Using Clustering to Support the Migration from Static to Dynamic Web Pages

Filippo Ricca and Paolo Tonella
ITC-irst
38050 Povo (Trento), Italy
{ricca, tonella}@itc.it

Abstract

Web sites of the first generation consist typically of a set of purely static Web pages. Content and presentation are not separated, and a same page structure is replicated every time a similar organization of the information is devised. Such a practice poses several problems to the evolution of these sites. It is not easy to update the content, and each time the HTML structure is modified, the same changes have to be propagated to all replications.

In this paper, an approach is proposed for the identification of the Web pages that are more amenable to be migrated into a dynamic version, in that they share a similar structure, filled in with a content organized according to a common scheme. Clustering is used for this purpose: a common template is extracted from the pages in the same cluster, and the variable information of the pages matching the template is migrated to a database. A server side program extracts the requested information from the data base, and generates dynamically the HTML pages to be displayed in the browser.

1 Introduction

In the last few years, Web sites have become important assets for several companies, offering new possibilities to advertise products, and to provide information and e-commerce services. The study of their production, evolution, and quality control has correspondingly become a separate thread in software engineering research, given their peculiarities over traditional software.

The first generation of Web sites consisted typically of a collection of static Web pages, containing advertising material, information about the company, and product-cards. Such Web sites have been progressively replaced by more dynamic sites, with functionalities devoted to accessing data interactively, performing transactions on-line, and exchanging information. Among the others, one of the advantages of dynamic Web sites is that content (data) and presentation can be separated. This has remarkably positive effects on the Web site maintainability and overall quality. In fact, if data are stored separately (e.g., in a database), and Web pages are created dynamically, based on a fixed template, the update of information can be directly performed on the separated data storage. Programs for the dynamic generation of the pages may also need to be modified, but still with the advantage that formatting is not intermixed with content. Moreover, using a single dynamic program to generate a whole set of pages ensures a consistent structure and formatting of the result.

For multilingual Web sites, the advantages of the dynamic approach are even more evident, since these sites have the additional problem of the consistency of structure and content *across* languages [12]. The dynamic generation of the pages in all languages from a common template, as well as the separation of the multilingual content from the page structure, alleviate the consistency problem and simplify Web site evolution.

Several existing Web sites have not yet moved to the next generation of dynamic sites. There are several examples of sites in which the products commercialized by a company are presented in static Web pages, where a given HTML structure is replicated for each product-card, with product-specific values in each page.

In this paper we propose a (semi) automatic process aimed at identifying static Web pages that can be transformed into dynamic ones, with the variable parts (e.g., product descriptions in product cards) replaced by dynamically generated ones. The proposed approach exploits clustering to recognize a common structure of Web pages. This is particularly useful when the physical organization of the pages does not hint any natural way to identify them. The result of the clustering phase provides the candidate template, to be used in the dynamic generation of the pages. A comparison between original pages and template provides the records to be inserted into the database. Then, a script generates the migrated pages dynamically, from the template and the database. Manual intervention is limited to the refinement of the constructed template and database.

Several works on Web application understanding and reengineering have been presented in the literature [2, 4, 6, 7, 11, 12]. The work most related to ours is [4]. In this work, the authors describe a system to extract some items, such as the textual content or the images, from the Web pages of an existing site and to store them into a database. The original Web pages are recreated by retrieveing the migrated information from the database. Replications of a same information are detected and removed. The main difference with our work is that the authors of [4] do not attempt to recognize a repeated structure in a subset of all available pages, and all text fragments are treated the same way: they become a record, stored in a flat database table, with (conceptually) only one column (record field), containing the textual content. On the contrary, we are able to treat different pieces of text in different ways. Some text portions are not moved to the database at all, since they are recognized as a part of the common template (for example, field labels in different product cards). Other text portions are recognized as being values of specific fields, which become entries of proper table columns. Instead of generating a record for each textual content, with only one content field, we generate a record for each migrated page, with record fields associated with the various fields that have been identified (for example, a record for a product card will have a field for the product name, a field for the product price, etc.). Only the pages that match the identified structure are migrated to the database (in [4] all pages are migrated).

In a previous work [11], we have investigated the application of rewrite rules to Web applications, with the aim of restructuring them. In another previous work [12], we considered the problem of retrieving traceability links between multilingual portions of a Web site. The main objective was restructuring to ensure multilingual consistency and to simplify future evolution of the site.

Clustering has several applications in software analysis and reverse engineering [8], and has been recently applied to Web applications [6] with the aim of supporting program understanding. Cloned Web pages are identified in [7] by exploiting an edit distance measure similar to ours, but used for different purposes.

The paper is organized as follows: after giving some general notions on clustering, the next section explains how clustering has been used to recognize Web pages with similar structure. In particular, this section presents the clustering algorithm adopted and the similarity measure used. Section 3 describes the restructuring process. The process is first presented in the simpler case of a mono-lingual Web site, and then it is extended to the general case of a multilingual site. Experimental results are provided in Section 4, where the feasibility and usefulness of the proposed process are discussed. The last section is devoted to conclusions and future work.

