requested it.

ASP pages may be used for nearly everything, like for example changing the text in the page depending on the day of the week in which it is displayed. But the most important use for them is certainly the ability to access databases. Among the commands accepted by ASP, indeed, there is the possibility to connect to a SQL server and make queries to it, getting responses and displaying them to the client.

Further information about ASP may be found at the location <http://www.learnasp.com/learn/index.asp>.

A.6 XML

As explained in the section about HTML, that language is gradually being replaced, at least in the intentions of the folks at W3C, by XML, the 'Extensible Markup Language' (extensible because it is not a fixed format like HTML), which is designed to enable the use of SGML on the World Wide Web.

XML is not a single, predefined markup language: it's a metalanguage – a language for describing other languages – which lets you design your own markup. (A predefined markup language like HTML defines a way to describe information in one specific class of documents: XML lets you define your own customised markup languages for different classes of document.) It can do this because it's written in SGML, the international standard metalanguage for markup.

XML is designed to make it easy and straightforward to use SGML on the Web: easy to define document types, easy to author and manage SGML-defined documents, and easy to transmit and share them across the Web. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML: moreover interoperability with HTML is another design goal of XML.

The most important concept in XML is probably the DTD (Document Type Description), which is already present in HTML but to a lesser extent.

A DTD is a file (or several files to be used together), written in XML, which contains a formal definition of a particular type of document. It sets out what names can be used for element types, where they may occur, and how they all fit together. For example, if someone wants a document type to be able to describe <List>s which contain <Item>s, part of her DTD would contain something like

```
<!ELEMENT List (Item)+>
<!ELEMENT Item (#PCDATA)>
```

This fragment defines a list as an element type containing one or more items (that's the plus sign), and items as element types containing just text. XML is the formal specification language which processors read to automatically parse the DTD and then use that information to identify where every element type comes and how each relates to the other, so that stylesheets, navigators, browsers, search engines, databases, printing routines, and other applications can be used. The above fragment lets a user create lists which get stored as:

```
<List> <Item>Chocolate</Item> <Item>Music</Item>
<Item>Surfing</Item> </List>
```

How the list appears in print or on the screen depends on your stylesheet: there is normally no need to put anything in the XML to affect formatting in the way that had to be done with HTML before stylesheets. Actually, XML may even exist without any kind of rendering, for example as an intermediate format to make queries to databases.

In effect, a DTD provides applications with advance notice of what names and structures can be used in a particular document type. Using a DTD means to be certain that all documents which belong to a particular type will be constructed and named in a conformant manner.

The rendering of XML is still the weakest part of the whole system: Microsoft's Internet Explorer handles XML in its version 5, but currently still renders it via CSS, using a largely HTML-derived model; thus not all the stylesheet options work. Microsoft are also the architects of a hybrid solution in which people can embed

fragments of XML in HTML files because current HTML-only browsers simply ignore element markup which they don't recognise. Netscape does not recognise at all XML, even if the publicly-released Netscape code (Mozilla) has resulted in a test XML implementation including an application of RDF plus James Clark's Expat XML parser. Therefore, only SGML parsers can at the moment display XML in its full glory. The decision of W3C to publish XHTML 1.0, a rewriting of HTML 4 with the stricter rules needed by XML, may ease the transitions to the new model.

Apart from the World Wide Web Consortium site, more information about XML may be found at <http://www.oasis-open.org/cover/sgml-xml.html> and <http://www.ucc.ie/xml/>.

A.7 PHP

PHP3.0 is a scripting languages meant to be used by the web programmer: its code can be inserted directly alongside HTML, which makes the language all the more convenient.

Created by Rasmus Lerdorf in what began as a personal project (titled PHP/FI), the language quickly gained popularity and was almost completely rewritten by a group of six developers, and in turn, reborn as PHP3.0. The language enjoys an extremely active developing environment, due, in large part, to the fact that the language is freely available for download on the web.

Much of PHP3.0 is a combination of Perl, Java, and C concepts. The syntax structure borrows heavily from C, making it an easy language to learn for even the novice programmer. PHP3.0 performs sophisticated mathematical calculations, provides network information, offers mail and regular expression capabilities, and much more; but its strongest feature is its database interfacing capability, supporting many of the most popular database servers on the market, including MySQL, Oracle, Sybase, mSQL, Generic ODBC, and PostgreSQL.

Keep in mind that PHP3.0 is a server-side scripting language. Thus the language interpreter must be installed and configured on the server before one can execute commands.

It has been said that PHP3.0 commands can be easily inserted alongside HTML. The syntax is quite simple: markup brackets `<? and ?>` are used to embed commands. If the command is simple, the syntax `<?php . . . ?>` may be used instead. Note that any HTML commands placed within the PRINT statements will be interpreted by the browser, and perform their usual actions.

Other very powerful use of PHP3.0 are the capability of building HTML templates that can be included in a page, and its capability of modifying variables passed from HTML forms.

Database manipulation is quite simple the usual commands for interacting with the database are used.

Further information about PHP may be found at <http://www.php.net/>.

A.8 Character encoding and fonts technologies

Originally, the ASCII character encoding scheme dominated the computer world. This is a 7-bit encoding scheme and does not assign codes to accented characters that might appear in Latin-based character sets (characters such as á). Actually so-called “national versions” existed with the common used accented characters, but the solution was really ugly, since they would replace other useful characters like brackets or braces. The various ISO-8859-x codes improved on this with 8-bit codes that could represent Western European, Cyrillic and Greek (<http://www.iso.ch/>). Other 8-bit standards have appeared in this series that can encode many other character sets. Independently conceived coding schemes were used for other scripts, such as Chinese (e.g. GB2312) and Japanese (e.g. Shift-JIS). These latter coding schemes requiring 16 bits due to the much larger number of characters that they have to encode. Note that not only characters collection, but also character terminology is not simple to understand: in Appendix 1 there is the explanation of the more common terms in the field.

