Restructuring Multilingual Web Sites

Paolo Tonella, Filippo Ricca, Emanuele Pianta and Christian Girardi
ITC-irst
Centro per la Ricerca Scientifica e Tecnologica
38050 Povo (Trento), Italy
{tonella, ricca, pianta, cgirardi}@itc.it

Abstract

Current practice of Web site development does not address explicitly the problems related to multilingual sites. The same information, as well as the same navigation paths, page formatting and organization, are expected to be provided by the site independently from the chosen language. This is typically ensured by adopting personal conventions on the way pages are named and on their location in the file system. Updates are then performed manually and consistency depends on the ability of the programmers not to miss any impact of the change.

In this paper an extension to XHTML, called MLHTML (MultiLingual XHTML), is proposed as the target representation of a restructuring process aimed at producing a maintainable and consistent multilingual Web site. MLHTML centralizes the language dependent variants of a page in a single representation, where shared parts are not duplicated. Existing sites can be migrated to MLHTML by means of the algorithms described in this paper. After classifying the pages according to their language, a page alignment technique is exploited to identify corresponding pages and to eliminate inconsistencies. Transformation into MLHTML can then be achieved automatically.

1 Introduction

Web sites are playing an increasingly important role in today's society and their economic relevance for the companies is continuously growing. Web sites represent not only a new possibility to advertise products, but are often an alternative to the traditional channels for distributing goods and offering services. As a consequence, the quality and reliability of Web sites are crucial issues; yet, typically the current development practice does not address them. The current situation is somewhat similar to the early development of software systems, before the advent of software engineering [8], when quality was totally dependent on individual skills and lucky choices: now better tools and techniques are needed to help develop, maintain and test high quality Web sites.

To be accessible to a larger audience, Web sites are often required to be multilingual, i.e., sites where the information is supplied in more than one language. For such sites the design, the check of the quality and the evolution phase are even more complicated. In a multilingual site every page in a language is expected to have a corresponding page in every other language of the site, and a change in a language has to be propagated to all other languages. In other words, the presented information as well as the site structure should be *consistent* across different languages. Not only the available information should be the same and should be displayed with the same format, but intra and inter-language hyperlinks should comply with the overall site organization, leading to the requested information in the right language.

The current lack of automatic support to multilingual site development typically leads to personal choices on the physical and logical organization of the pages in different languages. In some sites a directory for each language is accessible from the top level and inside each directory the same structure is replicated. In other sites, each page name has a suffix identifying the language (e.g., `-en` for English, `-it` for Italian). Others prefer to translate the name of each page into all supported languages, and the correspondence relies on the translation. Finally, some sites adopt a chaotic mix of all options above, with possibly additional variants (numbering the pages in different languages, etc.). In all these cases, the task of maintaining the different site portions aligned is under the responsibility of the developer and is performed manually.

In this paper we propose techniques and algorithms that can be used to support restructuring of multilingual Web sites so as to align the information provided in different languages and make it consistent across languages. The target of the restructuring activity is an XML representation, called MLHTML, which extends XHTML (the new version of HTML compliant with XML, promoted by the W3C con-

sortium) with one more tag (`ml`) for multilingual support. MLHTML was designed to simplify the evolution of a site while maintaining pages in different languages consistent. Existing Web sites can be migrated to MLHTML according to a process consisting of two main steps:

1. Page alignment.
2. Page merging.

The first operation in this process aims at aligning the pages in the different languages. Page alignment is conducted (semi) automatically: tools can be used to support the identification of the correspondences and of the inconsistencies, but the developer is still required to fix the problems discovered. Then, aligned pages can be merged into a single MLHTML page. The published Web pages can be extracted from the MLHTML pages offline, once for all; they can be generated dynamically by a server script, on demand, or they can be directly interpreted by the browser, if it can process XML. Every maintenance operation can now be performed on the MLHTML representation of the pages, thus ensuring a consistent propagation of the changes to all site versions in different languages. Tools have been developed to support all phases of the process.

It can be noted that the page alignment operation is desirable even if MLHTML is not adopted, since it produces a Web site in which the provided information does not contain differences depending on the chosen language. Moreover, the same restructuring process, with similar tool support, can be adopted for a target representation different from MLHTML (for example, the objective could be migrating to a content management system). The advantages of this restructuring are a simplification of the maintenance phase and a higher quality of the site (all portions are consistent). Of course, a site that is developed from scratch can be directly coded in MLHTML. The experience we made with real world sites, downloaded and analyzed by means of our tools, indicates that a high amount of multilingual Web pages is maintained consistent manually and the availability of a tool-supported restructuring process to migrate them would be very beneficial.

The paper is organized as follows: the next section describes the page alignment procedure. Then, MLHTML is considered and its features are discussed. The restructuring of a real-world multilingual Web site is presented in Section 4, where the usefulness of the proposed techniques is commented. Related works are reviewed in Section 5, while the last section is devoted to conclusions and future work.

2 Page alignment

The purpose of page alignment is to determine the correspondences between pages in different languages belonging to a multilingual site and to resolve inconsistencies (missing pages, tags, links or translations). A first problem is how to partition the pages in the site according to languages. In some cases it is easy to infer this information from the naming convention adopted for the pages, but more generally it may be useful to support this operation with a language identification tool, which analyzes the content of the pages. In this way it is possible to recognize pages that were not translated but respect the convention and pages that do not conform to the naming convention.

