Project P923-PF

Multilingual WEB sites: Best practice, guidelines and architectures

Deliverable 1

Guidelines for building multilingual Web Sites

Volume 1 of 5: Main Report

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 0007-0923

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.

Preface

Multilinguality is quickly becoming a major issue for the European Telecommunications Companies. Telcos are developing into full-service companies, that offer a wide range of services (e.g. e-commerce) via the Internet. Many Web services must be offered in several languages. The design and maintenance of multilingual Web sites require tools and procedures well beyond what is needed for mono-lingual Web sites. Without suitable tools - based on standardised architectures for multilingual Web sites - these sites and the attendant services are very expensive to create and manage.

In previous R&D projects in the field of Web and language technology it has appeared that truly multilingual developments are only possible if companies and research groups from different countries representing different languages collaborate. Therefore EURESCOM was the perfect frame to undertake studies in this area, and EURESCOM shareholders will reap substantial benefits from them.

Though the technology in that area is not expected to become completely mature before a few years, the speed of development is increasing. Thus telecommunications companies which have experience with cost-effective procedures for developing and maintaining multilingual Web sites will be in a position to acquire a considerable share of this emerging market.

The main focus of EURESCOM PROJECT P923 is the development of best practice guidelines for the design of multilingual (and therefore essentially also multicultural) Web-based information and transaction services.

The key results to be provided by the project include:

- a generic architecture for multilingual Web sites that are easy to design and maintain.
- best practice guidelines for building and maintaining multilingual Web sites.
- inventory of existing tools for multilingual text production and knowledge about their suitability for aiding the creation and maintenance of multilingual Web sites.
- practical experience with the creation and use of several multilingual Web sites.

This report is intended to address the first three points and to be used as input to the last part of the project which will take care of building a demonstrator as well as summarising the lessons learned in a second and last report ‘Experiences in designing and building multilingual web sites’ in Q4 2000.

The project has five participating companies and is lead by Els den Os from Koninklijke KPN N.V.

Executive Summary

As small businesses and large corporations alike attempt to gain the largest market possible for their products, they naturally have to address the global market place. Fierce competition in every business sector means that the businesses that only address the national market will find that they are too inefficient.

There are various barriers to addressing a global market including legislation, logistics, language and culture. In this document we are concerned with the methods of providing a multilingual WEB-based service. The scope of this document includes architectural, design, procedural and linguistic issues, but excludes legislative and logistical issues.

The key to creating a service or product, which can be cost-effectively localised, is for the whole design and development team to think multilingual right from the beginning. It is extremely easy to unconsciously make assumptions, which will increase the cost of porting the system to another locale. For example, poor choice of character representation at the outset of a project may demand significant re-engineering if it does not support locales, which are later needed. Another example is that it is common practice for a programmer to embed strings within source code. Some primitive text processing is often carried out such as appending two texts to make a sentence. This tendency to embed locale-specific information within programming logic is not tenable in a multilingual environment.

From a design point of view, there needs to be the cleanest possible separation between different kinds of data. Ideally, all locale-specific and locale-independent data and procedures would be stored separately. They should also be further subdivided according to the kinds of skills required to maintain and localise them.

One of the major expenses in localising any product is the cost of translation. Automatic systems currently perform very poor quality translations when compared with professional human translators. However, due to the cost and slowness of manual translation, automatic translation systems do have their place. If text is updated frequently, is domain-specific and is linguistically of the appropriate type, then automatic translation systems can produce sufficiently good translations.

Other possibilities for reducing the cost of translation include carefully designing the Web site to be global that is to contain only a small number of elements which cannot be directly used for other locales.

Other language tools are already in use on monolingual Web sites. For example, document indexing and retrieval systems are now very common. These can become considerably more complex in a multilingual environment, especially if cross-linguistic retrieval is required. For example, we might wish to enter a query in one language and yet have the system retrieve relevant documents that were written in a different language. We may also wish to have the retrieved documents automatically translated.

List of Authors

Luís Almeida (PT)

Stephen Appleby (BT)

Nuno Beires (PT)

Malek Boualem (FT)

Louis Boves (KPN/University of Nijmegen)

Gloria Branco (PT)

Maurizio Codogno (IT)

Carolina di Cristo (IT)

Els den Os (KPN)

Marta Pombo Prol (BT)

Jérôme Vinesse (FT)

Table of Contents

Abbreviations

ASP	Active Server Pages
CAT	Computer Aided Translation
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
DHTML	Dynamic HTML
HTML	Hypertext Markup Language
PHP	Parsed Header Processing
SSI	Server Side Include
WWW	World-Wide Web
XML	eXtensible Markup Language

Definitions

Locale

A locale refers to a collection of people who share language, writing system and any other properties which would require a separate version of a product. The way the World's population is partitioned into locales will depend on the details of the product.

In the software and information technology industries, the term is used to refer to the collection of procedures and data that vary from one localised version of a product to another.

Internationalisation

Internationalisation is the preparation of a product so that it can be customised for particular locales efficiently.

Globalisation

Globalisation is the design of a product so that it remains the same for all locales.

Localisation

Localisation is the customisation of a product for a particular locale.

1 Introduction

The idea for the EURESCOM project ‘Multilingual Web sites: Best Practice and Guidelines and Architectures’ (P923) was the result of a pre-study project that was also conducted in the framework of EURESCOM (P814: Pre-study on Speech-to-Speech Language translation). This pre-study gave an overview of speech and language technologies and possibly interesting services that can be supported by these technologies. Two project proposals were formulated at the end of the project, of which this one on multilingual web sites was finally approved by the EURESCOM board.

The aim of this project is to develop ‘best practice guidelines’ for building and maintaining multilingual Web sites and services.

We made a detailed investigation of the needs of multilingual web services within the Telco’s that participate in this project (IT, KPN, PT, FT R&D, BT). In addition, detailed surveys were made of

- Available tools and procedures for managing (multilingual) web sites
- Language related tools, like Machine translation, summarisation, etc.
- Possibilities to add voice input/output to web sites e.g. VoiceXML

Although we included an overview of the voice input/output issues, since this might become extremely important for Telco’s in relation to voice portals, the scope of the project does not lie on this topic. The next phase of the project was to define a number of possible services that can be built in the second phase of the project. These services have to be relevant for testing as much multilingual aspects as possible, including maintenance and usability of web and language tools. In this deliverable we present an overview of this first exploratory phase of the project. It gives a clear overview of possible ways to address the problem of multilingual web generation and maintenance. The target audience of this deliverable are web designers, product managers of web sites, and technical web builders.

.2 Why make your WEB site multilingual?

Telecommunication companies are developing into full-services companies that offer a wide range of services via the Internet. Many web services must be offered in several languages, since many Telcos will be active in other countries than their own, and web customers are global customers.

From reviewing existing multilingual sites and by interviewing web and service managers within the five network operators of this project, it is very obvious that multilinguality on the Internet is quickly becoming an important topic for all service providers in the world. Multilingual web sites may cost a lot of money and effort, since for most services it is inevitable that human translators are involved. Our review and interviews showed that presently localisation often is an ad hoc process.

At this moment the Internet contains millions of web sites and about 85% of them are in English. However, it is expected that the non-English speaking web users will outnumber the English-speaking users already in 2000 (see Figure 1). Thus, it is no longer enough to translate local web sites only to English.

In 2005, one expects the Web to reach one billion users and even 70% of them will be non-English speaking.

It means that much effort has to be put into localisation of existing web sites and into the creation of new multilingual services, since it is certain that most web users prefer to be addressed in their native language, at least at the top-level pages of services. These top-level pages need to be perfect, since otherwise the risk is high that the customer will gain a poor impression of the company. In the multilingual Internet the motto “The competition is just a mouse click away” is very true indeed. Forrester mentions that customers, who are addressed in their own language, will stay at a site twice as long. In addition, they will spend tree times as much money when they can use their own language. Given the enormous growth in web population, it is clear that you can earn much more money when you can offer the right language.