For multilingual applications, and many monolingual applications, these language-specific coding schemes will almost certainly give way to Unicode, a 16-bit encoding scheme which can encode the vast majority of characters in World's languages (<http://www.unicode.org/>, Moorcraft 1997). The production of a single 16-bit standard has been enabled partly through unification of the various Han (Chinese) characters in Chinese, Japanese and Korean.

Gradually, Unicode is being adopted at the system level in operating systems such as Windows NT, and in programming languages such as Java. At the applications level, the latest versions of both Netscape Communicator and Internet Explorer support Unicode to different degrees.

This list shows only the surface of the problems involved with the representation of characters: Appendix 2 explains in more detail many issues about multilingual text processing.

Deciding what character encoding scheme to use is only part of the problem. A font set will be required to map between the codes and the character images (glyphs) that appears on the screen or paper. Font sets are beginning to appear which contain a reasonably comprehensive implementation of the characters in the Unicode set. For example, Bitstream's Cyberbit font implements some 39,000 characters from the Unicode range (<http://www.bitstream.com/>).

The next problem with fonts is making sure that what looks good on your screen, looks good (or even appears at all) on other screens. Typically when designing WEB pages, only a small number of common fonts are used, since web clients had a fixed set of fonts for displaying. This means that you can have a reasonable confidence that your page will display on other screens roughly the way that it appears on yours, because you had no way to know which other fonts are installed on the remote computer.

Ideally, you would like to be certain that everyone who views your WEB pages has the same fonts installed that you have on your machine: eventually, the quest for new effects led both Netscape and Microsoft to devise new technologies to let the browsers use more fonts.

Netscape teamed up with a company called Bitstream to devise a new technology called TrueDoc. When built into authoring tools and web browsers, TrueDoc provides

an easy and secure way to transfer font data from a web author's server to a user's browser. It is used in Netscape Communicator 4.03 (and higher) or Internet Explorer 4 and 5.

TrueDoc has two main components: the Character Shape Recorder (CSR) and the Character Shape Player (CSP). Authoring tool developers include the CSR in their products. The CSR records characters from the fonts that web authors use in their documents and stores them in a highly compact data structure called a Portable Font Resource (PFR). The CSR is built into many HTML authoring tools, while the CSP is built into the browsers. The advantages of this technology are several: it allows web authors to format text without resorting to bitmap graphics (GIFs, JPEGs), reduces download time, preserves text which users can search through or copy and paste, and dynamically displays text based on the user's display. Dynamic fonts are downloaded with an HTML page, the same way as GIF and JPEG images are. A browser that can display TrueDoc font files renders the fonts on the screen (or on a printer). Browsers that cannot display TrueDoc fonts use alternate fonts on the user's system.

Dynamic fonts may be specified in one of two ways: using FONT FACE tags within the HTML document to specify the fonts; or (preferred) using a Cascading Style Sheet to specify the fonts. The CSS will include the URL for the source of the font data. In both cases, a LINK tag is necessary to associate the document with a PFR.

Further information about TrueDoc may be found at <http://www.truedoc.com/>

On the other side, Microsoft and Adobe started the OpenType Initiative. OpenType, also known as TrueType Open version 2, is an extension of Microsoft's TrueType Open format, adding support for Adobe Type 1 data. An OpenType font can have Type 1 data only, TrueType data only, or both. The Type 1 data can be rasterized by a Type 1 rasterizer (such as Adobe Type Manager) if installed, or converted to TrueType data for rasterization by the TrueType rasterizer. The exact rasterization behaviour will be a function of the rasterizers present in the system, and user preference. The subsetting and compression technology of OpenType makes the OpenType initiative especially relevant to the Internet and the World Wide Web, since it allows for fast download of type.

OpenType will make it possible for Web page creators to include high quality on-screen fonts with their online documents. The net effect is that page designers will be able to produce richer documents, and at the same time reduce the time required to download and display these documents on the viewers PC. Even if Microsoft and Adobe claimed that they would jointly submit a proposal to the World Wide Web Consortium for a standard to embed OpenType fonts in WWW pages, it was not possible to find such information on the W3C site.

Further information about TrueDoc may be found at <http://www.microsoft.com/OpenType/faq/faq9.html> .

A.9 Web Editors

There are too many programs which can be classified as "Web editors" for a comprehensive list to be useful in this context. Besides, the compilers of two such lists both confessed that they used vi or EditPad to write their HTML pages... Those confessions give the idea that a Web editor is not really mandatory, after all!

Here there is instead a list of some tool which can be used for web publishing at large: that is, an integrated environment for the user who wants not only to write text but also to add other features to her site. Needless to say, multilinguality is the feature we are interested in. The main concepts which can be found in all the products presented in this sections are two: a way to edit in parallel different parts of the site, so that the ancillary languages may be added even while completing the pages in the main language, and a eagle-eye view of the layout of the site, to help developers see what's still lacking. Actually there is not much difference among those products, at least in that respect, and therefore no-one is particularly recommended.

Microsoft FrontPage offers embedded in the design of the system an easily distributed editing framework, so that localizers may start working while the main designers have not yet finished the primary layout, and the Navigation View which is useful to see at once the structure of the site and therefore the parts which still need upgrading.

Also HotDog Professional has a concept equivalent to the Navigation View of FrontPage, while the possibility to work off line and upload just the modified pages may be seen as the logical equivalent of the distributed editing environment from the Microsoft product.

FileMaker Pro 5 Unlimited, instead, seems to lack the concept of the site view. On the other side, the integrated environment gives the capability of remote editing, easing thus the development when there are several branches of a company, located in different nations, working on the site.