After assigning pages to different languages, the correspondences between pages have to be retrieved. For such a purpose the *structure* of the pages is exploited: the syntax tree of the HTML code provides information on the page formatting and on the location of text, images and other objects contained in the page. A structural comparison algorithm will be described to infer the matching between corresponding pages. A relevant contribution to this algorithm comes from the comparison of attributes of the syntax tree nodes, and, among these attributes (node labels, etc.), of the text. A natural language processing method was employed to determine if the text attribute of a node is the translation of another.

2.1 Language identification

The problem of language identification was extensively investigated in the Natural Language Processing (NLP) research community, and some methods and tools have been proposed to address it [4, 5].

We decided to adopt a solution to this problem based on the entries available in a set of mono-lingual dictionaries, associated with the words in the page to be classified. Our tool determines the number of words in the page text that do not have an entry in each dictionary. The dictionary which gives the minimum number of errors (unrecognized words) for the given page is selected. If the ratio between number of errors and number of words is greater than a proper threshold (in our implementation set to 0.5), the language automatically assigned to the page is *unknown*, i.e. the high number of unrecognized words makes the language identification procedure unreliable. This case may occur when a page is internally multilingual, so that no dictionary can give a low number of errors on its words, or when a page contains language independent information (e.g., a list of person names and phone numbers). There is a second situation in which this algorithm cannot classify the page. This is when the same minimum number of errors is produced by more than one dictionary. The output of our tool is an *ambiguous classification* and the operator is then required to solve the ambiguity. In all other cases the algorithm can assign a language to each page in the site, thus partitioning it into mono-lingual portions, without having to rely on the

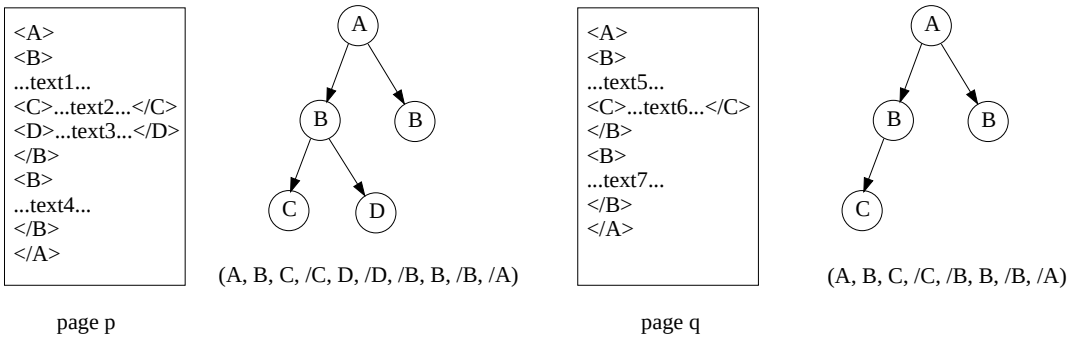

Figure 1. *The structural comparison of the pages* p *and* q *gives an edit distance equal to 2.*

naming conventions.

The dictionaries employed for this work are a subset of those distributed with the UNIX utility `ispell` (more specifically, the Italian, American and British English, Spanish, German and French dictionaries have been used). More dictionaries can be found in the public domain if needed.

2.2 Page matching

The basic property which is exploited to determine the matching of pages in different languages is the page structure, as coded in HTML. Given two HTML pages, *p* and *q*, their structure can be represented by the syntax trees resulting from parsing the two pages. Two examples of such trees are provided in Figure 1, where the pseudo-HTML tags `A, B, C, D` are used. In the syntax tree for a page, information nesting is mapped onto the parent-child relation. Moreover, nodes are assumed to be decorated with a label, equal to the HTML tag, and with a set of attributes[1] (not shown in the figure), as, for example, the *text* attribute, containing the text within the given tag, and the *image* attribute, containing the file name of the included image, if any.

In order to compare the syntax trees associated with two pages, a tree visit is performed, during which a list representation of the tree is generated. When a node is entered during the visit, the associated tag is inserted into the list, while the corresponding closing tag is put into the list when the node is left, after completing the visit of its children (this is done even if in the HTML source code the tag is not closed). The lists associated with pages *p* and *q* are given in Figure 1 below the syntax trees. It can be shown that trees and lists are isomorphic with each other, so that there is no information loss when considering the lists instead of the trees.

Dynamic programming algorithms for the comparison of sequences [2] can be employed to determine the number of edit operations that transform the list representation of a page *p* into the list associated with page *q*. The typical set of allowed edit operations on sequences consists of element deletion, insertion and update. In the example in Figure 1 it is easy to see that the sequence associated with *p* can be transformed into that associated with *q* by deleting two elements, `D` and `/D`. Therefore the two pages have a structural edit distance of 2.

Using the sequence edit distance for page comparison is very appealing because the resulting edit operations have a direct interpretation in terms of elementary actions that are executed by the developer to modify the first page and make its structure correspond to that of the second page. In fact, a Web site developer is expected to delete, insert or update the same tags as obtained from the distance computation in order to perform the transformation. When two pages are considered as matching, because they are at minimum distance, the result of the comparison provides information on the internal features of the two pages, by indicating explicitly the operations that must be performed to update the first page and make it consistent with the second with respect to the structure.

