JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. 2007; **19**:281–296
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/smr.359

Special Issue

Identifying similar pages in Web applications using a competitive clustering algorithm

Andrea De Lucia[1], Giuseppe Scanniello[2,*,†] and Genoveffa Tortora[1]

[1]*Dipartimento di Matematica e Informatica, Università di Salerno Via Ponte don Melillo, 84084 Fisciano (SA), Italy*
[2]*Dipartimento di Matematica e Informatica, Università della Basilicata, Viale dell'Ateneo, 10 Macchia Romana, 85100 Potenza, Italy*

SUMMARY

We present an approach based on Winner Takes All (WTA), a competitive clustering algorithm, to support the comprehension of static and dynamic Web applications during Web application reengineering. This approach adopts a process that first computes the distance between Web pages and then identifies and groups similar pages using the considered clustering algorithm. We present an instance of application of the clustering process to identify similar pages at the structural level. The page structure is encoded into a string of HTML tags and then the distance between Web pages at the structural level is computed using the Levenshtein string edit distance algorithm. A prototype to automate the clustering process has been implemented that can be extended to other instances of the process, such as the identification of groups of similar pages at content level. The approach and the tool have been evaluated in two case studies. The results have shown that the WTA clustering algorithm suggests heuristics to easily identify the best partition of Web pages into clusters among the possible partitions. Copyright © 2007 John Wiley & Sons, Ltd.

Received 12 February 2007; Revised 2 July 2007; Accepted 31 July 2007

KEY WORDS: similar pages; cloned pages; comprehension of legacy Web applications

*Correspondence to: Giuseppe Scanniello, Dipartimento di Matematica e Informatica, Università della Basilicata, Viale dell'Ateneo, 10 Macchia Romana, 85100 Potenza, Italy.
†E-mail: giuseppe.scanniello@unibas.it

Contract/grant sponsor: MiUR (Ministero dell'Università e della Ricerca); contract/grant number: PRIN-2006-2006098097

1. INTRODUCTION

Several Web engineering paradigms, methodologies, and approaches have emerged in the last years [1–5] due to the growing interest in Web applications as a mean allowing distributed organizations to communicate, share information, and manage production and distribution. However, the short time-to-market often forces Web applications to be developed and evolved without adopting a disciplined process. In case methodologies that anticipate changes and evolution are not applied, the comprehension, the maintenance, and the evolution of Web applications become harder and harder [6], thus requiring the use of reverse engineering methods and tools during the maintenance and evolution [7–11]. In particular, clustering-based methods and tools [12–16] support the comprehension and the improvement of the design of Web applications.

In this paper we propose an automatic approach based on Winner Takes All (WTA) [17,18], a competitive clustering algorithm, to group similar static and dynamic Web pages, thus improving the comprehension of legacy Web applications. Indeed, the approach is general and is able to identify similar Web pages implemented using any kind of server side scripting code. Distances between the Web pages are computed first and then groups of similar pages are detected using the WTA algorithm. We show an instance of the process in order to detect similar Web pages at the structural level in dynamic and/or static Web sites. Such a technique can be useful during Web application reengineering to identify cloned pages that need to be generalized into a single page [12,14,16]. Similar to the approaches proposed in [12,14,16,19], we use Levenshtein string edit distance [20] as a basic metric to identify page similarity at the structural level. In particular, the Levesthtein algorithm is used to compare strings of HTML tags representing the structure of Web pages. A prototype to automate the clustering process has been implemented that can be extended to other instances of the process, such as the identification of groups of similar pages at the content level. The approach and the tool have been evaluated in two case studies. The results have shown that the WTA clustering algorithm suggests heuristics to quickly identify the best partition of Web pages into clusters among the possible partitions.

The remainder of the paper is organized as follows: Related work is discussed in Section 2. In Section 3 we describe the clustering process and an instance for the identification of groups of similar pages at the structural level. Section 4 presents the system prototype, while the results of the case studies and their discussion are presented in Sections 5 and 6, respectively. Final remarks conclude the paper.