The already familiar concepts may be also found in PageMill by Adobe, with the names of Built-in Uploader and Graphical Links View. PageMill is the only system presented in this section with another module, Web Commerce, which may be exploited in a multilingual, or better a multinational, environment. Unfortunately it is not really certain that the module can explicitly deal with localisation, even if it will give help in addressing the problems arising with different currencies and such.

A.9.1 FrontPage

Microsoft FrontPage 98 is a quick, effective way to create and manage professional-quality Internet or intranet sites without programming. It makes it easy for new users and professional Web developers alike to build and maintain great-looking, professional-quality Web sites in no time.

The program brochure depict among the characteristics of the program: - easy-to-use, leading edge features which let the user create professional Web sites without programming; - possibility to create WYSIWYG frames pages and draw HTML tables in the WYSIWYG FrontPage Editor; - drop in sophisticated, interactive functionality using FrontPage components.

Comprehensive management tools let you quickly build and maintain well-organised Web sites. With automatic hyperlink maintenance, you never have to worry about

broken links. Plus flexible collaboration features let you work with others on your Web site.

Seamless integration with existing content and with desktop applications you already have makes you productive from the start. And strong browser integration makes it easy to customise and view your Web site's content.

Among the characteristics of FrontPage 98, there are: *FrontPage Themes* - consistent backgrounds, bullets, banners, hyperlinks and navigation bars across entire Web site; *Navigation View* - to create and manage the navigational structure of a Web site; *Shared Borders* - shared headers and footers, or right and left margins across the pages; Automatic Navigation Bars; WYSIWYG Table and Frame Editing Tools; Form Save Results to Email; Banner Ad Manager; Dynamic HTML and Cascading Style Sheet Support; new Image Editing Tools, namely *Image Composer 1.5* with *GIF Animator*.

As noticed above, FrontPage offers also a tight integration with Microsoft Internet Explorer 4.0. Actually, there is even too much integration: FrontPage reportedly lets the users write nonconformant HTML pages which can be read by IE, but not by Netscape, much to the dismay of those using the latter browser.

Further information about FrontPage 98 may be found at <http://www.microsoft.com/frontpage/98/98IntPg1.htm>.

A.9.2 HotDog Professional 5.5

An independent company, Sausage Software offers a suite of programs under the brand name HotDog. The most important of these is HotDog Professional, presently at version 5.5.

HotDog Professional 5.5 WebSites is based entirely on a *local mirror system* and does not use proprietary formats to define a Web site. A HotDog Professional 5.5 WebSite defines a set of folders and files on a local computer and provides the ability to quickly publish these folders and files to another computer (usually an ISP's Web server). This makes it possible to create, edit and preview a Web site on a local computer and then automatically copy the changes to a Web server where an entire audience can see the site via the Internet.

HotDog Professional 5.5 sports a powerful *file management interface* making it possible to create, add, remove and edit Web site files from a single view. Drag and Drop files from local or remote file lists to easily add files to a Web site. Full control is held over Web site structure and it's easy to re-organise entire folder trees quickly. Choose to apply Web site features across single or multiple documents, folders or even an entire Web site.

WebSites makes it easy to *upload the site* and any associated files that are needed to make the site work properly on a Web server by means of publishing. When publishing a Web site, the choice is given to: - publish to a local directory or a remote Web server; - upload any necessary support files (fully customisable); - only upload any changes that have made been made to the Web site; - replace user-definable sections of Web pages when uploading (for example, a legal notice that's common to every page).

A built-in mechanism is supplied for *error-checking* all of the Web page links across an entire site. The Error Checker will automatically: - check a Web site's local links, displaying any broken links that it finds; - check a Web site's remote links, which can

be checked for validity; - allow editing of any broken local or remote links in place and automatically change all references to the link across an entire Web site.

WebSites is also supplied with a built-in *Resource Checker* for locating external resources (links to files outside of the local Web site folder tree) across an entire Web site. Resource Checker will automatically: - validate a Web site's local links, displaying any external resources that it finds; - automatically collect the resource and alter all links to it; - allow editing of any external links in place and automatically change all references to them across an entire Web site.

Further information may be found at <http://www.sausage.com/>

A.9.3 FileMaker Pro 5 Unlimited

FileMaker Pro 5 Unlimited, by Claris, is an integrated system which includes FileMaker Pro 5 and the FileMaker Web Server Connector to integrate FileMaker Pro with robust Web Servers to provide increased performance, security and SSL (secure socket layer) support to your web solutions.

Here there is some feature offered by this product.

New Instant Web Publishing Features: Built-in HTTP support lets authorised users search edit, delete, and update FileMaker Pro records for an unlimited number of users. You don't need to write HTML code or have a separate web server! With support for CSS (Cascading Style Sheets) users now have the ability to publish databases to the Web using the layouts already designed in FileMaker Pro.

Web Themes: Instant Web Publishing in FileMaker Pro 5 Unlimited has seven style options for how the Web pages look, that is, seven Style options. Five of the Styles (Classic, Wheat, Purple, Neutral, and Text Only) have essentially the same functionality but are implemented with a different look and/or technology. The other two Styles are for specific user scenarios (Search Only and Entry Only).

Web Security: A new security option is available to limit access to your databases. You can now specify the IP addresses that can request data from the FileMaker Web Companion. For example, you can now specify that only IP addresses that start with 1.2.3.* can access your databases. Additionally, the log file format has been changed to comply with the NCSA/CERN compatible Common Log format. Also, there are new options for tracking Web Companion activity in log files.

QuickTime and QuickTime VR support: Include QuickTime movies, sound, and VR in container fields to view QuickTime movies inserted into FileMaker Pro via a Web browser. Movie functionality will let users play movies (with the sound enabled or muted), go to any point in a movie, and flip through a movie, either forwards or backwards, frame by frame.