2 Clustering

Clustering is a general technique aimed at gathering the entities that compose a system into cohesive groups (*clusters*). Given a system consisting of entities which are characterized by a vector of properties and are connected by mutual relationships, there are two main approaches to clustering [1]: the sibling link and the direct link approach. In the *sibling link* approach, entities are grouped together when they possess similar properties, while in the *direct link* approach they are grouped together when the mutual relationships form a highly interconnected sub-graph.

We investigate the usage of clustering with the purpose of recognizing cohesive groups of static Web pages that can be migrated into a database. In fact, if a set of Web pages comprises pages that share a common structure (*template*) and differ only for the specific values of the information displayed, it is possible to generate them dynamically through a server side script that fills-in a fixed template with the records extracted from a database.

We expect that the static Web pages to be restructured into a database be similar (but not necessarily identical) from a structural point of view. The typical example is a set of product-cards, where only product-specific information varies. Clustering can be applied to recognize the common elements in Web pages that are potential candidates for restrcturing. Since such pages share a common structure, a natural choice of clustering technique is the sibling link approach, based on a structural measure of similarity between Web pages.

When applying clustering in the sibling link approach, two fundamental questions need be answered [1]:

- When are two entities similar?
- What algorithm determines the clusters from the similarity measures?

Given the nature of our problem, the *structural edit distance* [3] among the Web pages is a good choice to measure the dissimilarity of pages (its complement gives the required similarity measure). For example, product-cards have typically a low structural edit distance (high similarity).

In the literature there exist several different clustering algorithms [13], with different properties. Hierarchical algorithms do not produce a single partition of the system. Their output is rather a tree, with the root consisting of one cluster enclosing all entities, and the leaves consisting of singleton clusters. At each intermediate level, a partition of the system is available, with the number of clusters increasing while moving downward in the tree. Divisive algorithms start from the whole system at the tree root, and then divide it into smaller clusters, attached as tree children. Alternatively, agglomerative algorithms start from singleton

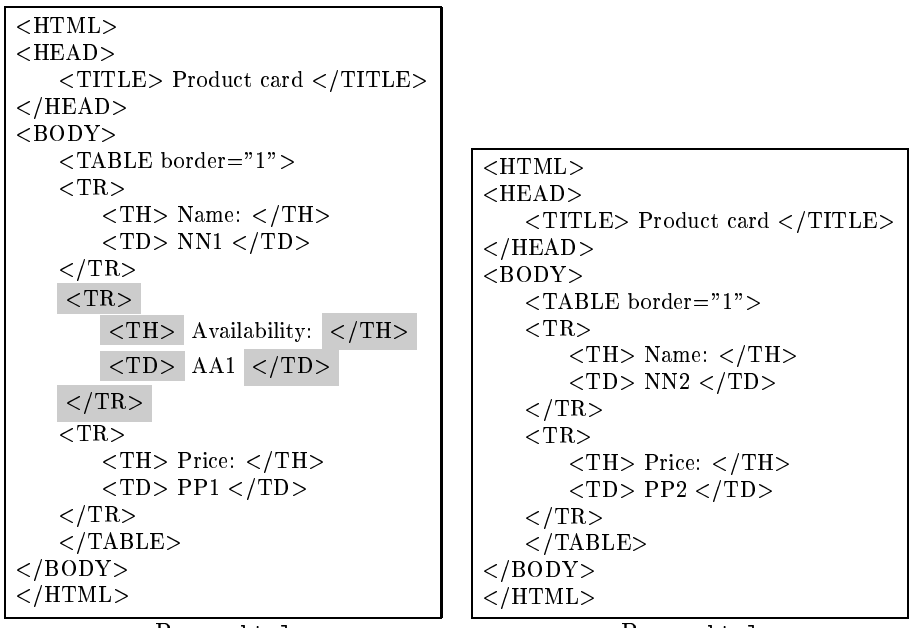

Figure 1. *The structural comparison of the pages* `p.html` *and* `q.html` *gives an edit distance equal to 6.*

clusters and join them together incrementally. Optimizing algorithms [9, 5] produce a single partition of the system, obtained by some heuristics that aims at maximizing a clustering quality (e.g., modularity) measure. We have preferred hierarchical algorithms over optimizing algorithms because they support a manual refinement of the clustering granularity. After (automatically) selecting a clustering tree level (see Section 3 for the method used in the present work), the user can (manually) move upward or downward in the tree, if the clusters at the selected level are too specific (some pages to be migrated are missing) or too general (migration candidates do not share a recognizable template).

2.1 Structural edit distance

Given two HTML pages, `p.html` and `q.html`, their structure can be represented by the syntax trees resulting from parsing the two pages. 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).