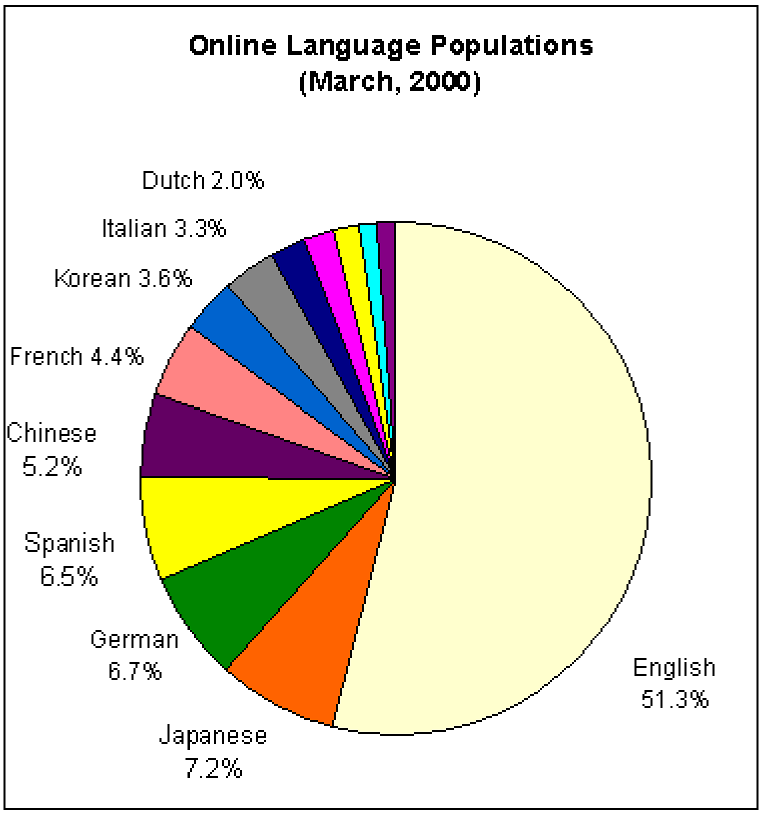

Figure 1
Online Language Populations

3 Multilinguality Framework

This chapter introduces a high-level overview on “a framework of the things to be considered” when addressing multilingual services on the Internet. It provides a perspective on a multilinguality framework that can be used to *set the scene* and to better understand the guidelines and contents presented in the remaining chapters of this deliverable. The multilinguality framework considered by the Babelweb Project corresponds to a group of human-related roles that interact with another set of roles (or functions) provided by technology-oriented capabilities in order to deliver multilingual web-based services.

3.1 Roles in Multilinguality

When building a web site, be it mono or multilingual, it can be useful to consider it as the composition of a series of *roles*. In this context, a "role" is simply a logical function that interacts with other roles and has an internal coherence of its own when considered in the global framework. No further commonality can be found among different roles; in fact it is possible to view roles to be played by persons (user, designer, etc.) and roles to be played by technology capabilities (tools, content resources, etc.). The focus here is much more on the “roles” themselves than on the “actors” (or the players). This is because actors may perform different roles depending on various circumstances external to our multilinguality problem domain (e.g. type of company, size or positioning on the Internet market for humans and product range or level of coverage for tools).

During its studies the Project has identified a list of roles and their interactions belonging to the multilinguality framework. Some of these roles are of course general, but some others are specific to multilingual web sites and some others may provide other functionality. The following human-related roles are included:

- Multilingual Web Designer
- Builder
- Translator
- End User
- Web Manager

The following technology-oriented roles are considered in the multilinguality framework:

- Browser
- Web Support Tools
- Language Tools
- Content Management Tools

These roles and the relations between them can be observed in Figure 2. A brief explanation of each role and its relationships with other roles is presented below.

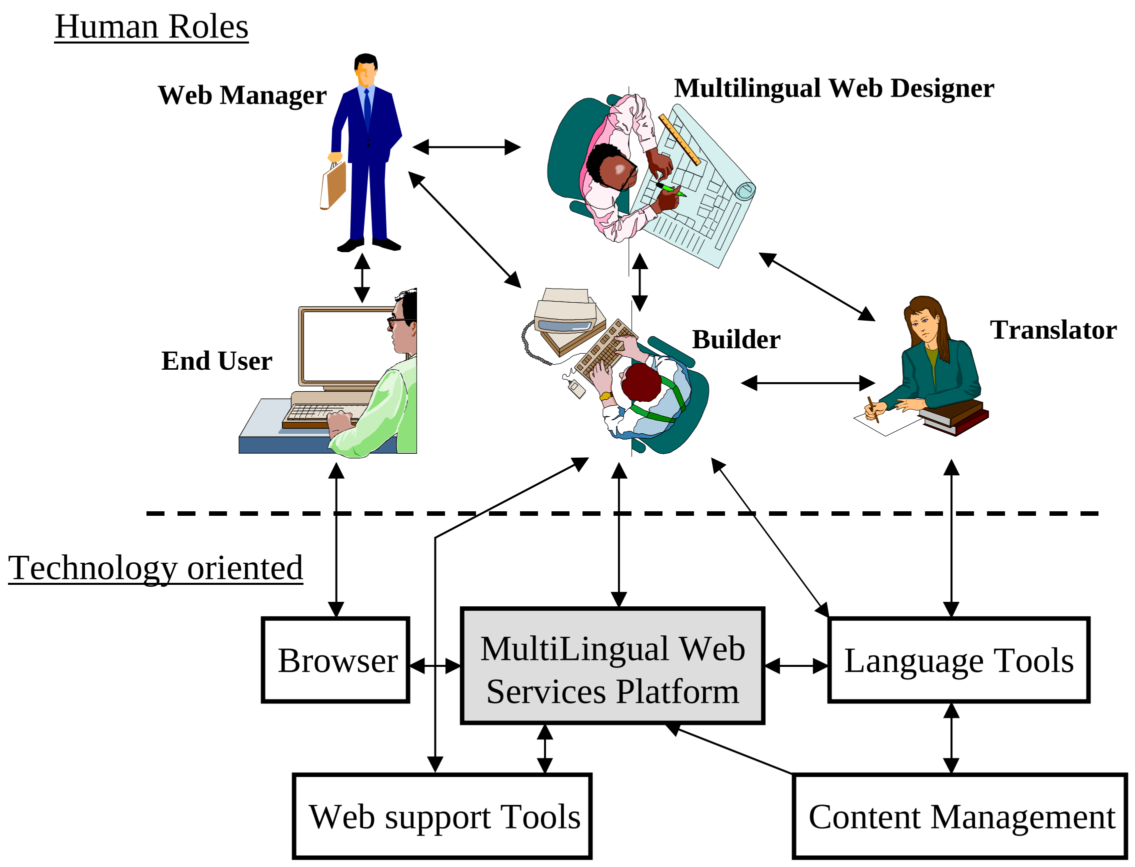

Figure 2
Relationship between the different roles associated with a multilingual Web site.

3.2 Human-related Roles

3.2.1 Multilingual Web Designer

This is the person that sets up the structure of the web site. In the case of a multilingual site, the web designer must define how the various language versions fit into the overall structure. Moreover, the designer must take into account the fact that localised pages may be created in different ways, and sometimes have a different layout due to language constraints. The web designer should define and organise the web site having in mind a set of rules that facilitate the information localisation process and help to decrease the overall cost involved. An example of such rules is included below:

- leave text outside graphics,
- provide source graphics with layers and fonts,
- use a database system to allow a separation between navigation and contents,
- leave sufficient space for translations,
- separate strings from code when using scripts,
- use variable names that are not words in the scripts,
- avoid the breaking of sentences inside the script code;

From the rules presented above we can retain the idea that the solution found by the designer should separate the language-independent content from the language-dependent content that requires translation to all languages supported by the web site.

During the conceptualisation of the web site, the designer should analyse carefully the content of each page and decide which are the most adequate language resources and tools to translate the page. There are some aspects that must be considered such as the complexity of the text, the data-update rate and the page access rate to the pages of the site. If the page contains only words or short sentences a machine translation could be used, corresponding to an on-line translation of the content. On the other way, if the page contains complex text a human translation is better recommended, which corresponds to an off-line translation. The data-update rate is also a criterion for the designer to decide whether to use on-line translation or off-line translation. If the pages are updated at a high rate and the content is composed by short sentences an on-line translation tool can be used to reduce the maintenance costs of the web site. Usually the pages in a web site are organised hierarchically and the pages located in the top levels are accessed more often. For these pages it is recommended to use off-line translation, since it is not possible with the current automatic translation tools to guarantee a good quality translation.

The use of on-line, off-line translation or a combination of both allows a complete parallelism making all documents available in all languages. However there are situations where it is not possible to achieve a complete parallelism for all the languages due to economic reasons or the target users profile. The other extreme of the parallelism is the degree to which all the information is available and accessible in the same form independently of the language.

The designer interacts with the builder, giving him actual instructions about how to build the site, and with the translator, to get a feedback about the localised pages and have an idea about the needs of the site users.