Secure Socket Layer (SSL) Support: Use the FileMaker Web Server Connector to leverage SSL technology built into the powerful Web Servers FileMaker Pro 5 Unlimited works with so that data passed between web browsers and the web server is encrypted to provide maximum security.

FileMaker Pro 5 Unlimited is designed to work with the major Web Servers: it is also possible to remotely manage the server, via a Web Connection.

Further information about FileMaker Pro may be found at Claris' site, http://www.claris.com/products/fmu_features.html

A.9.4 Adobe PageMill

Adobe PageMill (Windows and Macintosh versions) provides an intuitive, easy-to-use interface to create professional-looking Web pages quickly and easily. Access common functions and commands on the convenient button bar or via pull-down menus.

Web pages are created by typing in text or dropping in content from current office and graphics applications. The same thing applies to tables, frames, animation, sounds, movies and Java applets. Among the extensions, there are:

Built-In Uploader: when a site is completed, it can be transferred directly to the Web using the built-in FTP Uploader, which will upload the entire site, or just changes, to any Web server, without requiring any special server extensions.

Graphical Links View: a new graphical view of page links enables to see a visual map of site's links and other resources.

Other Site Management Features: new site-wide spelling checks and search and replace for text, images, and links make site maintenance fast and efficient.

Errors Directory: after loading a site, Adobe PageMill automatically analyses it and reports errors in an Errors directory.

Externals Directory: files from locations outside of their intended site folder are not included when a site is uploaded to a Web server, thus they appear as broken links. Now, it is possible to copy quickly external local files to the site's resource folder.

Web Commerce: with a built-in link, it is possible to use Open Market ShopSite Express software which lets you add secure online commerce functionality to the created Web site. Drag and drop order buttons from ShopSite Express into the Web pages to create a convenient shopping cart so your customers can shop directly from the site. It is also possible to calculate automatically shipping and tax charges and e-mail receipts to online customers.

Moreover: Support for Java and ActiveX Controls; Ready-to-use examples of Web Graphics and Animations; CGI and JavaScripts.

Further information about PageMill may be found at Adobe's website: <http://www.adobe.com/products/pagemill/main.html>

A.10 Development systems

Usually it is not enough to have a Web editor, how powerful it could be, to create a Web site. The reason for this is that it is common for a site not to be static, but to change with respect to user input (for example, different basic requests could lead to different views of the same data) and dynamic material (for example, a system which traces the current position of a parcel needs to be tied to a changing database). This led some company to the development of integrated systems which make it simpler to perform such tasks.

In this section, Visual InterDev, from Microsoft, and ColdFusion, from Allaire, will be presented. Keep in mind that PHP is functionally equivalent to those products, at least if considering the main features. Its interface is much more raw, of course: this decided against presenting PHP in this section, since it is not probable that a site will be built using it.

Both Visual InterDev and ColdFusion seem very promising in helping people to develop multilanguage web sites, and they are probably really useful. Moreover, they both include a sophisticated Web Editor, avoiding therefore the need to buy (and learn!) another paradigm to build pages. Database access is well supported, and so is the creation of dynamic pages; last, both products have a large installed base and support groups, usually run by enthusiastic users: this is certainly a bonus.

There are however some difference in the products, which do not involve multilinguality but are nevertheless important in choosing the support package. Visual InterDev is a Microsoft product: this means that in a Microsoft-based environment it could be preferred to have a uniform system. The strongest point of ColdFusion is instead its really powerful markup language. Unfortunately this is not standardised, so developers have to learn it, but the labour is paid for.

A.10.1 Visual InterDev

Visual InterDev, produced by Microsoft and now at version 6.0, has been created for rapidly designing, building, and debugging data-driven applications. Among its features, there is:

An integrated WYSIWYG page editor to visually construct sophisticated HTML and ASP pages, easily switching between multiple page views.

Step-through debugging of client- and server-side scripts for Microsoft Visual Basic Scripting Edition (VBScript) and JavaScript.

Build cross-platform applications that target any HTML 3.2 browser, on any platform, or optimise for Dynamic HTML

Microsoft Transaction Server is included in the package, for building enterprise-ready, scalable Web applications.

It is possible to connect to data from any ODBC-compliant database, like Oracle, Microsoft SQL ServerTM, Sybase, Informix, DB/2, Microsoft Access, dBase, Paradox, and others.

Database schemas may be visually designed and procedures may be stored. The basic idea is to Attach to and browse the data in tables and views, and visually create SQL queries for any ODBC- or OLE DB-compliant database. It is also possible to drag and

drop from the Data Environment to quickly create sophisticated, database-driven HTML forms, and reports.

Needless to say, there is a strong synergy with Microsoft SQL Server: the format is the same, so the data can be migrated to a pure SQL Server solution.

Further information about Visual InterDev can be found at <http://msdn.microsoft.com/vinterdev/prodinfo/datasheet/default.asp>.

A.10.2 ColdFusion

ColdFusion, a product from Allaire which is now at version 4.0, is an integrated development environment which includes visual programming, database, and debugging tools.

The idea underneath Allaire strategy is to have a special markup language, CFML, which is a superset of HTML that let the user specify more complex processing and database access. It is also possible to create and manipulate variables within dynamic pages, pass variables between pages and create browser cookies.

A rich function library is provided, with more than 200 functions for evaluating text strings, dates, times, lists, queries and mathematical expressions. Moreover, there are locale functions for accurate date and time formats world-wide.