Dynamic programming algorithms for the comparison of sequences [3] can be employed to determine the number of edit operations that transform the list representation of a page `p.html` into the list associated with page `q.html`. The set of allowed edit operations on sequences consists of element deletion, insertion and update (the latter weighted 2, since it is equivalent to a deletion followed by an insertion). In the example in Figure 1, it is easy to see that the sequence associated with `p.html` can be transformed into that associated with `q.html` by deleting the six elements with gray background: `TR`, `TH`, `/TH`, `TD`, `/TD`, and `/TR`. Therefore the two pages have a structural edit distance of 6.

In the example in Figure 1, the common template for product description can be obtained either by deleting the six highlighted tags from `p.html`, or inserting six new ones, with related information, in `q.html`. In both cases, six elementary edit operations are necessary to make the given page aligned with the chosen template.

The computational complexity of the sequence edit distance is $O(N^2)$, where N is the sequence length (proportional to the page size). In our experience (see Section 4), this is acceptable, since typically the size of a static Web page is quite small, so that the comparison of each pair of pages can be achieved in a short time (order of seconds for the sites we have analyzed).

2.2 Agglomerative hierarchical clustering

The agglomerative hierarchical clustering algorithm builds a hierarchy of clusterings starting from the bottom of the hierarchy, where each entity is in a different cluster. In each following step, the two most similar clusters are joined. After $N - 1$ steps (with N the number of enti-

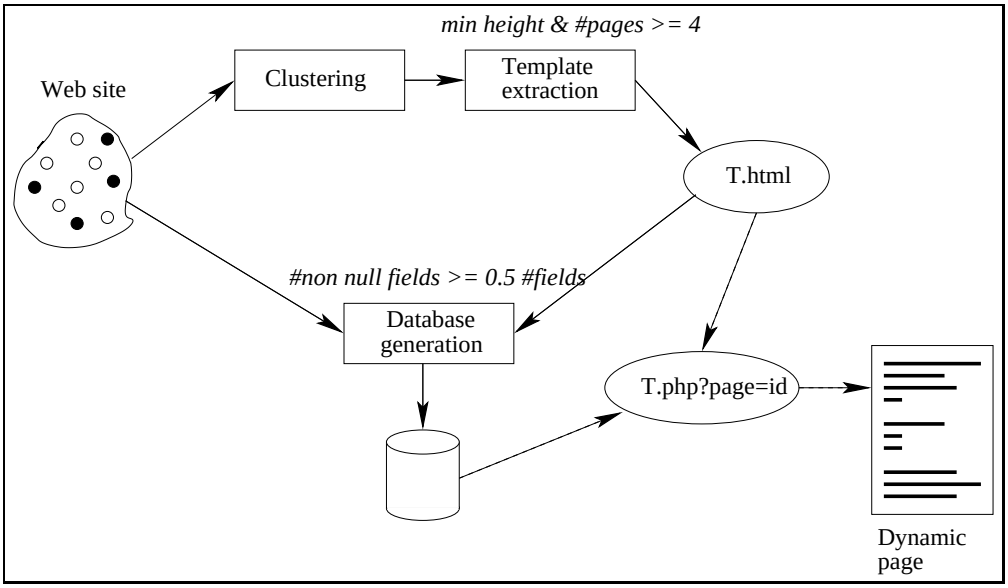

Figure 2. *The restructuring process.*

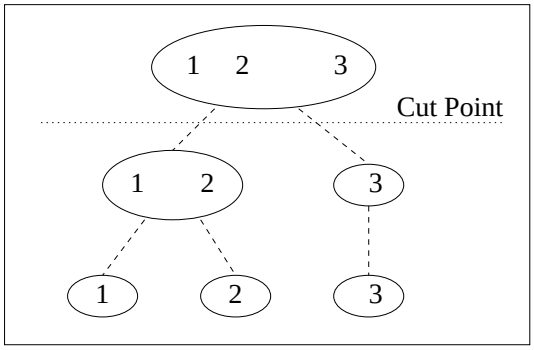

Figure 3. *Hierarchy of clusterings for three entities. The cut point determines 2 clusters,* $C1 = \{1, 2\}$ *and* $C2 = \{3\}$

ties), all entities are grouped into one cluster. Each level in the hierarchy defines a partition of clusters (i.e., a clustering). To select the resulting clustering, a *cut point* has to be determined (see Figure 3).

We have adapted the agglomerative hierarchical algorithm known as Johnson's algorithm [13] to our purposes:

1. begin with N clusters, each containing one Web page (N is the number of Web pages in the Web site), and compute the structural edit distance between pages.
2. **while** there are more than 1 clusters **do**
   (a) find the pair of clusters at least distance;
   (b) merge these clusters into a new cluster;
   (c) update the distance measures between each pair of clusters.

   end while

To update the distance measure between the clusters, we have chosen the so called *complete linkage rule* [1]. This rule states that the distance measure between the already existing cluster C and a new cluster D, formed by joining clusters A and B, is the minimum between *dist(*C, A*)* and *dist(*C, B*)*. It privileges cohesion over coupling [1].

3 The restructuring process

The restructuring process (see Figure 2) starts with *Clustering*. An HTML parser produces the abstract syntax trees of the Web site pages. Then, for each possible couple of syntax trees the edit distance is determined. At this point it is possible to apply the clustering algorithm, as described in the previous section. The result is a hierarchy of possible clusterings.