3.2.2 End User

It is not immediately clear that the end user also plays a role; but, of course, a site is useless without them! They play an active part in the global scheme, since they access the site and choose the preferred language, maybe switching between languages while browsing the site. The only direct interaction that end users have is with their browsers.

3.2.3 Builder

He is the person who actually builds the site. Note that he does not necessarily understand every language involved, which in fact is quite common. The builder may however notice flaws in the practical implementation of the structure of the site, interacting therefore with the web site designer. He also interacts with the translator, exchanging information about the rendering of the site, and may directly interact with the language tools if needed.

3.2.4 Web Manager

The web manager has to guide the work of the Web developing team (Web designer, Builder) according to the business goal of the site taking in consideration the opinions and suggestions expressed by the end user during the contacts established between them. The web manager is also the person responsible by the maintenance and the update of the web site. This role is particular important in a multilingual web site context due to the complexity of this kind of sites compared with monolingual sites, as often it requires a more demanding maintenance and updating policy. There are some aspects that can influence such policy: the information update rate, the structure of the web site (translated off-line pages versus translated on-line pages), and the degree of parallelism between the languages supported in the web site.

3.2.5 Translator

The role of the translator consists of course in translating the pages of the site in the various languages. In fact, this role may be acted either by a person or by an automatic system. In an actual site both actors will be probably present, unless the site is small or rather static, in which case only the human being is necessary. This may depend also on the type of information on the site and on the adopted translation approach: either off-line or on-line (automatic) translation or a combination of both. The translator has a strong interaction with the language tools, other than with the designer and the builder. Noticeably, he does not have any direct interaction with the site itself.

3.3 Technology-oriented Roles

3.3.1 Browsers

It may not be obvious why there is a browser role in a (multilingual) web site. However, the answer is simple: the browser is the interface towards the end users, so it mediates their perception of the site. A browser must therefore be capable of supporting the character sets for the involved languages. The browser interacts with the end users and with the site.

3.3.2 Language Tools

These tools are necessary for the translator, if he is a human being, to help him building up the localised versions. Examples of such tools are online dictionary and context-based translation tools, which offer the translator a suggestion for a version of a sentence, based on the similar occurrences in the previous contexts. There are some other tools like machine translation that can be incorporated in the web site to support automatic translation of the pages without human intervention. Machine translation tools may be combined with on-line dictionaries and context translation tools to achieve a translation with higher quality. There is an interaction between language tools and content management tools, since the former may access a database of translations. Builder and translator may also use those tools.

3.3.3 Content Management Tools

The instruments that belong to this role are those which are necessary to manage all the information contained in the site. Examples of these tools are database system, which tie together the versions of the same logical page; and the web editing languages, like PHP or the ASP system, which act as an interface between the databases and the appearance of the pages. The only interaction of these tools is the one with language tools, and with the site itself. By this way it is possible to achieve a separation between the content and the design. Additionally, content management systems can be used to keep track of what files and what database content has been modified and has to be translated. Alternatively a file exchange format can be developed to check out files from the content management system, and check in them again.

3.3.4 Web Support Tools

In this broad category we find all those tools which are necessary to build and maintain the site, and which do not belong to the other roles. The most important tools in this role are the web editors, which must accept text in the various languages involved and if possible must allow a side-by-side editing, to check at a glance the parallel versions. These tools interact only with the site itself.

Comprehensively, the multilinguality framework presented above is very much centred on the Multilingual Web Designer role since that was the approach followed by the Project. The main objective was to produce a set of focused guidelines that may structure the multilinguality problem domain and offer guidance to the designer in delivering effective multilingual web services.

4 Guidelines for Multilingual Services

The primary aim of these guidelines is to simplify the management of a multilingual WEB-based service. In practice, a large part of this consists of attempting to maintain a separation of different kinds of information that constitute a multilingual WEB site. The information should be separated roughly according to the roles of the people who have to deal with this information. This means that say, translators should not find they have to edit JavaScript, and graphic designers should not have to work in several languages.

The various kinds of information present on a WEB site are:

- Formatting
- Content (text, graphics etc.)
- Navigational information

Even for monolingual WEB sites, we would wish to maintain the cleanest separation possible between these different kinds of information. One of the criticisms of HTML is that it does not provide any mechanism to separate content from format. In the case of multilingual WEB sites, we have the additional dimension that each of these may contain locale-dependent and locale-independent parts. The following table summarises the situation.

	Locale-dependency
Formatting	Locales with similar scripts are likely to share the same format. E.g. all Western European locales could use the same formatting. However, scripts that use, say, Han characters will most likely require a significantly different format.
Content	Content is normally highly locale-dependent. However, this is not always the case. Images might be global, and in some cases a kind of "globalised" English might be used for the text. For simpler sites, it may be more efficient not to separate locale-dependent content from formatting.
Navigational information	This depends very much on how parallel the different localised versions of the WEB site are (see below).

Table 1
Factorisation of information common across locales.

Ideally we would like to maintain a clear separation of formatting, content and navigation and for each of these separate the locale-dependent parts from the locale-independent parts. In practice this is unlikely to be achievable. For very small WEB sites, the overhead of setting up an infrastructure to maintain separation may be too great and it may be more appropriate to allow some overlap of different kinds of information. For larger WEB sites, the management costs will be greater, and therefore there will be more benefit from maintaining separation in this way.

4.1 Parallel vs. non-Parallel WEB sites

The different language versions of a WEB site may exhibit various degrees of parallelism. At one extreme, we could have each version of our WEB site so dependent on locale that its localised variants bear virtually no resemblance to each other. At the other extreme, it could be that the information on every page is localised and the relations between them are preserved, making the sites exactly parallel. It could be that the structures of the sites are identical, but that certain information is only relevant to particular locales. For example, certain legal disclaimers may be required for certain countries and not for others.

We may identify a scale of parallelism (of course there could be finer sub-divisions):

1. Completely independent sites for each locale,
2. Parallel structure to the sites, but the information present on each page is completely different,
3. Parallel structure to the sites, but the information present on each page is slightly different,
4. Parallel sites, but where some pages have not been localised (e.g. due to cost),
5. Completely parallel sites with identical structure and information.

This document does not address (1), since these are not really multilingual sites, but a collection of independent monolingual sites.

In case (2), we will require that all information is generated separately, but that the navigation and storage structures remain the same. In order for the sites to have a similar storage or navigation structure, it must be that each localised version of a page plays a similar role, even though the information presented by each localised version is different. For this to be apparent, the pages must be labelled with the role that they play (perhaps by appropriate choice of filename).

In case (3), the units of information must be smaller than a page. Otherwise it will not be possible to indicate what information should be shown for a particular locale and what should not.

Case (4) is extremely common. Often a commercial WEB site will have the top-level pages professionally translated, but will not have the lower level pages translated. This is quite straightforward to deal with by having a default locale (or a series of preferred locales).

Case (5) is the most straightforward from an architectural point of view, but less common than the previous case.

When different localised versions of WEB sites contain non-parallel information, there may be some impact on the navigation. If the targets of navigation links are pages and all pages have corresponding versions for each locale, even though the details of the information present may not be the same, then there navigation will not be disrupted. It may be that entire pages are present for certain locales, and they do not exist at all for others. In this case, it is potentially very easy to have broken links.

To overcome this, and facilitate management of the site, it is important to know what the 'atomic' units of information are, which units are relevant for which locales, which units are present for which locales, and what the navigational structure is that relates these units.

4.2 Locale-independent navigation

When trying to manage a complex multi-locale Web site, it is useful to factor out those elements that are common between the different localised versions of the site. When the different localised versions are highly parallel, then they will share much of the navigational model.

We would ideally like to completely divorce the navigational model from all other aspects of the Web site so that this could be managed independently of locale. This would allow us to specify policies in a locale-independent manner. Due to the nature of hyperlinks though, complete separation is impossible without placing undue restrictions on the way hyperlinks can be used. For example, we may wish to attach a hyperlink to a particular word in one language, but we will of course need to attach it to a different word (perhaps the translation of the original word) when the document is translated into another language.

In the case of a simple, monolingual Web site, one would embed each link in the HTML code and indicate the target page (and optionally location within that page) as a property of the link. If the page is to be translated to another language, the link will be associated with different text and the target of the link may also be changed to refer to the page in the appropriate locale.

The fact that the link will be associated with different text for each locale means that we cannot make all navigational information locale-independent. However, we could use a locale-independent indication of the target of the link and allow some other mechanism to decide which localised version of the target should be displayed. When language negotiation is used, this is effectively what happens. However, there are various reasons why it may be undesirable to rely on language negotiation as the complete solution.