It is also possible to create and manipulate n-dimensional arrays and structures (associative arrays) for sophisticated application logic and more manageable code, and to control the content of dynamic pages with conditional statements (if-elseif-else branching) that can use full Boolean operators (AND, OR, NOT, etc.) as well as CASE statements, and to control application logic and execution with looping.

Inclusion of pages within other pages allows developers to reuse complex code, formats or functions throughout an application and create reusable components with ColdFusion Extensions (CFX).

Editing may be performed with ColdFusion Studio, an editor for HTML, CFML and XML, which has among other features tools for creating complex SQL statements to select, insert, update, or delete data from any database with drag-and-drop, and a built-in validator for links and syntax.

Any ODBC compliant relational database may be accessed, and any SQL may be used to select, insert, update, or delete data. It is possible to connect to Oracle, Sybase, DB2, and Informix with native database drivers for enhanced performance and control.

Conditional and dynamic SQL may quickly be created.

Allaire has a nice strategy to bring developers in contact: it sponsors users' mailing lists which can be useful to help people quickly find information and answers to their questions.

Further information about ColdFusion can be found at <http://www2.allaire.com/Products/ColdFusion/>.

A.11 Other issues

The list of tools and procedures reviewed in this document is certainly useful, but they are of course biased, since they are meant to be used in a monolingual system. In this section some of the most important issues which involve multilingual systems are presented, together with a short discussion of their impact on the system.

Typesetting: while Western languages are written left-to-right, this need not be the case for other languages. Arabic is written right-to-left, Japanese is often written up-to-down. Moreover, there is a lot of smaller differences which may not be noticed at first, but may anyhow affect the global output: for example, while in Western languages justification is performed by adding interword space, in Arabic the character forms are changed. Third, there is some typesetting difference even among Western languages. Think for example at the way that quotation marks are rendered: every language has its conventions.

Appendix 2 describes in detail the difficulties involved in multilingual text processing, at various levels: keyboard input; character, font and language coding; bidirectionality, justification and everything belonging to multilingual text editing, and printing and exchanging of multilingual data in general.

Some of these problems will probably be eventually solved at other levels. For example, browsers automatically care for justification, and it can be supposed that translators, either human or automatic, know how to deal with inter-language quirks. The issues about bidirectionality, or *bidi* for short, are more problematic, and there is a lot of work going on both at W3C and in ISO. For the time being, however, it may be more useful to avoid such problems, limiting ourselves to the Western languages: the present operational target of the companies involved in the Project is anyway that one.

Even limiting ourselves to Western languages, developers and web page designers should face a very important problem while designing the site layout. As we have seen in the main text of this deliverable, the length of a text is not the same when translating from a language to another. Empirically it has been shown that Italian and German versions of a text document are 20% longer than the corresponding English version, with French lying more or less in the middle. Thus, a "perfect balance" of text and images in the English version of a site may as well result in a ugly mishmash in the Italian version, which is less compact.

This problem is however not addressed by any of the tools discussed in this document, since from their point of view there is not a single document, but a series of documents which in the best of scenarios is simply loosely related. However, a tool which lets the developer create "templates" is rather useful, because it helps her to work on smaller subsystems which can be checked easier.

Such a solution is also great when people decide to develop a system which is not completely multilingual, but only partially so. This means that for each document there is not necessarily a version in every supported language, but rather that if the version exists it is displayed, and otherwise either its equivalent in the default language is displayed, or a page is presented to the user where she may choose the preferred language among those available. The latter approach may be seen in Europe (<http://www.europa.eu.int>), the official server of the European Union. Since the EU is multilingual on nature, it is politically necessary to present the information in the site

in every official language of the Union; but sometimes this is not feasible, for it still has to be translated.

While it is possible to specify such a behaviour by just working with separate files, conditional HTML is much more useful if a design is preferred which mixes information in different languages. The developer may localise all general information, such as header and footer, and may exploit the preferences of the user in order to present her the other part of the page in another language which is at least partially known.

The last point is the character set used for the pages. The logical choices, at least for our companies and limiting to Western Languages, are Unicode and the ISO8859-x family. The difference between these choices does not usually concern the user, since the major browsers support Unicode and even if the user has no complete Unicode font, this is not relevant, provided that only Western characters are used. ISO8859-1 is the default language for HTTP, so there is no problem both for development system and for users. ISO8859-15 (ISO Latin 9, the character set with the Euro symbol) is being implemented in browsers, but developers should check that the relevant tools support that charset. There is however a big difference in the way data is stored: Unicode needs two octets per character, while ISO8859-x uses just one octet. The problem does not lie of course in the space used for the text, since this is negligible in relation with the other elements of the web sites: if Unicode or a charset different from ISO8859-1 is used, however, a misconfigured server could lead to serious problems resulting in not reading anything at all. Think at what happens when a two-octet character is read as two distinct bytes!

A.2 Metadata Standards

From the point of view of Content Management, it is really useful to know the standards relative to metadata, so that one can choose that or those best suited to the site which must be built.

In general terms, a metadata is "data about data", that is information about the data, which describe content, quality, condition and other characteristics of data, all in a standardized representation. The main goal of a metadata standard is therefore to provide interoperability between applications that exchange machine-understandable information.

Everyone knows that "the good thing of standards is, there are so many to choose from". Metadata is no exception, and so there are many standards to choose from. Here some of them are listed.

MPEG-7, an ISO/IEC standard developed by MPEG (Moving Picture Expert Group), is formally named "Multimedia Content Description Interface". It aims to create a standard for describing the multimedia content data that will support some degree of interpretation of the information's meaning, which can be passed onto, or accessed by, a device or a computer code. Since it is a metadata standard, it is not aimed at any application in particular, and supports as broad a range of applications as possible. More information about MPEG-7 may be found at <http://www.cselt.it/mpeg/standards/mpeg-7/mpeg-7.htm>.