The next step is the selection of a cluster from which a template page with all fixed information and common HTML tags can be extracted. The algorithm that has been defined to automate such a choice takes into consideration cut points at increasing height (at height 0 all pages are in singleton clusters, at the top all pages are in one cluster). When a cluster is encountered which contains a number of pages greater than or equal to a given threshold (set to 4 for the experiments in Section 4), such a cluster is selected for the computation of the template.

```
<HTML>
<HEAD>
    <TITLE> Product card </TITLE>
</HEAD>
<BODY>
    <TABLE border="1">
    <TR>
        <TH> Name: </TH>
        <TD> <DB table="table1" field="name"/ > </TD>
    </TR>
    <TR>
        <TH> Price: </TH>
        <TD> <DB table="table1" field="price"/ > </TD>
    </TR>
    </TABLE>
</BODY>
</HTML>
```

Figure 4. *Example of template obtained as the LCS from the pages in Figure 1.*

PageId	product card	name	availability	price
p.html	Name:	NN1	AA1	PP1
p'.html	Name:	NN1'	AA1'	PP1'
q.html	Name:	NN2	null	PP2
q'.html	Name:	NN2'	null	PP2'
q".html	Name:	NN2"	null	PP2"
r.html	Name:	NN3	null	null

PageId	name	price
p.html	NN1	PP1
p'.html	NN1'	PP1'
q.html	NN2	PP2
q'.html	NN2'	PP2'
q".html	NN2"	PP2"

Figure 5. *An example of database generation. Before (top) and after (bottom) disregarding some records and fields.*

In the *Template extraction* phase, the Longest Common Subsequence (LCS) algorithm [3] is applied to the set of pages in the selected cluster, resulting in the sequence of tags and texts that are common to all pages, i.e., the template of the dynamic pages to be generated (`T.html` in Figure 2). Note that while for *Clustering* the text between tags is ignored, computation of the LCS involves the comparison of the text between tags in addition to the tags. When a difference is detected, the template includes a placeholder for a value retrieved from the database, consisting of a new tag `DB` with the attributes `table` and `field` for the identification of the database entry to retrieve. A page identifier is used to select a given record from the database table. Figure 4 shows the LCS produced by comparing the pages `p.html` and `q.html` of Figure 1.

The next phase is *Database generation*. During this phase, not only the pages in the selected cluster are migrated to the database: all Web site pages are re-examined, and each time the number of non null field values that can be determined for a page is greater than or equal to half of the total number of fields, the static page is converted into a dynamic one, and the variable information it contains becomes a new record inserted into the database. For each page, a page identifier is used as primary key of the related database record. Record fields (table columns) are determined in the following way. They are initially approximated as the set of fixed text strings appearing in the template. However, not all of them are actually fields. Those that are not followed by a variable text (the field value) can be disregarded. Moreover, all fixed text strings for which only a minority of the records (less than 50%) have a non null value (among the records that satisfy the criterion in Figure 2 for *Database generation*) are disregarded as well. For each record, the values of the fields to be inserted into the database are determined as the variable text in each page that appear after the fixed text recognized as a field name.

Let us suppose to analyze a Web site with the following six pages: `p.html` and `q.html` as in Figure 1, `p'.html` equal to `p.html` but with different values (NN1', AA1', PP1' instead of NN1, AA1, PP1). `q'.html` and `q''.html` equal to `q.html` but with different values (respectively NN2', PP2' and NN2", PP2" instead of NN2, PP2). Finally `r.html` with a structure very similar to `q.html`, but without "Price:" and "PP2" inside the `<TR>` tag. If the cluster chosen for *Template extraction* consists of the pages `p.html` and `p'.html`, the generated template `T.html` is similar to the template of Figure 4 with an extra `<TR>` row (labeled "availability"). Given this template, the list of potential fields is composed of four elements: "product card", "name", "availability", and "price". During the phase of *Database generation*, some potential fields (columns) and a potential record (row) of the initial version of the database (Figure 5, top) are disregarded. Precisely column 1 ("product card") is disregarded because the related values are not variable (all of them are equal to "Name:"), and column 3 ("availability") because the number of non null values is less than 50%. Row 6 (`r.html`) is disregarded for the same reason. Thus, page `r.html` will remain a static page. The final result of database generation is shown at the bottom of Figure 5.

Once template and database have been generated, all static pages migrated to the database can be replaced by a server side script that generates them at run time (`T.php` in Figure 2). Each original request of a migrated page will be replaced by an invocation of `T.php`, with a parameter `page=id` that identifies the information of interest. The server side script uses this parameter to select the record with primary key equal to `id` from the database table gener-

ated before. Each time a DB tag is encountered by T.php, the database is accessed to retrieve the record indexed by id. The field specified as a DB attribute is produced in output, as a replacement of the tag.