The algorithm sketched above can be extended to incorporate any node attribute, other than the node label, into the comparison procedure. With reference to the *text* attribute, an edit operation will not be necessary to transform a first node into a second one, provided that they have the same label, only if the text associated with the first node is a translation of the text associated with the second node. In the example in Figure 1, the three nodes labelled `B, C` and `B` are considered matched, and thus require no edit operation, only if `text1`, `text2` and `text4` are respectively translations of `text5`, `text6`, and `text7`. If some of these pairs are not translations of each other an update edit operation needs be executed, and the associated cost will increase the final value of the edit distance. The comparison of node attributes can be extended to other attributes (e.g., *image*). The comparison of the *text* attribute, aimed at determining if a text is the translation of another text, is the subject of the next sub-section.

[1] Node attributes should not be confused with attributes inside tags.

```
MATCH(S1: Set<Page>, S2: Set<Page>)
-- Exploits |nodes[p] − nodes[q]| as lower bound for DIST(p, q)
1   for each page p in S1 do
2       m ← min_{q∈S2} |nodes[p] − nodes[q]|
3       S ← {q ∈ S2 • |nodes[p] − nodes[q]| = m}
4       r ← arg min_{q∈S} DIST(p, q)
5       d ← DIST(p, r)
6       if d > m then
7           S ← {q ∈ S2 • m < |nodes[p] − nodes[q]| < d} ∪ {r}
8           r ← arg min_{q∈S} DIST(p, q)
9           d ← DIST(p, r)
10      end if
11      if d < threshold then
12          MATCHED ← MATCHED ∪{(p, r)}
13      end if
14  end for
15  DEL1 ← S1 \ {p ∈ S1 • (p, r) ∈ MATCHED}
16  DEL2 ← S2 \ {r ∈ S2 • (p, r) ∈ MATCHED}
```

Figure 2. *Pseudocode of the algorithm to determine the match between two sets of pages, S_1 and S_2.*

Given two sets of pages, S_1 and S_2, associated with two different languages, it is possible to match each page p from S_1 with a page q from S_2 by choosing the page from S_2 at minimum edit distance from p. Since edit distance determination is computationally expensive ($O(n^2)$, where n is the number of tree nodes), an algorithm was devised to reduce the number of page comparisons to be performed (see Figure 2). This algorithm exploits the difference between the number of nodes in the two sequences associated with the pages as a lower bound for the edit distance. Such a measure is actually a lower bound for the edit distance because all nodes contributing to it have necessarily to be deleted, if belonging to the first page, or inserted, if belonging to the second page. Additional edit operations may be required for the other nodes. An initial subset of S_2, S, is first determined, containing all pages for which the distance lower bound is minimum (the minimum value is assigned to m). With reference to the example in Figure 3, the minimum lower bound is equal to 4, and 3 pages, r, q_1 and q_2, are in S. The edit distance computation (DIST) is then executed on the elements of S, which is expected to be a small subset of S_2. In Figure 3, the distance follows the lower bound (a slash separates them). If the minimum edit distance (assigned to d) between p and the elements of S is equal to m, the algorithm stops, since all nodes in $S_2 \setminus S$ have necessarily a greater edit distance, being the related lower bound greater than $m = d$. If $d > m$ (this is the case for the example in Figure 3), the subset of S_2 inside which the minimum distance is looked for has to be enlarged, so as to include nodes with lower bound greater than m and lower than d. This new set, indicated as S' in Figure 3, contains all pages with lower bound equal to 5, since the minimum distance on S is equal to 6. The minimum distance d is recomputed on this set (augmented with the previous minimum distance page). The resulting value is a global minimum. In fact, each node outside S has a distance lower bound greater than or equal to d, since S contains all elements with a lower bound between m and d, (elements with lower bound equal to m have already been considered) and the new value of d can only be lower than, or equal to, the previous one. Therefore, no node with distance lower than d can be found outside S. With reference to Figure 3, the minimum distance page is r', with a distance equal to 5. Such a distance is lower than 6 (the previous minimum, r) and outside S' distances are surely greater than or equal to 6, since nodes outside S' have lower bound greater than 5.

The two pages at minimum distance are considered matched, and the related pair is added to the relation MATCHED, if such a distance is below a given threshold. All non matched pages are inserted into the sets DEL_1 and DEL_2. Pages having no counterpart in the second language belong to DEL_1, while the pages that are missing in the first language are those in DEL_2.

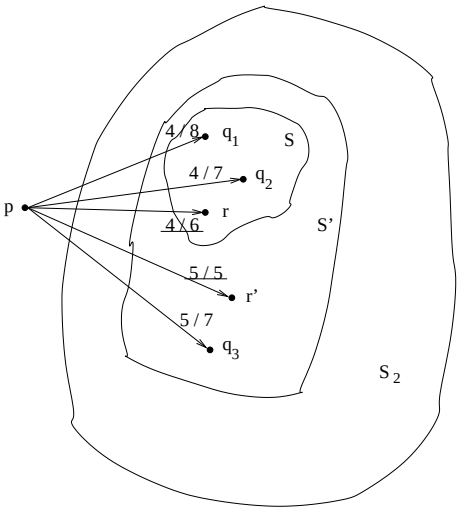

Figure 3. *Lower bound and distance between p and each page in S and S' are shown on the edges, separated by a slash.*

The result of executing this algorithm on a multilingual site is a set of corresponding pages in different languages as well as a set of pages with no counterpart. Moreover, for each pair of matching pages the outcome of the edit distance computation includes the *edit script*, i.e., the set of edit operations that have to be performed to update the pages so as to achieve full alignment of structure and node attributes (including the text). The Web site developer can exploit this information to update the site and make its multilingual portions consistent with each other.