One possibility would be to insert anchors for links into the text as usual, but instead of referring explicitly to the target, they are de-referenced through a locale-independent table. This would allow the behaviour of links to be made explicit through the use of rules.

Possible triggers for different behaviours are;

1. Does the target exist in the preferred locale?
2. Is the target part of our site or an external site?

Possible behaviours are;

1. Remove the source of the link
2. Display the target information in a representation for some other locale than that selected (perhaps using language negotiation, or having some default locale for the site).
3. Pass the target page through a utility such as a translation tool or a summarisation tool.

So, an entry in the link table could be,

link1 => translate(eng, http://www.someone_elses_site.co.uk/page1)

Whenever anyone clicks on this link, they would be presented with a translation of the target page into English.

4.3 Separation of Content from Formatting

As stated in the previous section, it is desirable to factor out that information which is repeated multiple times in order to facilitate its management. In multilingual WEB sites, this means that we should attempt to separate out all information that is the shared by several locales. Generally speaking, even for monolingual WEB sites, we should try to separate formatting from content. For a multilingual WEB site this is even more important, since we will often wish to share styles amongst several locales.

One of the criticisms of HTML has been that it provides poor separation of content from format. With dynamic HTML this is much less the case, since style sheets can be used to give sufficient control over formatting for many purposes. Style sheets can be stored separately from the content of the page and shared by any number of pages.

This is certainly an improvement on putting the font details etc. directly into HTML files, but we really need to do much more than separate details of the style from HTML files. In the case of multilingual WEB sites, we need to take into consideration that translators will work with the content files. Therefore, we would like the content to be as free from markup instructions as possible (although it may be said that, if a particular word is made italic in the source language, then the translator will have the responsibility of choosing which word or phrase is made italic in the target language).

A reasonable solution is that complete HTML pages are built by combining units of content with an HTML template (using a style sheet as well if necessary). XML provides one way to do this, but it is quite possible to achieve this separation without the use of XML.

Units of information that have been localised should be labelled with a locale-independent information label and the locale label. This will make it much easier to identify content units that are translations of one another automatically.

One of the problems with separating formatting from content, is that different language texts will be different in length. It is important then, that the presentation process can properly display strings of different length in the same fields.

4.4 Active client components

As well as presenting information, WEB pages can contain various other components, such as scripts, Applets etc. These too, will need to be subject to the same principles of multilingual management as any other kind of content.

It is useful to distinguish between compiled and non-compiled information. The localisation of compiled components is outside of the scope of this document since there will be various mechanisms available for internationalising these components depending on the language they are written in and the development environment.

When components are not compiled, they may essentially be considered as normal content from the point of view of multilinguality. For example, a JavaScript component may contain data and procedures that are locale-specific. These should be treated as any other textual content, with locale-independent parts of a script being kept separate from locale specific parts.

Where there are locale-independent scripts, these will be stored in the same way as all other locale-independent content. As mentioned above, it may be practical for small sites to allow locale-independent information to be stored as part of the formatting. For larger sites, it will be more efficient to maintain a clean separation of formatting from content and so scripts should be stored separately from formatting information.

For larger sites, it will be useful to be able to separate the locale-dependent information according to what kind of skills are required to localise it. For example, a JavaScript that is to display the date will be a localisable element. Generally, translators do not have the skills to localise JavaScript and therefore any such scripts should be labelled in a way that distinguishes them from say, ordinary text (for example, by using a context-sensitive editor which is capable of recognising JavaScript).

4.5 Active server components

Active server components includes CGI scripts, Server Side Includes and a variety of other mechanisms for processing or generating WEB pages before they are passed from the server to the client.

As with active client components, a distinction may be made between compiled components and interpreted components. Compiled components are out of the scope of this document.

Any scripts which are stored with the HTML page should be treated in the same way as active client components mentioned above. Scripts which are stored separately could again be managed in the same way, by removing all locale-specific parts of the script and putting them in separate files for easy management.

4.6 Off-line vs. dynamic WEB page creation

The idea of creating the WEB site off-line by combining data elements with HTML will work for simple WEB sites which are designed to present information that is updated infrequently, but it is not appropriate in certain circumstances. A static WEB site would not be appropriate where the content of the WEB page is generated automatically, perhaps in response to a user query, or where the data needs to be updated quickly or asynchronously, so that batch mode updating of the site in not feasible. It is also impractical where a very large number of pages would be created. For example, it would not be appropriate to generate a static HTML page for every book in amazon.com!

Dynamic creation of WEB pages is possible by executing code on either the server or the client. There are a number of technologies available for both (see Annex A, sections 5: Dynamic HTML and ASP). Where WEB pages need to be built using data that is held on the server, server-side code is the best option. However, it does place a far greater demand on the server, which can result in very poor performance. These performance issues are being addressed. For example, ORACLE claims to have an efficient system of using "Cartridges" which can also delegate processes to other machines. Microsoft's IIS WEB server uses lightweight processes that are managed by the server itself to reduce start-up times and memory overheads. Apache provides for lightweight CGI scripting using a built-in Perl module instead of running an instance of the Perl interpreter each time a page is accessed.

From a multilingual point of view there are considerable benefits from using dynamic generation of WEB pages. It allows the possibility of controlling locale in a dynamic way, dependent on whether pages are present or not and allowing pages to be generated from on-line data.

A server-side process could be used to maintain the separation of locale-specific information from locale-dependent information, just as in the case of static WEB sites. That is, resource identifiers can be used to mark areas in HTML files into which locale-dependent text must be inserted. Separate files can be used as resource files that contain the values of locale-dependent resources.

Any server-side process can do this and much more besides. One possibility is that a JavaScript menu is present on (at least) the home page and is used to set the language and any other locale preferences and store them in cookies. Each time a page on our WEB site is accessed, the CGI script queries the cookies to determine the parameters of the page that should be returned.

Dynamic generation of WEB pages also allows the possibility of processing pages external to your WEB site before presenting them to the user. This is similar to say, the AltaVista WEB site, which allows you to translate a WEB page at a given URL. If you click on a link on a translated page, you will be presented not with the page itself, but with a translation of that page.

There are many cases where HTML pages are not built directly, but are created from structured data. The data typically comes from a database, but may come from a real-time feed (e.g. stock market information, weather etc.).

A typical application would be a catalogue for an e-commerce system, where the user can browse or search the catalogue for a particular item and the appropriate page will be displayed.

When an HTML page is to be created from data, there will be an HTML template which will include some instructions indicating where the data should be inserted and how to get it. This is the almost standard model used by Active Server Pages. It may be that the template itself is locale-dependent, and it will certainly be the case that the information to be displayed is locale-dependent.

Dynamic generation of WEB pages allows considerably more flexibility than the static pages described above, albeit at the cost of greater complexity and slower retrieval of pages. In this situation we can make use of cookies to store and act on user preferences.

There is also the possibility of processing other sites' pages before presenting these to the user. For example, if it is decided that information from other sites should be translated before being presented to the user, then this can happen. The translation service on the AltaVista WEB site is such an example. Here if you request that a page at a particular URL be translated, then all the links on the translated page will be modified to pass through the AltaVista site for translation as appropriate. This is quite simple to achieve with a dynamic WEB site.

4.7 Language Negotiation

Language negotiation is where the user sets their preferred languages, in order of preference, in the browser. The browser then "negotiates" with the server to attempt to find a page in a suitable language.

If we are to rely on language negotiation, we cannot allow users to choose their own language from a control on the page. This is because language negotiation requires that the URLs of the pages requested do not indicate language, but that language preference is taken from the preferences set in the user's browser. If the browser requests a language-specific URL, then this will override language negotiation.

It may be difficult to rely on language negotiation since it may not be set correctly. If language negotiation is to be used, it would be useful to indicate this explicitly on the front page of the site, along with some instructions for setting language preferences, otherwise the user may never know that other language versions are available.

Ideally, we would like a control to be present on our WEB pages which allowed the user to select their language preference. This control would in turn set the language preference in the user's browser. Unfortunately, it is not possible to set the language preference in this way (in Netscape's Communicator, it is not even possible to read the language preference).

If we are not relying on language negotiation, then each parallel version of HTML pages (on the live site) will not only differ in text and resource values, but the links that they contain will need to be different so that they point to files in the appropriate language. This effectively overrides language negotiation and so once the user has selected a particular language, language negotiation will be ineffective. It would still be useful to store HTML pages in a way that is consistent with language negotiation though, since at least the page that the user first visits on the site will be selected correctly. However, if the user changes their language preferences on their browser whilst viewing some WEB page, the page will still display in the language that they chose on the original page. The only way to change language preferences would be to make a control available on every page.