When more than one group of static pages can be migrated to dynamic pages (e.g., product cards for different product categories), the process in Figure 2 is iterated, until no useful cluster can be determined. In this way, all site portions that share a common structure are identified and restructured.

The dynamically generated pages are equivalent to those in the purely static version of the Web site (they can also be generated offline, to improve performance), with the remarkable advantage that they are ensured to have all the same structure. The final result is that maintenance of the page specific information is centralized in a database and only one server script has to be modified to correct defects or add new features, instead of an arbitrarily high number of static pages.

Up to this point, the whole process of migration to a dynamic site has been fully automated. However, a final intervention of the user may be necessary to refine the template and the database table automatically generated. If template and database include less (or more) information than desired, the user can edit them directly. Alternatively, the user can select a different cluster for *Template extraction*, so as to have fewer, more similar pages (more fields), or additional pages that introduce further constraints (less fields). This is easily achieved by changing the cut point in the hierarchy of clusterings and re-executing *Template extraction* and *Database generation* (both of which are completely automatic).

3.1 Multilingual pages

The structural edit distance described in section 2 has been used in two different ways: excluding the text between tags (for *Clustering*) or including the text between tags (for *Template extraction*). In presence of pages in different languages (multilingual Web sites), the comparison of the text between tags cannot be a simple comparison of strings. Two texts in different languages are matched if they are the translation of each other, rather than being exactly the same.

In presence of multilingual pages, it is always possible to split the Web site into mono-lingual portions (*Language division* phase) and apply the process in Figure 2 to each subsite. However, additional benefits are achieved if a cross-language template and a unified database are generated.

In the example in Figure 1, if page q.html is an Italian page, instead of an English page, the text within <TITLE> tag and the texts within the two <TH> tags will be translated ("Scheda prodotto", "Nome:" and "Prezzo:" respectively). As a consequence, the extraction of a common

```
<HTML>
<HEAD>
   <TITLE>
   <ML lang="en"> Product card </ML> </TITLE>
   <ML lang="it"> Scheda prodotto </ML> </TITLE>
   </TITLE>
</HEAD>
<BODY>
   <TABLE border="1">
   <TR>
      <TH>
      <ML lang="en"> Name: </ML>
      <ML lang="it"> Nome: </ML>
      </TH>
      <TD> <DB table="table1" field="name" / > </TD>
   </TR>
   <TR>
      <TH>
      <ML lang="en"> Price: </ML>
      <ML lang="it"> Prezzo: </ML>
      </TH>
      <TD> <DB table="table1" field="price" / > </TD>
   </TR>
   </TABLE>
</BODY>
</HTML>
```

Figure 6. *Example of MLHTL template obtained as the LCS from the pages in Figure 1, with* q.html *in Italian.*

cross-language template requires a more sophisticated analysis, exploiting Natural Language Processing (NLP) techniques. More details on this technique are provided elsewhere [12]. The basic idea is to compute an index (called TCI, Translation Correspondence Index), between 0 and 1, measuring the likelihood that a text be the translation of another text:

$$TCI = \frac{content\ words\ with\ translation}{total\ number\ of\ content\ words} \qquad (1)$$

Its computation is based on the entries found in a bilingual electronic dictionary. If the TCI exceeds a given threshold (currently set to 0.3, as determined experimentally [12]), the two texts are considered matching.

In order to generate a cross-language template and database, the Web site is first partitioned according to the language of each page (more details on the related procedure are given in [12]). Then, clustering is used within each language to extract mono-lingual templates. Finally, templates are merged into a common representation. For such a purpose, a possible choice is the adoption of ML-HTML [12]. This slight extension to XHTML (the new version of HTML compliant with XML, promoted by the W3C consortium) adds just one new tag, ML, to the language. When a language-dependent content is embedded in the page, it is surrounded by an ML tag, with an attribute

Web site	Pages	Cards	Languages
www.motturasergio.it	41	16	Italian/English
www.anticahirpinia.it	36	16	Italian/English
pwh.tin.it/aldicu/Alberi	26	25	Italian
www.bio.unipg.it/ittiofauna	33	25	Italian
www.federcoopesca.it/Iniziativa/Progetti	32	29	Italian
www.giornaledibrescia.it/iniziative/funghi	35	32	Italian
www.dambravini.com	63	32	Italian/English
www.ficsvernici.it	44	35	Italian
www.promoturpejo.it/estate/micologia	41	36	Italian
www.assfor.it/funghi	53	49	Italian
www.acquariofiliaitalia.it/pescimarini	103	102	Italian
Total	507	397	

Table 1. *Features of the analyzed sites.*

`lang` specifying the language. Alternatively, a database table can be generated storing the multilingual contents found in the templates. A single template is thus obtained, where multilingual texts are replaced by `DB` tags pointing to proper records of the newly generated database table.

The server side script that constructs the pages dynamically has an additional parameter (`lang`) specifying the selected language. Values are retrieved from the database table in the language indicated by such a parameter.