SMPTE, the Society of Motion Picture & Television Engineers, is devoted to advancing theory and application in motion imaging, including film, television, video, computer imaging and telecommunications. Among its operation there is the definition of many standards, recommended practices and engineering guidelines for

motion imaging. More information about those standards may be found at <http://www.smppte.org/stds/index.html> .

Dublin Core is a simple content description model for electronic resources: that is, something that aims to facilitate discovery of electronic resources. Its work started from the librarians' necessity to widen their cataloguing capacity, and got many contributions from the networking and digital libraries research communities. Its main characteristics are simplicity - non-catalogers must be able to use it, semantic interoperability across disciplines, and extensibility. More information may be found at <http://purl.oclc.org/dc/>.

World Wide Web consortium started from Dublin Core to build its own standard, RDF (Resource Description Framework). It integrates a variety of web-based metadata activities, including sitemaps, content ratings, stream channel definitions, web crawling data, digital library collections and distributed authoring. XML was chosen as the interchange syntax to keep together all this data.

W3C hosts a series of RDF Working Groups: further information may be found at their site, <http://www.w3.org/RDF/> .

From the point of view of a multilingual web builder, the importance of a metadata standard is that it gives him a framework where he can put information about the languages of the various parts of the site, in such a way that some savvy external application may exploit them.

A.13 References

- [1] ISO/IEC 10646-1:1993. International Standard - Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane.
- [2] H. Nussbacher, "Handling of Bi-directional Texts in MIME", RFC 1556, December 1993.
- [3] D. Connolly, L. Masinter, "The 'text/html' Media Type", RFC 2854, June 2000.

Appendix 1: Charset terminology glossary

- **UCS** Universal Character Set, the
- **Repertoire**: the set of abstract characters which must be coded: for example, the Greek alphabet (note that even if "Greek uppercase Alpha" has the same form as "Latin uppercase A", they belong to different repertoires. On the contrary, "space" is a part of both repertoires, but of course there is not a "Greek space" different from the "Latin space")
- **Coded Character Set (CCS)**: a mapping of a repertoire on a set of integer numbers: for example, ISO8859-1, where "Latin A" is mapped into number 65.
- **Character Encoding Form**: a mapping of a CCS on a particular sequence of data types, like bytes: for example, UTF-8, in which a character is mapped in a number of bytes which varies from 1 to 5.
- **Character Encoding Scheme (CES)**: a serialised encoding form, in which the transmission order of the bytes in a multi-byte sequence is defined, for use on the network: for example, UTF-16LE, where each character is coded in a 16-bit number, and the first octet is the least significant part of the number.
- **Transfer Encoding Syntax**: a CES modified for transmission over a protocol: for example, Quoted-printable is a TES in which escape sequences are used to code characters not allowed in a 7-bit environment.
- **Font**: the visual representation of a group of characters (that is, not necessarily all of them) of a repertoire: for example, Arial Narrow.

Appendix 2: Multilingual text processing

1 Introduction

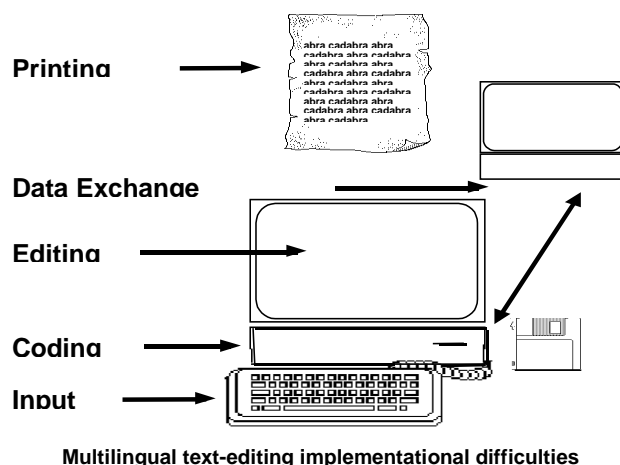
Users of more and more applications now require multilingual text processing tools, including word processors, database creation and management systems, desktop publishing systems and more recently multilingual web-based applications. In the area of automatic or machine-aided translation, multilingual text editors are a basic tool for pre-editing source text and post-editing target text. Another new area where multilingual text editors could be of great use is that of internationalization and localization of software and associated documentation for use in a multi-cultural environment. These areas were born as a direct effect of the emergence of new technology and the globalization of the Information Technology market. Many organizations and projects work to one extent or another within these areas (CEC, CEN, Esprit, Eureka, Internet, JSA, Linux International, Unicode, TEI, etc.).

Processing languages not based on the Roman alphabet poses a number of difficulties. For example:

- Arabic is written from right to left.
- Chinese contains thousands of ideograms, which obstructs a one byte coding.
- In Thai and other Indian languages, the sequence of characters does not correspond to its phonetic equivalent and one character may even be drawn encircling others.
- In Korean, characters are fused to make syllables.
- etc.

2 Multilingual text processing difficulties

Multilingual text processing difficulties occur on several levels: input (keyboard), coding, editing, printing and data exchange (see Figure 1).



Multilingual text-editing implementational difficulties

Figure 1.

3 Multilingual text input (Keyboard)

3.1 Keyboards

Though many keyboards represent only ASCII graphic characters (or ISO 646), certain localized (adapted) keyboards may also include keys for special or accented characters. For example, French keyboards generally feature keys corresponding to "à ç é è ù" accented characters, while characters which contain circumflexes or dieresis "ê ï" are input by two successive key presses. In addition, there is generally no single key on a French keyboard that allows one to produce characters that exist in other European languages, such as "ñ" or "ò".