If language negotiation is considered to be acceptable from the user's point of view, then this would make it very much easier to build the WEB site.

Even if language negotiation is used, it does not preclude the inclusion of other user preference information in the localisation process. For example, suppose we need to know the user's actual location as well as their language to decide which page to present. This can be done by including a separate control for the user to select their location. The batch process will need to set the links of the HTML pages to refer to those for the appropriate location.

4.8 General Guidelines for HTML files

As far as the organisation of the HTML files themselves is concerned, there are some general guidelines that are even more important in a multilingual environment.

The HTML code should have the appropriate META tags set. In particular, language, locale and character encoding. If possible, the character encoding should be UTF-8, although older versions of browsers may only support native encodings since the standard for HTTP is ISO-8859-1, which only supports 256 characters.

Cascaded Style Sheets (CSS), which form part of Dynamic HTML (DHTML) can be of benefit too for multilingual WEB sites. Not all languages will require identical formatting and Cascaded Style Sheets give a useful way of providing default formats for particular pieces of text that can be overridden for specific languages. For example, it is unlikely that the set of styles that work for Western European languages would work say, for Japanese or Arabic. Using Style Sheets enables the translator to concentrate on the text alone in a way that is partially decoupled from the layout.

Another feature that Dynamic HTML provides is for the client to request fonts from the server. This means that the author of a WEB page can ensure that the fonts that they used are available to the client (subject to copyright). There are two incompatible systems for doing this, one from Netscape/Bitstream called TrueDoc and one from Microsoft/Adobe called OpenType. In both systems the process is the same, a local font is stored in a compressed file for transmission across the network when requested by the client. Part of the format ensures that the font can only be downloaded from servers within specified domains (see Annex A, section 8: Character encoding and fonts technologies). At the time of writing, there is no software that can process double byte characters for the TrueDoc system. OpenType appears to handle double byte characters without a problem.

Although it may be argued that a speaker of a particular language will have at least a minimal set of appropriate fonts installed, it is poor engineering practice to rely on the client configuration for the user to be able to see your WEB site. Also, it gives a poor impression if say, an English speaker viewing a multilingual page sees the Japanese characters represented as missing character symbols.

It is obvious that the 'charset' attribute should be set correctly for a WEB page, and this is particularly important for multilingual WEB sites since relying on a default character set will be particularly unreliable. However, the most popular two WEB browsers (Netscape and Internet Explorer) do not interpret the charset in a way which is consistent with one another, let alone consistent with the standard. For example, named and numbered entities should always be interpreted according to the Unicode character set irrespective of the 'charset' attribute. This is behaviour defined as part of the SGML specification of HTML 4. Both the mainstream browsers allow the 'charset' attribute to influence interpretation of entities. Some of the less popular browsers however do show conformance with the standards.

4.9 Pages with embedded scripts

HTML pages can include scripts either to be executed on the client or the server. There are various scripting languages (JavaScript, VBScript, ASP, PHP etc - see Annex A, sections 4, 5 and 7). Scripts may also need to be translated. Any script needs to be very carefully written with this in mind. General programming guidelines for writing international software apply here. Primarily, this means a clear separation of localisable and non-localisable elements. If the translator is expected to translate HTML files which contain scripts, then any localisable elements in the script must be made explicit. Translators are not normally experts in script-writing.

The best way to achieve this is to place the localisable elements of the scripts into separate resource files and use the batch process to produce the 'live' files.

Sometimes the procedural information needs to be localised. For example, there may be procedures for constructing strings from data, such as say, a string representing a date. This will be locale-dependent. If each resource is just a string, then there is no reason why the string could not represent say, a function in a scripting language.

	Where processed	*What generates*	*Need new browser?*	*May use database*	*Program mable*
HTML	Client	Display	maybe	N	N
XML	Client	HTML, Display	Y	Y	Y
Java	Client	Display	N	N	Y
JavaScript	Client	Display	N	N	Y
CGI	Server	HTML	N	Y	Y
ASP	Server	HTML, Display	N	Y	Y
SSI	Server	HTML	N	N	N
CSS	Client	Display	Y	N	N
PHP	Server	HTML	N	Y	Y

Table 2
Relations among various components of a Web page.

Table 2 shows at a glance the relationship among various components of a web page. The columns show where component is processed (client- or server-side), what generates (HTML, display properties, client-side code), if it needs a modern (>1998) browser, if it may interact with a database, and if it is a programming language.

4.10 Graphics

Any WEB site would be pretty dull without some graphics and sometimes these will need to be localised as well as the text. It may be that the image in the graphic is different, because of differing cultural sensitivities or marketing decisions, or it may be that embedded text has to be localised. Sometimes, both cases will occur, if only because the text is longer in the target language than in the original language.

It is important to keep the text layers separate from the graphics layers so that the text can be translated independently of re-drawing the graphic (if re-drawing is necessary). It is possible that the text and the graphic could be combined in an off-line process such as that advocated here. However, the results are more likely to be satisfactory if text and graphic are combined by hand.

4.11 Use of a Translator's Workbench

For all WEB sites (that are of any interest), it is likely that information will need to be updated from time to time. This process of updating a WEB site is quite different to that of re-issuing a paper document. Most WEB sites undergo frequent small changes almost continuously. This means that the ongoing cost of maintaining a multilingual WEB site can be very high. Considerable savings and improvements in consistency can be achieved by using a translation memory tool. With such tools, each phrase that has been translated is stored. Next time a translation needs to be done, phrases that have already been translated will thus not be re-translated and phrases that are similar to previously translated phrases can be presented to the translator to improve speed and consistency.

Related to translation memory is the issue of terminology management. Specialised words are used in particular domains or even to achieve a particular effect. For example, the top-level page on a commercial WEB site normally has very carefully chosen terminology to project the right image for the company and to be technically accurate. These terms need to be carefully chosen in each language and so will typically need to be translated by someone who is both a native speaker of the target language and has the appropriate technical and marketing skills. They also need to be translated consistently each time they are used. When such terms are chosen, they should be stored in a term bank for use by translators so that the effort of choosing these terms does not need to be duplicated.

4.12 Web Content management tools

Management of a Web site is the part of the process of building a site which lies between the content creation, that is tools and repositories, and the content delivery, performed by web servers and applications. This means that developing, deploying and quality review take part at that phase.

Many systems, either commercial or free, exist to ease the task of content managing for a Web site. Annex A shows (in sections 9 and 10) some of them in detail: the interested reader may refer to that Annex for further information. Among the products that may be used, there are: Vignette V/Series, a complete platform, from project management to site building (see http://www.vignette.com/); Allaire HomeSite and ColdFusion Studio, an integrated Web editor and a comprehensive system to build a site (see http://www.allaire.com/); Macromedia DreamWeaver family (especially the just born UltraDev, which helps the developer create web applications) and Macromedia Generator (see http://www.macromedia.com/); TeamSite from Interwoven, which prefers to concentrate on Content Management rather than to cover all steps of the process (see http://www.interwoven.com/); Frontier, from UserLand, contains a HTTP server, programming, database and XML environment, together with their workgroup application Manila (see http://www.userland.com/); Last, Zope is the leading Open Source web application server: this may be seen as a drawback from some people, but it may give really good results if the web developer is used to such products (see http://www.zope.org/).

As this brief listing of offerings shows, many firms have products which cover more or less the whole spectrum of tools and applications needed to build a site. It's really difficult to say which product is the best choice: the answer depends from the environment where the Web designer works and from his tastes. Unfortunately, however, all of these products are designed to build a monolingual web site, and they do not take into account the fact that the same page will be present in different languages, or that the application should interact in different ways with the same database in order to come out with a localised page.

Metadata standards are also important as Web Content Management Tools, since they ease the task of putting information about the kind of data present in web pages so that other external applications may exploit it. Annex A chapter 12 provides information on some of the existing standards.

4.13 Web site management tools

There exist tools (such as SDL's WEB Flow) that will monitor a site at pre-set intervals to check for files that have changed or been added. These files will automatically be passed to a translation memory for a measure of how much new text needs to be translated. They can even produce an estimate of the cost of translating the new pages. If necessary, the text of these files will be filtered out and passed to a human translator.

4.14 Scripting languages and character encoding.

In the architecture proposed in this section, the batch process will need to recognise strings in a text and carry out some kind of pattern matching. A very popular language for such processing is Perl. Perl claims to handle Unicode through UTF8 encoding and later releases of Perl will handle Unicode more directly if the underlying operating system supports it.