After aligning pages in the multilingual portions of the

site, hyperlinks can be analyzed to identify potential problems. In fact, the evolution of the site could have led to a mismatch of the pages as well as of the hyperlinks. Two categories of hyperlinks should be considered: links internal to each monolingual site portion and links crossing the monolingual boundaries.

If a hyperlink **internal** to a monolingual site portion connects page *p1* to page *p2*, then the two corresponding pages in every other monolingual portion, say *q1* and *q2*, obtained by the alignment procedure described above, should also be connected by a hyperlink. If this is not the case, a warning is issued on the presence of an *internal hyperlink inconsistency* which needs be fixed.

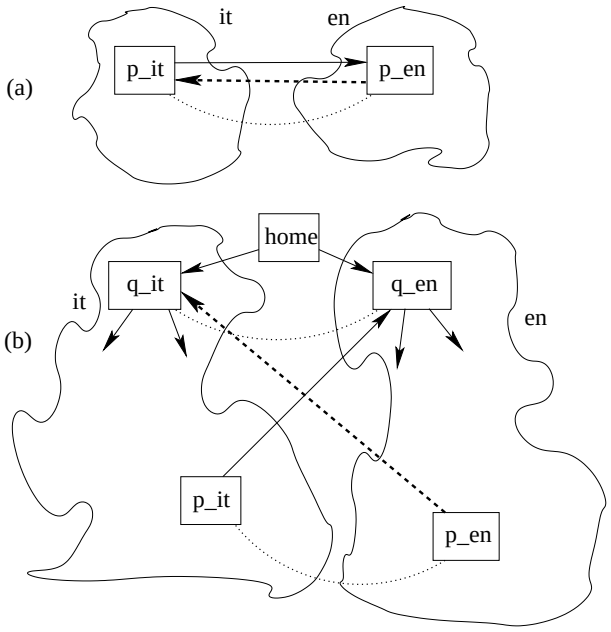

Figure 4. *If a cross-language hyperlink connects two pages, a hyperlink between the corresponding pages must also exist (dashed edges).*

If **cross-language** hyperlink connects a page *p_it* to a page *p_en*, and *p_it* and *p_en* are corresponding nodes according to the page match algorithm (see Figure 4 (a), dotted edge), a symmetric hyperlink from *p_en* to *p_it* is expected to exist. If this is not the case, the developer is warned on the presence of an *external hyperlink inconsistency*, which can be fixed by removing the hyperlink from *p_it* to *p_en* or adding a hyperlink from *p_en* to *p_it* (dashed link in Figure 4 (a)).

A second case occurs when *p_it* and *q_en* are connected by a cross-language hyperlink, but are not translations of each other. For example, the hyperlink may lead from a nested page in a language to the top page in another language (see Figure 4 (b)). In this situation, the two corresponding pages, say *p_en* and *q_it*, should be connected by a hyperlink from the former to the latter (dashed link in Figure 4 (b)). Otherwise, an *external hyperlink inconsistency* is signalled.

2.3 Text comparison

The algorithm for page matching described in the previous sub-section relies on the possibility of deciding whether two texts are one the translation of the other or not. This problem, which we will call the *translation pair problem*, is a natural candidate for the use of NLP techniques. Such techniques can be quite expensive from a computational point of view, and require large repositories of linguistic knowledge, such as lexicons, grammars, knowledge bases, etc. However, in this section we describe a low cost NLP technique able to solve the problem at stake with acceptable confidence, given the aims of the overall application.

Some preliminary evaluations [13] showed that elementary techniques based on the comparison of the number of characters in the two texts [9] are not satisfactory in our case. In fact, even if it can be shown that if T_1 is the translation of T_2 the difference in the number of characters between the two texts is very likely to be lower than, or equal to, 5 percent, the inverse implication is much less likely to be true. Also, the correlation between having a comparable number of characters and being a translation pair is particularly weak when short texts are considered. Note that the texts attached to the nodes under comparison can be very short indeed: for instance a text can be the title of a page or a phrase enclosed between some formatting tags or even a word which has some hyperlink attached to it.

To improve on the character based approach we tried a different algorithm making use of larger linguistic knowledge. The new algorithm is based on the hypothesis that if a text T_1 is a translation of a text T_2, then it is highly probable that any word in T_1 will correspond to a word in T_2 which is its translation. In this case the inverse implication is much more likely than the inverse implication applied to the comparable number of characters [9]. That is, we assume that if all the words of a text T_1 have a translation in T_2 and vice versa, then there is a non negligible probability that T_1 and T_2 form a translation pair, independently from the truth of other conditions.