Figure 6 shows the MLHTML template for the pages in Figure 1 (assuming `q.html` translated in Italian), as produced by the LCS algorithm. Each multilingual text that is part of the template (i.e., is replicated unchanged in all pages within each language) is surrounded by an `ML` tag with the specification of the related language. This is the case of the text within `TITLE` and `TH` tags. Variable parts of the page are retrieved from the database. They are indicated within a `DB` tag. Specification of the language is not necessary, since the template is instantiated with a parameter, `lang`, holding the selected language. Such a parameter is used also for the selection of the appropriate information, when this is extracted from the database.

In presence of multilingual pages, the benefits of the proposed restructuring process are even more evident (see also [12]). In addition to the separation of data from presentation and of the simplification of the source code to mantain, the resulting Web site ensures a consistent presentation of information in all languages. Usage of MLHTML for the common template centralizes maintenance of all different versions for all supported languages.

4 Experimental Results

A preliminary evaluation of the restructuring process described in the previous section has been conducted on a sample of Web sites, chosen among those that seemed to include pages with product-cards or, more generally, repeated descriptions of items, and that are not generated dynamically. The selected sites have been downloaded by means of the tool **ReWeb** [10].

Table 1 contains some information about the eleven Web sites analyzed. In all these sites, there is no easy way to separate pages with cards from other Web pages. There is no directory containing only cards and there is no recognizable naming convention to identify them.

After downloading all Web site pages, their internal structure has been analyzed. The JavaCC HTML parser written by Brian Goetz (`www.quiotix.com`) has been employed to obtain the syntax tree associated to each page. Syntax trees are decorated with attributes, such as *text*, which are used by subsequent analysis phases. The *text* attribute is used to recognize the page language and in the LCS computation, to generate the template, while the tree structure is exploited to compute the edit distance for clustering.

Multi-lingual sites have been split into mono-lingual portions by applying the *Language division* algorithm. The implementation of this algorithm [12] is based on the electronic dictionaries distributed with the UNIX utility `aspell`.

After this preliminary phase, for each Web site in the test suite, the restructuring process, presented in the previous section (Figure 2), has been applied to generate the database (in case of multi-lingual Web sites, a table for each mono-lingual sub-site), the template `T.html` and the server script `T.php`. A summary of the results obtained on the available test suite is shown in Table 2.

This table is divided into ten columns. Web sites restructured, cut point and number of pages of the cluster (*Size*), selected by our algorithm for the computation of the tem-

Web site	Cut point	Size	Recs	Add	Del	Fields	Add	Del	Vals add/del
www.motturasergio.it/ita	3	4	9/9	0	0	12/13	1	9	0
www.motturasergio.it/eng	3	4	7/7	0	0	5/12	7	0	0
www.anticahirpinia.it/ita	4	5	8/8	0	0	4/4	0	1	0
www.anticahirpinia.it/eng	2	4	8/8	0	0	4/4	0	1	2
pwh.tin.it	2	5	25/25	0	0	8/9	1	0	0
www.bio.unipg.it	1	6	25/25	0	0	7/7	0	4	2
www.federcoopesca.it	1	11	29/29	0	0	10/12	2	0	2
www.giornaledibrescia.it	1	5	32/32	0	0	5/10	5	1	0
www.dambravini.com/ita	3	4	16/16	0	0	9/10	1	6	0
www.dambravini.com/eng	3	5	16/16	0	0	9/10	1	6	0
www.ficsvernici.it	4	6	31/35	4	0	14/16	2	6	52
www.promoturpejo.it	1	6	35/36	1	1	3/6	3	0	3
www.assfor.it	3	4	47/49	2	0	10/12	2	0	60
www.acquariofiliaitalia.it	1	21	102/102	0	0	8/9	1	5	0
Total			390/397	7	1	108/134	26	39	127

Table 2. *Results of* Database generation*: cut point, size of selected cluster, records (recognized, added, deleted), fields (recognized, added, deleted), and field values (added, deleted).*

plate, are in the first three columns. *Recs* gives the number of correct records recovered by the *Database generation* algorithm, over the total number of records to be retrieved (according to a manual inspection of the Web site), while *Fields* gives the same measure on the recovered fields. *Add* and *Del* give, for records and fields respectively, the number of row and column additions/deletions that the user has to make to produce the final database. *Vals add/del* is the total number of value insertions or deletions in the database to be performed on the field values determined for each record.

Data in Table 2 show that the entire process of restructuring described before works quite well on the chosen examples. Some final user refinement interventions are always necessary, but they are generally limited to a few operations, accounting for less than 1 hour per site.

In particular the additions of records and fields, a very expensive operation from the point of view of the user, are not so many. The worst case of record insertions is `www.ficsvernici.it`, where the user has to insert four records. The *Database generation* algorithm was not able to recover these four records because these cards are not concerned with paints as the other 31 cases but with dilutors, with some field names different.