However, to get a Perl script to work on multilingual text, it is advisable to set the 'locale' correctly so that say, date formatting and string processing functions behave correctly.

However, a multilingual WEB site will have different locales for different resources (of course!), therefore it may be necessary to alter the locale dynamically whilst generating the HTML files.

A preferred alternative would be to use UTF8 as the character encoding scheme and restrict all locale-independent information to being written using characters in the ASCII range. In this way almost any scripting language will function correctly without needing to know the locale and any locale-specific information is just treated as data by the script without needing to know the character set.

5 Possible Implementations

The previous section described guidelines for organising a multilingual WEB site. These amount to aspirations which allow a clean separation of the different kinds of information that constitute such a site. In practice, certain compromises may be required when the complexity of a WEB site is such that complete separation presents too much of an overhead compared with the complexity of the WEB site, or when total separation would lead to an inflexible design. In this section, some examples of WEB sites of varying complexity are discussed, along with concrete suggestions as to how these WEB sites may be implemented using existing WEB tools and techniques

5.1 A simple Web site

Perhaps the simplest way to build a WEB site is just to create a few pages in HTML and put them in appropriate directories on a WEB server. Such a WEB site could be localised by having a translator translate the files and then storing them in the appropriate way to allow language negotiation to retrieve the appropriate version of the page. The translator would not have to update links, since language negotiation would retrieve the appropriate pages each time.

It is clear that this approach to WEB site design completely ignores any of the guidelines described above. So, what can be done to improve it?

The first step is to provide an alternative way for the user to select locale. An explicit control on at least the front page of the Web site that allows the user to select language would be useful. When the user operates such a control, the page would be re-loaded. If the user clicks on a link on the newly loaded page, we would like to make an attempt to stay with the locale that they have selected. If we are not using language-negotiation (which we are not, if there is an explicit language control), then the target URL will need to indicate the locale desired. This means that as part of the translation of the page, we would need to alter those URLs whose targets we knew were available in the appropriate locale. Since the URLs in this scenario are locale-specific, this will override language-negotiation completely.

Clearly we would like to avoid having to alter all the target URLs by hand. A better solution would be to have an automated process do this. However we will need to state somewhere which URLs should be altered and which should be left the same (we would need a policy for this even with a human process). We might have a policy such as internal WEB pages are displayed in the appropriate locale, external ones are displayed in the default locale for the target WEB site.

With this approach to structuring a multilingual WEB site, we have kept formatting and content together. Any alterations to the format would need to be replicated for all locales. For very small number of pages this may not be a problem. However, if the number of pages is moderately large, then it will be very inefficient to do this. Even with a monolingual site we might wish to keep formatting separate from content just to be able to store all formatting information in one place.

To achieve this separation we need to maintain content separately from formatting and combine them to produce an HTML page. This will also make the translation process easier as the translator does not need to deal with HTML formatted content.

Combining the content with the format can either be done dynamically when the page is requested, or off-line as a batch process. With the simple site described here, there is little benefit to building pages dynamically.

Clean separation of content from format is very difficult to achieve in the general case. If we are displaying mainly textual pages. Our content may take the form of paragraphs to be displayed. These paragraphs may require some embedded formatting, such as italicising certain words. This means that in practice, there may be some benefit to relaxing the requirement for separation of formatting from content.

5.2 A basic site with embedded scripts

This is similar to the previous example, except that some of the HTML pages contain scripts. It has already been mentioned that it is important to keep the locale-dependent parts of scripts separate from locale independent parts. The locale-independent parts could be left in the HTML files and the locale-dependent parts stored in separate files. Again these would need to be combined either dynamically or off-line. If the script is to run on the client then either is possible. However, if the script is a server side script, then dynamic combination might present some problems since the locale-specific parts of the scripts will need to be inserted before being interpreted by the server.

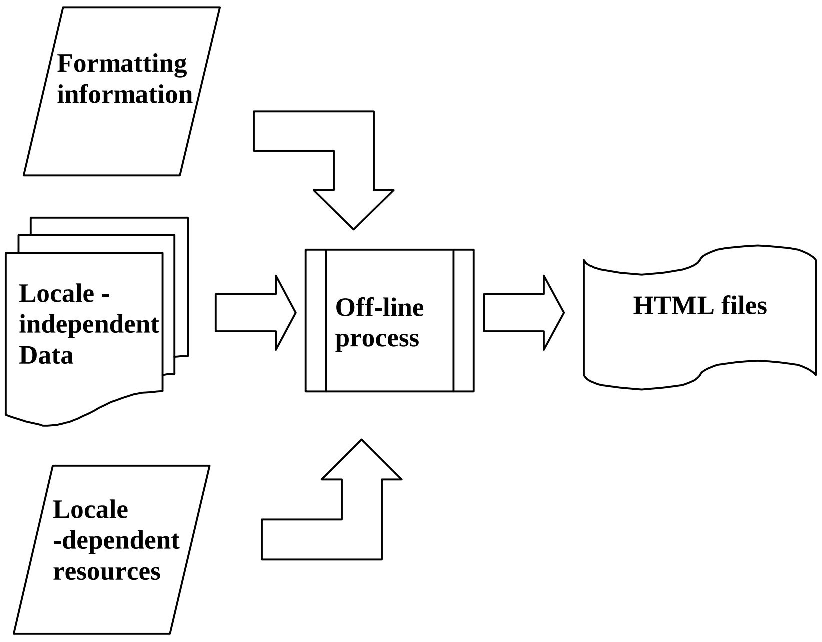

Figure 3
An off-line process for managing a multilingual WEB site

It is straightforward to create a process that would combine the locale-specific scripts with the locale independent parts of the HTML files. This same process could be responsible for combining the content with the HTML templates. This is shown in Figure 3.

5.3 A semi-parallel site

A more complex site might have many more pages and also might have a slightly different content for each locale. This is one of the most difficult situations to manage since we wish to factor out commonalities between different localised versions of a WEB site, whilst still allowing the freedom to have differences in different locales. In practice, it is also one of the most common situations.

Variations between locales might include the presence or absence of particular content from certain pages, the presence or absence of certain pages and variations on formatting from one locale to another.

In dynamic HTML, cascaded style sheets can be used to manage small differences between locales. Each locale could have its own set of style sheets, which are variations on a shared set of style sheets. This way each locale-specific style sheet would only have to specify differences from the generic styles.

Layout differences will require different HTML templates. HTML files are not cascadable, and therefore if there is even a small difference between locales, we would need to have a different HTML file. To construct a particular page in a given locale, we would need to find the HTML template for that locale and combine this with the content for that locale. If we wish to share HTML templates amongst more that one locale, we will need to provide an explicit mapping between HTML template and locale. Again, this combination can be done either dynamically or off-line.

When pages are only present in some locales (which will be typical of any even moderately large site), we need to have a policy of what to do with "missing" pages. It was mentioned above that in such cases it would be useful to have an explicit policy (e.g. remove the link, link to a default locale, or refer to a locale-independent page which in turn refers to a localised version). The policy may be general for the whole site, but is more likely to be link-specific (or at least target-specific). This means that certain targets say, may only be applicable to one locale, in which case we may want to remove the link altogether. For other pages, it may have just been too expensive to localise the page and therefore we should display the page in some other locale version. For links whose targets are external to our site, we may wish to attempt to find a localised version of the page, or we might wish to carry out some processing on the external page, such as machine translation.

In this case, there is a significant advantage to generating pages dynamically. We can determine the status of the target page at runtime and make a decision then as to what to do. For example, we could check for the presence of the target in the appropriate locale.

The various components of this kind of system would be,

Resource files

These contain locale-dependent resources, such as localisable script elements, names for menu items etc.

Content database

This stores the bulk of the content for the WEB site

Link policy file

For each hyperlink, we will need a policy giving information about the target of the link. For example, it could be that the target should be translated by a machine translation system if it does not already exist in the appropriate locale.

HTML templates and Style Sheets

These contain the layout and the details of the format. Each HTML template will refer to a Style Sheet. Each HTML template will correspond to a page for one or more locales.

Locale to template map

If the HTML templates are to be shared across locales, then there will need to be some way to state which template is to be used in which locales. When a client requests a page, we would begin by identifying the appropriate template for that page. This would contain the instructions for building the complete HTML page.

5.4 WEB sites using databases

Many WEB sites, particularly those that support e-commerce, are built around a database. Typically, the database will contain say, product information, which will be used to generate an HTML page dynamically. Normally, HTML pages are not stored directly in the database since this is very inflexible. It is also very inefficient, since the database would be storing repeated layout information for very many pages.