Unfortunately deciding whether a word of T_1 translates a word of T_2 is not a trivial task, as it requires that the meanings of the two words in the two texts are disambiguated. For this reason we considered a simpler problem, that is finding whether a word of T_1 *can* be a translation of a word of T_2. We assume that the answer to this question is yes if one of the possible lemmas of the word in T_2 is reported as translation equivalent of one of the lemmas of the word in T_1, according to some bilingual linguistic resource, such

as a bilingual dictionary. To check the possible translation correspondences between T_1 and T_2 we also used morphological analyzers (in this work, for Italian and English), able to derive all the possible lemmas for the words of the two texts. Moreover, we treat a number of special cases in which the translation correspondence cannot rely on bilingual dictionaries. For example numbers, dates, and proper names tend to occur unchanged in texts of different languages.

To maximize the significance of the translation correspondence we concentrated on content words, ignoring frequent functional words such as propositions, articles, particles etc. This allowed us to calculate for any candidate translation pairs the *Translation Correspondence Index* (TCI), defined as the ratio between the number of content words that have a possible translation correspondent and the number of content words in the two texts.

In an ideal situation the TCI of any translation pair is 1.0. In reality, very often the TCI calculated by our algorithm is lower. This happens because of limitations in the available resources (morphological analyzers, Collins bilingual dictionary) or because of free or approximate translations. After a preliminary analysis of a corpus of Italian-to-English translation pairs, we arrived at the conclusion that in most translation pairs the TCI is equal or higher than *0.3*. Using this threshold we checked the reliability of the inverse implication, that is, we checked how often a pair that has a TCI equal or higher than 0.3 is a translation pair. The algorithm solves the translation pair problem with a precision (correct pairs over pairs retrieved) of *0.24* and a recall (correct pairs over pairs to be retrieved) of *0.73* on a corpus of translation pairs extracted from a set of Web pages [13]. Although precision is apparently low, the combination of this algorithm with the structural page matching algorithm gives good performances (more detailed comparison data between the alternatives are provided in [13]).

From a qualitative analysis of the cases in which the proposed algorithm fails, it is clear that using more sophisticated techniques and larger linguistic resources would improve the performance of the algorithm. For example the algorithm would gain from the application of sense disambiguation techniques and from the ability to recognize and translate multiword lexical units instead of only single words (*agriturismo* = holiday flats in a farm house). However we think that the technique based on the possible translation equivalents presented here is a good compromise between complexity and efficiency.

3 Multilingual XHTML

MLHTML is an XML representation of multilingual Web pages, with information in all languages inserted into a single source file. The physical proximity of the information is the key to ensure consistency. In fact, document format and structure are shared among languages, and thus cannot be inconsistent. The multilingual content is inserted inside adjacent lines, so that content inconsistencies can be detected easily by visual inspection.

MLHTML adds just one new tag, `<ml lang="`L`">`, to XHTML. The tag includes the mandatory indication of a language identifier, L, assigned to the `lang` attribute, and is matched by the associated closing tag, `</ml>`. The text between opening and closing tags is assumed to be in language L. For example, a multilingual page providing information in English (`en`) and Italian (`it`) may include the following lines, mixed with normal XHTML code:

```
<title>
<ml lang="en"> Publication List </ml>
<ml lang="it"> Lista delle pubblicazioni
</ml>
</title>
```

MLHTML pages can be transformed into HTML code statically (offline) or dynamically (by the server/browser). In the former case, multilingual references to objects (HTML pages, images, etc.) have the form:

```
<ml lang="en">
<a href="ref-page.en.html"> en-text
</a></ml>
<ml lang="it">
<a href="ref-page.it.html"> it-text
</a></ml>
```

while in the latter case the language becomes a parameter to be handled dynamically:

```
<ml lang="en">
<a href="ref-page.ml?lang=en"> en-text
</a></ml>
<ml lang="it">
<a href="ref-page.ml?lang=it"> it-text
</a></ml>
```

Once a multilingual Web page *p.ml* has been coded in MLHTML, HTML pages in any requested language can be produced in three different ways:

offline: HTML pages are produced statically, once for all, and published by the Web server, similarly to any other static HTML page;

by a server script: the request of a multilingual page is recognized (e.g., by the extension) and redirected to a server side script that generates dynamically the HTML page with the content in the requested language;

by the browser: XML enabled browsers can directly process multilingual pages, accompanied by a stylesheet for the selection of the portion in the requested language.

For the offline generation of static HTML in all supported languages, it is possible to use any of the available standard XSL processors (for example, *Saxon*[2]). Their input is the XML document (in our case, *p.ml*), and the stylesheet to use for its processing. We have developed (and made publicly available) the XSL stylesheet *ml.xsl* to be used in conjunction with the MLHTML format. Language specification is obtained by passing the parameter `lang=L` on the command line to the XSL processor.

For the dynamic generation of the HTML pages by the Web server, the PHP script *ml.php* was developed. Every file with extension `ml` has to be preprocessed with *ml.php*. This can be declared to the Web server by modifying some of its configuration files (with Apache, `.htaccess`).

Finally, XML enabled browsers can directly process *p.ml* (possibly renamed *p.xml*), accompanied by the same stylesheet (*ml.xsl*) used for offline HTML generation.

Maintenance of the multilingual pages can now be centralized. Each time a change has to be made, only the MLHTML version of the page (`p.ml`) is modified. Then, the pages in the different languages are re-generated automatically, either offline or dynamically. Consistency of the evolved site is thus granted.