The low number of fields automatically extracted in the English portion of the site `www.motturasergio.it` (5 out of 12) is due to free Italian to English translations of field names, with variants in different cards. In some English cards "caratteristiche del vigneto" has been translated as "Characteristics of the vineyard", in others just as "Vineyard". The words "Perfume" and "Arome", and the words "Taste" and "Flavour", have been used as synonyms. These linguistic variants make the automatic extraction of these fields a hard task. By choosing a different cluster for *Template extraction* (cut point 1 instead of 3), the results are improved. In particular, the number of recovered fields becomes 12 out of 12. However, the number of *Vals add/del* increases, because when the whole site is analyzed for the insertion of database records, the linguistic variants make it anyway impossible to determine several values.

The number of record and field deletions is generally small. The worst case of field deletions is in the Italian portion of `www.motturasergio.it` (9 columns deleted), followed by `www.dambravini.com`, Italian and English portions (6 columns deleted for each mono-lingual sub-site), and `www.ficsvernici.it` (6 columns deleted). In `www.motturasergio.it` the algorithm recovers also wine names as fields, because the menu in each page changes (the HREF for the current page is absent), thus making the successive text variable. In `www.dambravini.com`, the extra fields extracted are due to the presence of a `<BR>` tag inside some composite field names (as, for example, in "Technical `<BR>`Production"), which confuse our field extraction algorithm.

Generally, the number of value additions and deletions (*Vals add/del*) is very low. Even in the worst cases (`www.ficsvernici.it` and `www.assfor.it`), this number is still reasonable, if compared to the total number of values retrieved (434 and 470 respectively). The main reasons for the inaccurately retrieved values are miss-

PageId	Antica Hirpinia (Old Hirpinia)	Uvaggio (Grapes)	Zona di Produzione (Production Country)	Altitudine (Altitude)	Conservazione (Conservation)
aglianico	I. G. T.	Aglianico 100%	Colline dell'Irpinia Centrale	350/450m	botti di rovere
codadivolpe	I. G. T	Coda di Volpe 100%	Colline dell'Irpinia Centrale	350/450m	acciaio inox
fiano	D. d. O.	Fiano 100%	Avellino e colline limitrofe	300/400m	acciaio inox
fiordivigna	I. G. T	Aglianico 100%	Colline dell'Irpinia Centrale	350/450m	acciaio inox
greco	D. d. O.	Greco 100%	Tufo, Santa Paolina, Altavilla	450/600m	acciaio inox
sofia	I. G. T	Fiano 80% Coda di Volpe 20%	Colline dell'Irpinia Centrale	400/450m	acciaio inox
taurasi	D. d. O.	Aglianico 100%	Area dei Colli Taurasini	350/400m	botti di rovere
vignare	I. G. T	Aglianico 100%	Taurasi	360m	rovere e carati

Table 3. *The Italian Anticahirpinia database generated by our automatic process (I. G. T. = "Indicazione Geografica Tipica", D. d. O. = "Denominazione di Origine").*

ing texts (as assessed manually) in the Web pages, spelling errors in the field names ("mofologia" instead of "morfologia"), or modified field names ("Cappello -" instead of "Cappello:").

4.1 A complete example

In the following, the operations involved in the restructuring process are considered in detail on a small example.

`Www.anticahirpinia.it` is a bi-lingual Web site of a small wine factory. This Web site consists of 36 static HTML pages: 8 HTML pages are Italian wine cards, 8 pages are the corresponding cards in English, while the other 20 pages (in English and in Italian) contain descriptive material on the company or provide services (e.g., to order products on-line).

The result of the language division phase was 30 correct language classifications out of 36 pages analyzed. Five incorrect and 1 "Don't Know" outputs are associated to pages with very little text inside. Considering only cards, the result of language division was 16 correct language classifications out of 16 cards, with no error.

Italian clusterings:

```
Clustering Number: 4
Solution: [C1{azienda}, C2{biancoerosso}, C4{contatti},
C5{fiano}, C7{form}, C8{framealto}, C9{framecent},
C10{framecentvini}, C11{framelat}, C12{framelatvini},
C13{greco}, C15{index}, C17{taurasi}, C20{homeit, vini},
C23{sofia, fiordivigna, codadivolpe, vignare, aglianico}]
```

Figure 7. *Example of clusterings for the Italian portion of* `www.anticahirpinia.it`.

When the clustering algorithm is applied to the Italian portion of the Web site, 15 possible partitions are obtained. Among these, the partition number 4 (cut point equal to 4) has been chosen automatically by our algorithm, because it contains the cluster C23 (see Figure 7), with five pages (criterion: "min height & #pages >= 4" satisfied).

By applying the LCS algorithm to the set of pages in C23 (*Template extraction* phase), the Italian template has been obtained. It contains the list of potential fields ("Antica Hirpinia", "Uvaggio", "Zona di Produzione", "Altitudine" and "Conservazione"), used to automatically generate a data base with the variable data (see Table 3). According to a visual inspection of the Web site, the database generated in this case is almost perfect. The user has to cancel only the column *Antica Hirpinia*. When navigating the new Web site, the server script `T.php`, extracting the information specified in each `DB` tag from the generated database, is referenced instead of the initial set of (inconsistent) static HTML pages.