This kind of WEB site then, is similar to the case above, where a WEB page is built from content elements of some kind. There could be some additional issues to be aware of when using databases. If any database procedures are used which are sensitive to character encoding it will be necessary to ensure that the database can use, or be converted to the same character encoding as the rest of the system. For example, Java uses Unicode 2.0 whereas the default encoding for Oracle is Unicode 1.2. In most cases this will make no difference, however it is possible that if say, the 'ORDER' operator is used in an SQL query, then for certain locales the sort order may not be correct.

It is not essential to maintain the same character encoding throughout the system, converters may be used to map between the formats used. It may be convenient to store text in a locale-specific encoding. It will make system design much easier though if Unicode can be used throughout.

6 Tools for multilingual web sites

This section presents an inventory of tools that are required for building, managing and exploring multilingual web sites. It is not aimed to list specific commercial or non-commercial tools, but it presents types of tools that should be involved in localising existing web sites or creating new multilingual web sites and exploring them.

Tools can be classified into two categories:

1. Enterprise/Management tools: those involved in building and maintaining multilingual web sites. This category also includes external localisation agencies.
2. Exploration tools: those involved in using multilingual web sites. These tools are mainly language-related tools (or linguistic tools).

Detailed information about tools together with examples can be found in the deliverable annexes:

– Tools for building and maintaining web sites (see Annex A).
– Linguistic tools (see Annex B).

6.1 Enterprise/Management tools for multilingual web sites

A short list of Enterprise/Management tools is:

– Multilingual authoring tools (HTML/XML editors) for creating and maintaining multilingual web pages.
– Tools for HTML/XML parsing (identification and extraction of web page components: text, graphics, links, etc.) and translatable text quantification. These tools have an important role in preparing the web site localisation.
– Off-line machine translation systems for localisation tasks.
– Specific scripts and procedures for dynamic and personalised interaction with the user depending on his language and culture (CGI, SSI, Dynamic HTML and ASP).
– Tools for security and privacy mechanism (useful for e-commerce).
– Tools for database management.
– External localisation agencies (assimilated to Enterprise/Management tools). External agencies can have an important role in multilingual web sites creation (text drafting, localisation and/or translation, graphics design, etc.).

Other tools are necessary to access to multilingual web sites:

– Search engines with multilingual capabilities (language identification, etc.).
– Multilingual web browsers.
– Tools for Voice input/output with multilingual capabilities.

6.2 Linguistic Tools

This section is not aimed to develop all aspects related to language tools that are useful or suitable for multilingual web sites. There is a lot of literature that contains more complete descriptions of such tools. The reader can also refer to the annexes to this deliverable, especially Annex B, where these tools have been described. In this section a non-exhaustive list of language related tools is introduced, it includes machine translation and machine-aided translation, automatic generation of text from data, text summarisation and cross-language information retrieval. Other language-related tools, such as speech tools, are also suitable for web environment. The reader can refer to Annex C for a detailed description of speech-related tools.

6.2.1 Translation

6.2.1.1 Machine Translation

Of course the greatest additional cost of running a multilingual Web site is translation. It is therefore extremely desirable to be able to automate fully the translation process. There are certainly places where machine translation can be used, but it needs to be used cautiously and with appropriate management of the expectations of the user. In a multilingual Web environment a machine translation system is suitable for two main tasks:

- Web site localisation.
- Cross-language information retrieval. This point is described in section 6.4.

For Web site localisation a machine translation system is generally used in off-line mode to translate textual parts of web pages. Human revision of translation (post-edition) is strongly required, especially for top-level pages of the web site. Translation revision is necessary not only to correct wrong translations but also to check whether the produced texts are suitable for cultural implications of the target language. In addition to translation correctness, one of the major technical problems in machine translation of web pages is related to format preservation of text, with regard to HTML tags and to other components of a web page. To be translated, the text is extracted from the source document and then it is restored, after translation, into the target document, according to formatting codes. One example where format preservation of text is required arises when textual segments change position. Consider the following sentence to be translated from English to French**: "He bought a <B> red </B> car"**. This sentence risks to be translated **"Il a acheté une <B> voiture </B> rouge"** where formatting code does not apply to the right word (rouge/red). In conclusion machine translation systems that are used for web site localisation may consider not only textual parts of web pages but also formatting tags and other significant parts of a web page.

A wide list of machine translation systems can be found in the John Hutchin's MT Compendium: http://www.eamt.org/archive/compendium.pdf

6.2.1.2 Translation Memories

Translation memory is a machine-aided translation and it is widely used nowadays and recognised as a very practical utility, especially for translation experts. Due to the fact that repetition is very common in technical fields and also in frequently updated Web pages or help files, translation memory comes as a very handy and efficient option. A translation memory utility takes a certain source text and it stores it together with its correspondent human translation. Before a new translation starts, the translation tool will scan the text and find exact or fuzzy matches for the new sentence, suggesting previously stored translations to the translator. The translator can usually interact with the system and choose whether to accept them or not, but it is recommended to stick to an existent terminology in order to avoid extensive updating across files.

Some of the most well known products in the market are TRADOS's Translator's Workbench http://www.trados.com/workbench/[1], Déjà vu http://www.atril.com/[2], and IBM's Translation manager http://www.software.ibm.com/ad/translat/[3].

6.2.2 Automatic generation of text from data

The field of natural language generation (NLG) is concerned with the ways computer programs can be made to produce natural language text from computer-internal representations of information[4]. Text generation is an essential part of many natural language applications. For instance, Machine Translation and Summarisation are not possible without some kind of natural language generation. One of the generation techniques consist in using 'templates', string patterns that contain empty slots where other strings must be filled in. This type of generation is used in several applications, for instance in the automatic generation of fairly standard letters. Linguistic notions do not play a crucial role in this simple technique. Some systems can handle agreement and /or conjunction, but not in a theoretically sound way. This way of language generation may be used in limited domains. The advantages are that the technique is very simple and fast (real time operation). NLG is perhaps the NL component for which it is most clear that a large degree of domain and application dependence is inevitable. For some applications seemingly simple template based techniques are fully adequate, whereas other applications might need the full power of linguistically inspired approaches, and probably even more.

The more scientifically oriented approaches to text and language generation are mostly based on some kind of linguistic theory. All approaches that have received some attention in the community appear to distinguish two or three major stages of generation: single-sentence generation (also called "*realisation*" or "*tactical generation*") and multi-sentence generation (also called "*sentence planning*" or "*micro planning*") and content selection (also called "*text planning*" or "*macro planning*"). Sometimes content selection and multi-sentence generation are also collectively referred to as "*strategic generation*".

A survey of the literature and the most relevant web sites suggests that NLG is an active research topic, but that relatively few commercial products are available.

More details about NLG techniques and literature can be found in Annex B (section 6: Language Generation).

6.2.3 Automatic Text Summarisation

Text summarisation is a term that is used to designate a large number of different operations. In a strict interpretation 'summarisation' refers to the process that generates a complete abstract of a potentially lengthy document, in such a way that the most important information in the text is represented in the summary. The text of the summary is generated on the basis of the meaning of the document. Software that is able to produce this type of summary does not yet exist. The process implemented by summarisation products is more accurately characterised as 'abridgement': instead of computing the meaning and argument structure of the document key sentences, key phrases and key words are determined, and concatenated to produce a shorter version of the text that still contains the essential information. Some 'abridgement' products are capable of shortening the key sentences that are taken from the text, for instance by deleting parenthetical remarks.

One essential difference between abstracting and abridging is that the former is virtually impossible without substantial world knowledge, whereas the latter can suffice with clever statistical processing of a document, probably against the background of a thesaurus, or a list of potentially relevant terms [5].

6.2.3.1 Types of Summaries

The uses of Text Summarisation vary with different needs and applications. The amount of compression (ratio of summary length to source length) or the "most relevant content" depends on the intended use. In order to develop consistent procedures to create summaries, responding to these needs and applications, it is necessary to identify and to take into account the "context factors": input (source form, subject type and source unit), purpose (audience and function) and output (material, format and style). A typology of summaries can be found in Annex B (section 8: Text Summarisation)[5,6]. Each type of summary (or the combination of types) has different features, need different methods and techniques to be created and must be evaluated according to different criteria.

Multilingual summarisation consist in developing engines that employ language-neutral methods or simplified language-specific methods to work across languages, and linking summarisation engines to translation engines[7,8].