Given an existing site in which different pages were written for the different supported languages, it is an easy task to restructure it into MLHTML, provided that pages in different languages are aligned. A futher requirement is that the input is compliant with XHTML. If this is not the case, automatic conversion tools such as *Tidy*[3] can perform the job of updating the HTML page to the new standard. Then, since the structure of the corresponding pages is the same and the only differences are in the content, provided in different languages, it is possible to automatically merge the multilingual contents into a single page by exploiting the `<ml lang="`*L*`">` tag and to retain the XHTML code for the shared page structure. This work is performed by our tool *PageMerger*. The result of tool execution is an MLHTML page ready for publication on the Web. To improve its readability, pretty printing tools for XML documents (*Tidy* does also this job) can be used. After completing the restructuring process, the old multilingual pages can be disregarded and only MLHTML documents are evolved.

Web pages are often maintained at a higher level than pure HTML code. Graphical tools with nice formatting facilities are available for the creation of Web pages. A new generation of such tools can be devised embedding support for multilingual sites. Their graphical presentation of the page under edit may include support to switch from a language to another, with the effect of changing the content and retaining the formatting. Similarly to current tools, which internally reference the HTML code of the displayed page, multilingual tools could internally reference the associated MLHTML page.

MLHTML pages can also be exploited when a site generates its pages dynamically. In this case, typically, when a page is requested, the server side program that generates it inserts a content in a language depending on some parameter, associated with the selected language. This complicates the code of the server side program, in that every print instruction must include a conditional statement to produce the text in the appropriate language. All such switches can be eliminated if MLHTML is adopted. The server side program can simply generate an MLHTML page including the content in all supported languages. Then, its output is filtered by *ml.php* before being sent to the client browser, in order to allow only the selected language to pass through. This operation is not even required if the browser is assumed to be XML enabled. Such a solution has the great advantage of completely separating page generation from language selection. The current practice mixes these two operations, thus giving rise to complex server programs. MLHTML gives the opportunity to simplify them: language selection is done once for all by *ml.php* or *ml.xsl*.

Restructuring an existing dynamic site is far more complex than restructuring a static site. It is necessary to update the server programs by identifying the code portions the execution of which is language dependent and by replacing them with the production of MLHTML statements. Such a transformation can be partially automated, for example by exploiting pattern matching, but this investigation is out of the scope of the present work.

All tools developed to support migration toward and development in MLHTML (*PageMerger*, *ml.php* and *ml.xsl*) are available from the Web site:
`http://star.itc.it/`.

4 Case study

The tools developed to support the restructuring process described before have been applied to the Web site `www.ien.it`, the official site of the National Electrotechnical Institute (IEN) “Galileo Ferraris”. This Institute performs important duties in the metrological field regarding time and frequency, electromagnetic, photometric and acoustic quantities. Specifically, IEN carries out studies and researches oriented to the realization of the primary standards of the IS measurement units and to the determination of fundamental physical constants. Among the other services, IEN holds the clock giving the official Italian Stan-

[2] http://saxon.sourceforge.net/
[3] http://tidy.sourceforge.net/

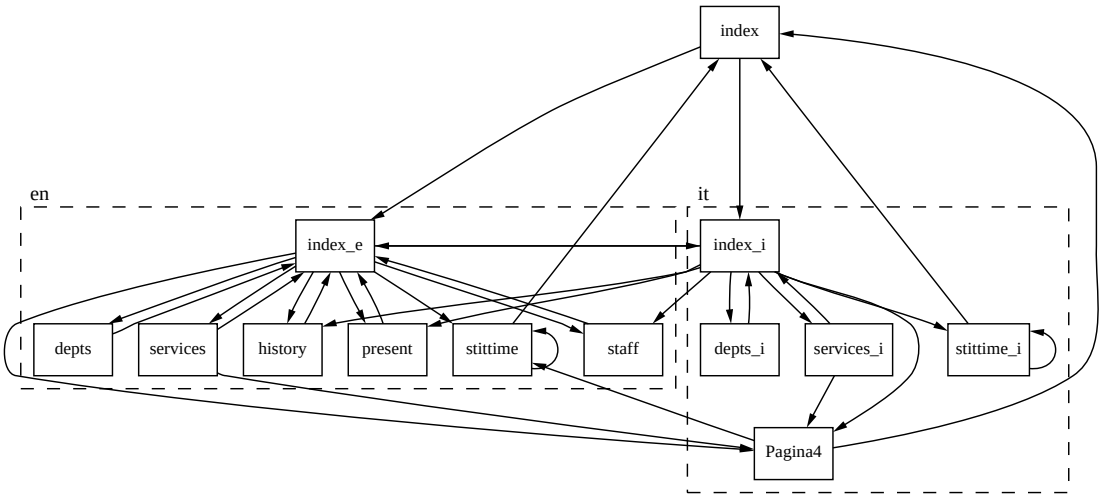

Figure 5. *Graphical view of a portion of the bilingual site* `www.ien.it` *computed by* **ReWeb**.

dard Time, which is also accessible from the Web site.