When repeating the restructuring process on the English portion of the site, 11 partitions are obtained and the cluster C17 (cut point equal to 2), with 4 pages, is chosen for template generation. Given the template, a data base containing texts in English has been automatically generated. Even in this case, the intervention of the user on the database is minimal, being limited just to canceling a column (again, "Antica Hirpinia").

To obtain additional benefits and a unified database, the Italian and English templates produced by clustering within each mono lingual site portion have been merged into a single, common template. A (simplified) version of the cross-language template (for readability, images and some tag attributes have been removed) is shown in Figure 8.

5 Conclusions and future work

Eleven static Web sites have been restructured into dynamic Web applications by automatically generating a tem-

```
<HTML>
<HEAD> <TITLE> ANTICA HIRPINIA </TITLE> </HEAD>
<BODY> <TABLE>
    <TR> <TD>
        <P>
            <ML lang="en"> Grapes </ML>
            <ML lang="it"> Uvaggio </ML>
        </P>
        <P> <DB table="table1" field="field1" </P>
        <P>
            <ML lang="en"> Production country </ML>
            <ML lang="it"> Zona di produzione </ML>
        </P>
        <P> <DB table="table1" field="field2" </P>
        <P>
            <ML lang="en"> Altitude </ML>
            <ML lang="it"> Altitudine  </ML>
        </P>
        <P> <DB table="table1" field="field3" </P>
        <P>
            <ML lang="en"> Conservation </ML>
            <ML lang="it"> Conservazione  </ML>
        </P>
        <P> <DB table="table1" field="field4" </P>
    </TD> </TR>
    <TR> <TD>
        <P> <DB table="table1" field="field5" </P>
    </TD> </TR>
</TABLE> </BODY>
</HTML>
```

Figure 8. *Simplified MLHTML template of* `www.anticahirpinia.it`.

plate and isolating the variable information to be moved to a data base. Clustering has revealed itself as an effective method to recognize Web pages with cards. The computation of the Longest Common Subsequence has provided an automatic way to generate the page template to use for dynamic page generation. The amount of manual work necessary to refine it was definitely reasonable and limited to a few additions and deletions of records, fields or values.

The proposed approach is expected to improve the maintainability of the considered Web sites. Page information is stored in a data base, while the presentation format is defined in a single place, the template used in dynamic generation, for all cards. This ensures consistency of the displayed information. Multi-lingual data have been handled by performing the migration to a dynamic site of each mono-lingual portion and then merging the resulting templates and databases.

Future work will be devoted to a further in field experimentation and to automating (as much as possible) the consistency checks that are necessary when multilingual databases (and templates) are merged.

References

[1] N. Anquetil and T. C. Lethbridge. Experiments with clustering as a software remodularization method. In *Proc. of the 6th Working Conference on Reverse Engineering (WCRE'99)*, pages 235–255, Atlanta, Georgia, USA, October 1999. IEEE Computer Society.

[2] 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.

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

[4] 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.

[5] M. Harman, R. Hierons, and M. Proctor. A new representation and crossover operator for search-based optimization of software modularization. In *Proc. of the AAAI Genetic and Evolutionary Computation COnference 2002 (GECCO)*, pages 1359–1366, New York, USA, July 2002.

[6] G. A. D. Lucca, A. R. Fasolino, U. D. Carlini, F. Pace, and P. Tramontana. Comprehending web applications by a clustering based approach. In *Proc. of the 10th International Workshop on Program Comprehension (IWPC)*, pages 261–270, Paris, France, June 2002. IEEE Computer Society.

[7] G. A. D. Lucca, M. D. Penta, and A. R. Fasolino. An approach to identify duplicated web pages. In *Proc. of the 26th Annual International Computer Software and Applications Conference (COMPSAC)*, pages 481–486, Oxford, England, August 2002. IEEE Computer Society.

[8] S. Mancoridis, B. S. Mitchell, Y. Chen, and E. R. Gansner. Using automatic clustering to produce high-level system organizations of source code. In *Proc. of the International Workshop on Program Comprehension*, pages 45–52, Ischia, Italy, 1998.

[9] S. Mancoridis, B. S. Mitchell, Y. Chen, and E. R. Gansner. Bunch: a clustering tool for the recovery and maintenance of software system structures. In *Proceedings of the International Conference on Software Maintenance*, pages 50–59, Oxford, England, 1999.

[10] 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.

[11] F. Ricca, P. Tonella, and I. Baxter. Web application transformations based on rewrite rules. *Information and Software Technology*, 44(13):811–825, 2002.

[12] P. Tonella, F. Ricca, E. Pianta, and C. Girardi. Restructuring multilingual web sites. In *Proc. of the International Conference on Software Maintenance (ICSM 2002)*, pages 290–299, Montreal, Canada, October 2002. IEEE Computer Society Press.

[13] T. Wiggerts. Using clustering algorithms in legacy systems remodularization. In *Proc. of the 4th Working Conference on Reverse Engineering (WCRE)*, pages 33–43. IEEE Computer Society, 1997.