6.2.3.2 Commercial Systems

Nowadays no experience of integration of text summarisation tools to the web is known as having been successful, especially in a multilingual context. However some commercial systems exist, among them:

– Extractor (NCR / IIT): http://extractor.iit.nrc.ca/[9]
– MS Word AutoSummarize: http://www.slate.com/features/cogitoautosum/cogitoautosum.asp[10]
– ProSum (British Telecom): http://transend.labs.bt.com/prosum/word/index.htlm[11]
– LinguisticX – (Xerox Company): http://www.inxight.com/products/enterprise/summserv.htlm[12]
– ConText (Oracle Corporation): http://oracle.com.ar:1000/products/context[13]
– WebSumm (MITRE): http://www-i.mitre.org/pubs/edge/july_97/first.htlm[14]

Descriptions of these systems can be found in the Annex B (section 8.5: Commercial Systems).

6.2.4 Information retrieval

6.2.4.1 Cross-language information retrieval

Cross-Language Information Retrieval means the retrieval of documents based on explicit queries formulated by a human using natural language when the language in which the documents are expressed is not the same as the language in which the queries are expressed. Users seeking information from a particular information source could benefit from the ability to query large collections once using a single language, even when more than one language is present in the collection. It is the ability to issue a query in one language and receive a document in another that distinguishes cross-language information retrieval from monolingual information retrieval. Monolingual and cross-language retrieval functionality can certainly both be provided by a single system.

Concretely, Cross-Language Information Retrieval is insured through the integration of a machine translation system to a search engine. The machine translation system is used in on-line mode and serves for two different translation tasks:

- Query translation, and/or
- Document translation.

When storage is limited or several languages must be accommodated, translating the query is more practical than translating each document into every language. On the other hand, a strategy based on document translation can permit the translation workload to be performed at indexing time. These alternatives and variations on them, such as mapping both the queries and the documents into language-independent representations, present fundamental tradeoffs that designers of cross-language information retrieval systems must consider.

The most known machine translation system associated with information retrieval engine is *Systran* associated with the *AltaVista* research engine *(http://www.altavista.com)* and with the international *Voila* portal of *France Telecom (http://www.voila.com)*. Other machine translation systems are also available such as *Reverso* (*Softissimo France*) which has been integrated to the local *Voila* portal of *France Telecom (http://www.voila.fr)* and *Logos* with Yahoo search engine (*http://www.yahoo.com*[15]).

6.2.4.2 Information Extraction

One of the typical features of standard search engines is their impermeability to user needs. Statistical techniques of relevance calculus can help a lot in refining and expanding queries, but they are hardly effective in the task of fulfilling more specific desiderata of the user. In a sense, they tend to interpret every query as a request of information about a certain topic, whereas the user might be interested in other (and more specific) forms of interaction, such as buying, selling, renting, downloading, talking, etc. Most Web search engines are based on keyword retrieval of text, however, where the intention is to identify a product or service from within a catalogue, other types of retrieval might be more appropriate, which take advantage of the structure of the product information. For example, when buying a car there are particular fields that can be used to capture the customer requirements, such as price range, model, colour, engine size etc.

It is assumed that a functional dimension has to be added to the indexing and retrieving machinery of a standard web based search engine. It is evident that this dimension can not be reached by using standard information retrieval techniques. This new dimension is called "Information extraction". Indeed, once user's expectations have been identified, information extraction techniques are able to provide much more effective results, as they can analyse small parts of documents just for the purpose of mining the kind of data in which the information seeker might be interested.

The information extraction domain is a new and active research topic. Except for some research prototypes based on reformulation of questions or on machine learning, almost no commercial products are available.

7 Conclusions

The need for multilinguality in Web presence is already evident. Soon the number of non-English speaking customers who are connected to the Internet will outnumber the native English speakers. Moreover, it has been shown that the large majority of Internet users prefer to be addressed in their native language. However, true multilingual Web sites are difficult and expensive to create and maintain, because of the lack of experience with such services. This document provides the starting point for the development of 'best practice guidelines' that will make the creation and maintenance of multilingual Web sites more cost-effective. To that aim we have analysed the roles, the architectures and the tools that are involved in multilingual Web sites.

We have distinguished roles related to the human actors from those related to technology.

Human roles include the Web manager, the Web builder, the end user and some more crucial roles, i.e. the translator and, probably the most important one, the designer.

The designer is the person who creates the structure of the Web site in such a way that it can accommodate the various language versions of the site. A common thread is the need to choose structures and architectures that support the separation of content, formatting (or rendering) and navigation, which is essential for all but the simplest Web sites. This separation is especially important for multilingual sites that bring an additional level of complexity of their own.

A proper separation of content from formatting and navigation also frees the translator from the need to understand the details of the programming in a Web site. It also enables him to reap the maximal profit from language tools like translation memories.

Besides human roles, four major technology-related roles were identified, i.e.. browsers, language tools, content management tools and Web support tools. The discussion was focused on language tools.

In its second part this document provides high-level descriptions of several different architectures for multilingual Web sites, starting with a very simple one and proceeding to potentially very complex applications. Here, the value of separating content, formatting and navigation is again emphasised. With reference to the previously defined (human and technology-oriented) roles it is explained how sites of different complexity are best designed and managed.

This document also provides an introduction to four fairly comprehensive appendices. These appendices deal with Web site technologies (Appendix A), Language tools (Appendix B), Applications of Speech Technology (Appendix C), and possible multilingual architectures (appendix D), respectively. Appendix B discusses language tools, including tools for machine translation, translation memories, text summarisation, text generation, cross-lingual information retrieval and information extraction. The Appendix includes appraisals of the capabilities of generic tools. It appears that fully automatic translation does not provide high quality results. Therefore, it is recommended that the top-level pages of a site are translated by knowledgeable human translators. Pages deeper down in the hierarchy of a site, which are probably requested much less frequently, and probably by users who have a genuine interest in the contents, can be translated automatically to enable the user to grasp the main message. Pages which are updated automatically, like weather conditions or stock data, should be written in a standard language to make the use of automatic translation tools easier.

It was pointed out that translation memories are very attractive tools to make the maintenance of Web pages that are subject to frequent minor updates much more cost-effective. These memories store previous translations. By simply retrieving pre-existing translations the task of the translators is simplified, and at the same time consistency of the translations is enhanced either when a translation is carried out by different translators, or by the same translator on different occasions.

In summary, this document shows that the ‘total cost of ownership’ of multilingual Web sites can be reduced and controlled by a proper design of the site and the proper deployment of a growing range of tools. At the same time, this document provides an expert appreciation of the possibilities, but also of the limitations of the present generation of language tools.

8 References

[1] 'The Trados Workbench' http://www.berlitz.ie/twe/default.htm

[2] 'The DéjàVu WEB site' http://www.atril.com/

[3] 'IBM TranslationManager WEB page' http://www.software.ibm.com/ad/translat/tm/

[4] 'Chapter 4 Language Generation' http://cslu.cse.ogi.edu/HLTsurvey/ch4node2.html

[5] Grishman R., Hobbs J., Hovy E., Sanfilippo A. and Wilks Y. 'Cross-lingual Information Extraction and Automated Text Summarization' pp in 'Multilingual Information Management:Current Levels and Future Abilities. A report Commissioned by the US National Science Foundation and also delivered to the European Commission's Language Engineering Office and the US Defense Advanced Research Projects Agency' eds. Hovy E., http://www.cs.cmu.edu/~ref/mlim/chapter3.html

[6] Jones K. S. 'Automatic summarising: factors and directions' in 'Advances in Automated Text Summarization' eds. Mani I. and Maybury M., MIT Press, (1998)

[7] Erbach G. and Uszkoreit H.'MULINEX - Multilingual Indexing, Navigation and Editing Extensions for the World Wide Web' D 0.19 Final Report (1997)

[8] 'MINDS project WEB site (New Mexico State University)' http://crl.nmsu.edu/research/projects/minds/goals.htlm

[9] 'Extractor WEB site' http://extractor.iit.nrc.ca/

[10] 'MS Autosummarize WEB site' http://www.slate.com/features/cogitoautosum/cogitoautosum.asp

[11] 'ProSumm WEB site' http://transend.labs.bt.com/prosum/word/index.htlm

[12] 'LinguiticX WEB site' http://www.inxight.com/products/enterprise/summserv.htlm

[13] 'Oracle Context WEB site' http://oracle.com.ar:1000/products/context

[14] 'MITRE WEBSumm site' http://www-i.mitre.org/pubs/edge/july_97/first.htlm

[15] 'Yahoo WEB site' http://www.yahoo.com