In a broad multilingual context one could scarcely begin to imagine a keyboard that contains all possible characters. The inclusion of such languages as Chinese (with more than 6000 ideograms) or Arabic (approximately 4 sets of 28 letters and 10 vowels) requires the definition of specific keyboard input programs. Solutions proposed by computer manufacturers are very heterogeneous. Theoretically there exists a standard input method for keyboards with 48 keys (ISO/IEC 9995-3), at least for the Roman alphabet, but it is rarely used. A number of keyboard input methods for the ISO 10646 characters was recently proposed using hexadecimal codes or composition. But these keyboard input methods always require the user to know and memorize a huge number of codes and it is necessary to develop more intuitive keyboard methods and, if possible, reduce the number of key presses by the user.

3.2 Interpreting characters and rendering glyphs (context analysis)

Character coding standards (including Unicode) do not define glyph images, they define how characters are interpreted, not how glyphs are rendered. In this case, a special program called "Context Analyzer" is necessary for rendering glyphs on the screen. For example an abstract character such "ARABIC LETTER AIN" which has the U+0639 Unicode value can have different visual representations (called shapes or glyphs) on screen or paper, depending on context (see figure 2). Different scripts which are part of Unicode can have different writing rules for rendering glyphs and also composite characters, ligatures, and other script-specific features. The results on screen or paper can differ considerably from the prototypical shape of a letter or character. For example:

- Greek: σ (beginning and middle of a word), ς (word ending)
- German: s + s => ß
- Arabic:

The ع character is written:

- ع at word beginning (science علم)
- ا inside the word (institute معهد)
- ع in the word ending (group جمع)
- ع when it is not linked (section فرع)

Figure 2. Arabic character associated glyphs

3.3 Ideogram phonetic input

Certain languages, such as Chinese, are based on a vast number of ideograms, each one representing a particular concept. Recently (1980), simplified and varied versions of Chinese have been adopted by the People's Republic of China, on one side, and by Taiwan and Hong Kong, on the other. Several different keyboard input methods exist, such as input via

character codes (e.g. **GB-2312-80** code) or the **Pinyin** method (see figure 3). , which consists of a phonetic representation of ideograms in Roman characters (420 syllables, complemented by one of five tones per syllable).

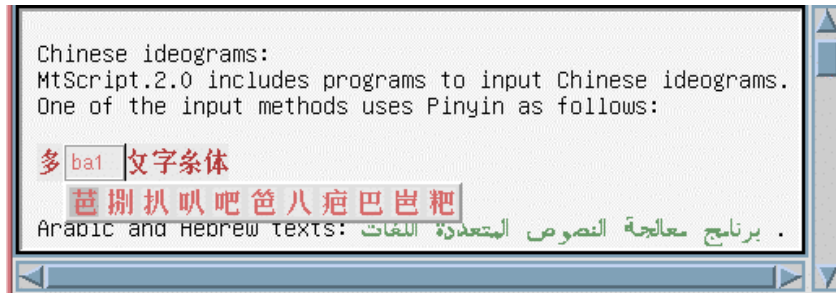


Figure 3. Chinese ideograms keyboard input using Pinyin

4 Multilingual text coding

4.1 Character coding

Computer manufacturers and software developers use numerous specific and non-compatible character codes (MS-Windows character set for Western Europe MS CP1252, DEC Multinational character set, International IBM PC character set IBM CP850, Macintosh Extended Roman character set, Hewlett-Packard ROMAN8, etc.). Meanwhile other character coding norms have been standardized on an international level and are already used in some environments (ISO-8859-1/2/3/4 for Roman Alphabet, ISO-8859-5 for Cyrillic, ISO-8859-6 for Arabic, ISO-8859-7 for Greek, ISO-8859-8 for Hebrew, GB-2312-80 /BIG 5-0 for Chinese, JISX-0208-1983-0 for Japanese, KSC-5601-1987-0 for Korean, ...).

More recently (1993) the ISO 10646 (*Universal multiple-octet coded character set or UCS*) proposed a universal character set including all the character sets of ISO-8859 as well as those for Chinese, Korean, Japanese, the International Phonetic Alphabet, etc. In its present form (ISO 10646-1), the UCS uses a 16-bit code (Unicode) which will be extended to a 32-bit one in future editions, thus permitting an effectively unlimited coding of characters. However, existing environments are not yet ready to implement character sets on multiple-octet code, even though the situation is rapidly improving (e.g. Windows-NT, AT&T Bell Plan 9 and Apple QuickDraw GX). Moreover SGML entities have been defined for encoding the characters of many different languages. SGML is being a standard for the multilingual document interchange.

4.2 Writing systems coding

In a multilingual text it is necessary to code not only individual characters but also scripts (Latin, Semitic, ...) and languages. In the case of a one-octet-based coding (e.g. ISO 8859-* character sets), it is necessary to mark the change from one set to another (e.g. changing from Greek to Cyrillic). This can be done using a code such as that proposed in the ISO 2022, which includes escape sequences (<SI> (shift in) and <SO> (shift out)) that encode a transition between the "main" and the "complementary" sets. However these techniques are limited and many difficulties can arise, especially when a single document includes one-byte (e.g., ISO 8859-*) and two-byte (e.g., GB-2312-80 or BIG-5-0 for Chinese, JISX0208-1983-0 for Japanese or KSC5601-1987-0 for Korean) characters. The UCS inventory solves one part of the problem by combining all these character sets into a single set, since it is no longer necessary to implement a means for switching between character sets. However the problem is not totally resolved because UCS does not explicitly encode some features of the character sets such as the writing direction (although bi-directional protocols have been proposed by the Unicode Consortium). Moreover language tagging is needed not only to indicate writing direction, but also to control hyphenation, ligatures, font selection and character/glyph mapping.