Original		Portion restructured	
Pages	857	Pages	440
HTML LOC	165887	HTML LOC	51674

Table 1. *Features of the IEN site.*

The considered site was downloaded by means of the Spider module of the Web site analysis tool **ReWeb** [10, 11]. The static portion of the site includes 857 HTML pages for a total of about 165 kLOC (thousands Lines Of Code). It is a bilingual site, providing the same information in Italian and English. The initial page contains two links to the Italian and English version respectively, and (almost) the same page organization is replicated within the two sub-sites.

Figure 5 shows a portion of the site, with pages split between English and Italian (the initial page, `index.html`, is bilingual). The two initial pages of the two subsites reachable from the initial page, `index_i.html` and `index_e.html` are connected by a bidirectional cross-language hyperlink. A visual inspection of this site portion immediately reveals some inconsistencies. Page `Pagina4.html` is available only in Italian, but it is referenced also by English pages. Simmetrically, pages `history.html`, `present.html` and `staff.html` are only in English, but are referenced also by Italian pages. A language-related inconsistency visible in Figure 5 is particularly awkward: if `Pagina4.html` – an Italian-only page – is reached from an Italian page (for example `services_i`) and then the standard time is accessed, the English version of the related page is showed (`stittime.shtml`), although the Italian version does exist (`stittime_i.shtml`). The presence of several inconsistencies even in this small site portion suggests that restructuring is highly desirable.

The HTML parser distributed with the *WebSPHINX* library [7] was employed to obtain the syntax tree associated to each page of the site. Syntax trees are decorated with attributes, such as *text* and *image*, which are used by subsequent analysis phases. The text attribute is used to recognize the page language, while the tree structure, combined with node attributes, is the basis of the page matching algorithm.

English	Italian	Unknown	Ambig.	Errors
268	167	5	0	10 (2.2%)

Table 2. *Page classification according to the language. Last column gives the number of classification errors.*

The restructuring process aimed at migrating the site to MLHTML was applied to a site portion consisting of 440 pages (51 kLOC). Personal directories of the Institute's staff and directories with no page translated were excluded from restructuring. The first step in the page alignment procedure was language identification. The tool for language identification produced the classification summarized in Table 2. Only 2.2% of the language labels produced by the tool are incorrect. An example is page `staff.html`, which contains a long list of names and only a few English words. The number of errors from both dictionaries is high and the page is classified as *Unknown* instead of *English*. Other incorrect classifications are associated with pages which are internally bilingual. The score produced by the tool is lower

than 0.5, so that the language label is not *Unknown*, but it is usually close to 0.5 (e.g., 0.49 for `disclaimer.html`). Consequently, these errors are easy to detect since the score produced by the tool hints a possible classification error (i.e., it is an indicator of confidence in the classification). The overall automatic support to language identification was considered good and manual interventions were very limited (a few minutes in total).

Matched pairs	Errors	Not transl.	Biling.	Errors
122	3 (2.4%)	193	3	0

Table 3. *Pairs of aligned pages and unaligned pages.*

The next step of the restructuring process was page alignment. Our tool was able to match 122 Italian pages with the respective translations in English, with 2.4% of alignment errors (see Table 3). 193 English-only or Italian-only pages were automatically detected by the tool, with no error. For example, page `present.html`, giving a presentation of the Institute, is in English even if accessed from the Italian sub-site. Bilingual pages are recognized as such during language identification. The page matching procedure marks them as unmatched, similarly to those not translated.

Totals			
Edits	Tags	Transl	Errors
622	212	410	348
Average values per page			
Edits	Tags	Transl	Errors
5.0	1.7	3.3	2.8

Table 4. *Edit operations for the alignment of the pages, classified into tag updates and missing translations. Last column refers to unrecognized translations.*

Aligned pages with edit distance greater than 0 were manually updated so as to become consistent with each other. The total number of edit operations in the edit scripts automatically produced by our tool is provided in Table 4. Operations are classified as tag updates (tags or tag attributes are modified, reordered, inserted or deleted to become consistent), and translations (the tool signals that a text pair is not a translation pair because of a TCI lower than 0.3). Errors in the specification of edit operations are always related to the *Transl* category. They are associated with translated text pairs that were not recognized by the tool, due to the presence of free translations, technical terms, abbreviations or contractions (e.g., “CNR” vs. “CNR”). As apparent from Table 4, such cases are quite frequent: 348 over 410. The time necessary to manually update a page and make it consistent with its translation strongly depends on the edit distance from its translation. Typically a few minutes are sufficient to complete the update of a page with about 5 edit operations to be performed, but this may increase up to half or one hour if the number of edit operations is higher and/or they require the translation of long texts that were not originally translated.

After aligning all pages, the *PageMerger* tool generated the MLHTML pages and the original site was (locally) replaced with the new one in MLHTML. Published pages are automatically generated by the server script *ml.php* on demand. A few MLHTML pages required some minor edit interventions, related to the cross-language hyperlinks. In the following code fragment, the language selection had to be changed, since the referenced page is in a language different from the current one. The following line was therefore modified manually:

```
<ml lang="it">
<a href="/index.ml?lang=en">
English version </a> </ml>
```

The resulting MLHTML site was navigated in parallel with the original one, providing the same page organization and information (when initially available), with the noticeable remark that the new site does not contain differences in presentation or content depending on the language chosen. Maintenance of the new site is expected to be simpler, since the number of pages is halved, structural consistency during evolution is automatically granted and texts in different languages are adjacent, thus enabling simultaneous updates.