4.3 Language coding

Linguistic processing of a multilingual text (segmentation, morphological and lexical analysis, etc.) requires the identification of the languages therein. Recognizing the character set or the writing system does not suffice to identify the language in which a portion of text is written: a document encoded in ISO 8059-1 could equally well be written in French, English, Spanish or even a combination of these languages.

Standards for coding the names of languages exist:

- • ISO 639-1988: 2 alphabetic letters code for about 140 languages (e.g. "en" for English, "fr" for French, etc.).

- • ISO 639-2: 3 alphabetic letters code, alpha-3, is currently in development (e.g "eng" for English, "fra" for French, etc.).

However, in the internal code of a document, these codes cannot be used such as they are. At this time there is no established standard method for escape sequences which would permit the representation of the change from one language to another, although it has been proposed that one use the ISO/IEC 6429 set of control sequence codes with a numeric conversion of the above alphabetic codes [2]. Language markup is also currently being defined in the SGML/HTML standard used by the World Wide Web [3].

5 Multilingual text editing

5.1 Bi-directional texts

The majority of languages are written horizontally from left to right. Some languages, such as Arabic or Hebrew, are written from right to left. Other languages, such as Chinese or Japanese, can even be written from top to bottom (especially in ancient texts). As a consequence, the co-existence of languages in the same document, and particularly on the same line of the text, poses huge problems when inserting or deleting text zones. The example in figure 4 shows that it is often necessary to rearrange words to maintain the semantic coherence of a sentence.

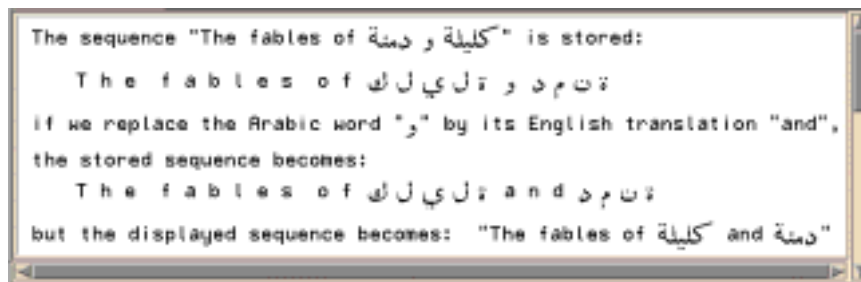


Figure 4. Editing aspects in a multilingual text

5.2 Insertion / deletion of characters in bi-directional texts

A significant problem with displaying multilingual texts is the co-existence of opposite writing directions on the same line of text. Insertion and deletion of characters must take into account their writing direction, according to quite complex rules. The text editor may allow the user to define interactively a main and a secondary writing direction for a text-zone (paragraph). The cursor moves only in the direction specified as the main one. When a sequence of characters is entered in a language written in the secondary direction, the cursor stays put and the characters are written in an insertion mode (see figure 5).

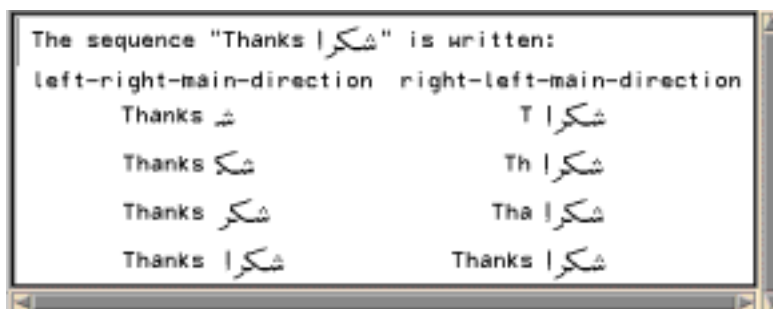


Figure 5. Bi-directional texts

5.3 Multilingual text justification

The justification of right-to-left written texts or bi-directional multilingual texts is a major problem. Contrary to Latin alphabet based texts which can be justified by using extra spaces, the justification of cursive right-to-left written texts requires the use of **dynamic fonts**, where most of the characters can be contracted or stretched dynamically, according to the line length. In the example of figure 6, the last character “Ba“ of the word “KaTaBa“ (to write) is drawn differently according to the line length. A special character such as “_“ can also be used to extend the character drawings.

کتب
کتب
کتب

Figure 6. Dynamic font for Arabic text justification

6 Multilingual text printing

Printing multilingual texts suffers most obviously from the lack of printer fonts (essentially PostScript fonts). Many PostScript fonts are now available (freely or not) for Roman characters, but only a few fonts have been developed for the other character sets. Significant efforts in designing multilingual Unicode-based PostScript fonts for editing and printing multilingual electronic documents include particularly the OMEGA project activities.

7 Multilingual data exchange

With the rapid growth in the use of the Internet, the electronic transfer of multilingual documents is becoming more and more necessary. Until recently, only one part of the standard invariant characters of the ISO 646-IRV (ASCII) could allow a non-corrupted electronic text exchange, and multilingual documents could be transmitted safely only with the assistance of coding utilities such as *UUENCODE* and *BINHEX*. However the situation is improving: standards have been adopted on the Internet which allow the transfer of 8-bit characters without corruption in the TCP/IP protocol (for example, applications such *TELNET* and *FTP* are "8-bit clean"). In addition, the MIME norm (*Multi-purpose Internet Mail Extension: RFC-1521 and RFC-1522*) allows uninterrupted data transfer in any circumstance by compressing and decompressing the files. Moreover the emerging general standard for text data interchange is SGML (and for certain areas TEI), although these standards are not yet universal and some transfer problems persist. In saying this, one must point out that the current guarantee for data transfer without corruption does not extend to the transfer of multilingual data. It is necessary that both parties involved in the transfer, the sender and the receiver, have the same systems of encoding characters, documents, languages and writing systems.