5 Related work

To the authors knowledge, the problem of restructuring an existing Web site to ensure consistency among its multilingual portions was not considered in the literature. Works on Web site restructuring focused on model-based reengineering [1] and on migration to dynamic content [3].

An experience of web site re-engineering is described in [1], where the target representation of the reverse engineering phase is based on RMM [6]. The recovering process and the following re-design activity were conducted almost completely manually.

The restructuring process described in [3] aims at removing the content from the HTML code and at storing it in a database. When a page is requested, its content is generated dynamically by accessing the database and restoring the requested information. The main advantages of this intervention are a higher maintainability of the resulting dynamic

site and the possibility to detect duplications of content, that are eliminated in the database. The basic techniques exploited in this work are similar to ours, in that the syntactic structure of the HTML pages is analyzed to extract the information of interest. Moreover, a text comparison utility is used in both cases to increase the consistency of the information provided. The main difference is that we consider cross-language consistency, while in [3] duplication removal and database storage ensure a higher level of intra-language consistency and accuracy. Consequently, our text comparison method is not just based on string equality, but has to revert to dynamic programming and NLP techniques.

Only a few works have insofar considered the problems related to the evolution and improvement of the quality of Web applications and to the construction of associated support tools [10, 11, 14]. One of the first studies on evolution of Web sites is [14], where the authors indicate the importance of the maintenance phase. Some experiences made in migrating from an English-only Web site to a bilingual Web site are described in [12].

In a preliminary work [13], we considered the problem of retrieving traceability links between multilingual portions of a Web site. Different techniques and algorithms were contrasted to solve the traceability recovery problem. The best performing one is used as a basic block of the approach described in this paper, where the main objective is restructuring to ensure multilingual consistency and to simplify future evolution of the site.

6 Conclusions and future work

The restructuring process described in this paper was successfully applied to a real world case study. The practice followed during the development and maintenance of the considered site led to several problems. Some pages were provided only in one language, portions of the content inside some pages were not translated, modifications of the page structure and formatting were not always propagated to the same page in the other language, internal and cross-language hyperlinks were not always consistent.

After restructuring, the MLHTML pages were fully consistent with respect to all the issues listed above. Every content fragment was translated (this is directly visible in the MLHTML page text, where adjacent `ml` tags are used for the different languages). Sharing the HTML tags ensures that the displayed page structure will be the same in every language.

The tools developed to support the restructuring process were extremely useful. The whole job was completed in only a few days, thanks to the level of automation achieved, but an estimate of a completely manual intervention is an order of magnitude higher.

Future work will be devoted to trying to apply the same approach to dynamic sites, for which code analysis techniques, such as program slicing, could help identify the language dependent portions of the script code that generate the page content. The target representation will still be ML-HTML.

References

[1] G. Antoniol, G. Canfora, G. Casazza, and A. D. Lucia. Web site reengineering using rmm. In *Proc. of the International Workshop on Web Site Evolution*, pages 9–16, Zurich, Switzerland, March 2000.

[2] M. J. Atallah (editor). *Algorithms and Theory of Computation Handbook*. CRC Press, Boca Raton, Florida, USA, 1999.

[3] C. Boldyreff and R. Kewish. Reverse engineering to achieve maintainable www sites. In *Proc. of the 8th Working Conference on Reverse Engineering*, Stuttgart, Germany, October 2001.

[4] W. B. Cavnar and J. M. Trenkle. N-gram-based text categorization. In *Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, CA, April 1994.

[5] G. Grefenstette. Comparing two language identification schemes. In *Proceedings of JADT 1995, 3rd International Conference on Statistical Analysis of Textual Data*, Rome, Italy, December 1995.

[6] T. Isakowitz, A. Kamis, and M. Koufar. Extending rmm: Russian dolls and hypertext. In *Proc. of HICSS-30*, 1997.

[7] R. C. Miller and K. Bharat. Sphinx: A framework for creating personal, site-specific web-crawlers. In *Proc. of WWW7, Brisbane Australia*, April 1998.

[8] R. S. Pressman. What a tangled web we weave. *IEEE Software*, 17(1):18–21, 2000.

[9] P. Resnik. Mining the web for bilingual text. In *37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, College Park, Maryland, June 1999.

[10] F. Ricca and P. Tonella. Web site analysis: Structure and evolution. In *Proceedings of the International Conference on Software Maintenance*, pages 76–86, San Jose, California, USA, 2000.

[11] F. Ricca and P. Tonella. Analysis and testing of web applications. In *Proc. of ICSE 2001, International Conference on Software Engineering, Toronto, Ontario, Canada, May 12-19*, pages 25–34, 2001.

[12] S. Tilley and S. Huang. Experiences migrating to a bilingual web site. In *Proc. of the International Workshop on Web Site Evolution*, pages 47–50, Zurich, Switzerland, 2000.

[13] P. Tonella, F. Ricca, E. Pianta, and C. Girardi. Recovering traceability links in multilingual web sites. In *Proc. of WSE 2001, International Workshop on Web Site Evolution*, pages 14–21, Florence, Italy, November 2001.

[14] P. Warren, C. Boldyreff, and M. Munro. The evolution of websites. In *Proc. of the International Workshop on Program Comprehension*, pages 178–185, Pittsburgh, PA, USA, May 1999.