2. RELATED WORK

The research community has extensively studied the problem of defining reverse engineering and analysis techniques for Web applications [7–12,21]. Ricca and Tonella [11] define a conceptual model for representing the structure of a Web site and several structural analyses relying on such a model, ranging from flow analysis to graph traversal algorithms and pattern matching. Di Lucca *et al.* [9,10] propose reverse engineering methods and tools to enable the extraction of the UML extension of Conallen [2] from existing Web applications through the analysis of static and dynamic contents. Differently, Antoniol *et al.* [7] propose a method for re-engineering static Web sites into dynamic Web sites using the Relationship Management Methodology [4,22]. Differently from our approach, these approaches do not consider the problem of identifying cloned pages.

DOI: 10.1002/smr

A large number of approaches and tools have been proposed to identify clones or similar Web pages [8,12,14–16,23,24]. For example, Calefato *et al.* [23] exploit a metric-based approach and a pattern matching algorithm to compare scripting code fragments. However, the authors do not consider the problem of identifying cloned pages, as they take into account neither the structure of Web pages nor their content. Boldyreff and Kewish [8] propose a reverse engineering approach for static and dynamic Web sites, enabling the extraction of styles and contents from Web pages. Content and styles are stored into a database and the HTML pages are modified by using scripts that retrieve information from the database and then dynamically generate the page. Similar page styles are manually identified by the software engineer analyzing the information stored in the system database. Rajapakse and Jarzabek [24] analyze the results achieved by applying different cloning approaches on Web applications of different sizes and developed for different application domains, by teams of different structures and in different development environments. The study revealed that cloning equally affects small, medium, or large Web applications, and the number of clones increases over time. The authors also showed that cloning could be even worse than that of traditional applications.

Similar to our approach, other authors [12,14,16,19] use the Levenshtein string edit distance [20] as a basic metric to identify similarity between the structure of Web pages. Di Lucca *et al.* [19] encode the sequences of tags of HTML and ASP pages into strings and identify pairs of cloned pages at the structural level by computing the Levenshtein distance between the corresponding strings. Unlike our approach, no clustering algorithm is used to group similar Web pages: pages are considered clones if their distance is zero. A similar approach is also proposed by Girardi *et al.* [14] for restructuring multilingual Web sites. On the basis of Levenshtein distance, the authors define a semiautomatic approach to identify and align static HTML pages whose structure is the same and whose content is in different languages. Also in this case no clustering algorithm is used to group similar pages. In [16] the authors enhance the approach based on the Levenshtein distance with a hierarchical clustering algorithm to identify groups of duplicated or similar static pages to be generalized into a dynamic page. Each page is initially inserted into a different cluster and at each step clusters with minimal distance are merged, thus producing a hierarchy of clusters. The clusters of cloned pages are selected by the software engineer, by choosing a cut level of the tree representing the cluster hierarchy. Unlike this approach, we use a partitional clustering algorithm, namely WTA [17,18]. Moreover, we also deal with dynamic Web pages in addition to static pages. De Lucia *et al.* [12] also use the Levenshtein edit distance to compute the similarity of two pages at structure, content, and scripting code level and to identify cloned navigational patterns. No clustering algorithm is used to group similar pages: the similarity measures are compared with thresholds to establish whether two pages are clones.

Hierarchical clustering algorithms have also been used for Web application comprehension [13,15]. Di Lucca *et al.* [13] suggest a clustering method for decomposing Web applications into groups of functionally related components. The authors define a coupling measure based on the number and type of links between pages and use an agglomerative hierarchical clustering algorithm to group pages together, simplify the navigational schema, and make it more understandable. A different approach is proposed by Ricca *et al.* [15]. Similar pages are grouped together based on the similarity of their content and using a hierarchical clustering algorithm. The authors use Natural Language Processing (NLP) techniques to weight each keyword in the text of Web pages and compute the similarity. Despite the similarity with the approaches described above, our

DOI: 10.1002/smr

approach is more general as it can be instantiated to group similar pages according to different similarity/dissimilarity measures and applying different clustering algorithms. Moreover, the hierarchical clustering algorithms used by previous approaches require an intensive interaction of the software engineer. Furthermore, we show how the WTA clustering algorithm can be used to quickly identify the best partition of Web pages into clusters among the possible partitions.

3. CLUSTERING SIMILAR PAGES

Reverse engineering tries to help software engineers in the comprehension of large pieces of software or whole systems. A key activity in reverse engineering consists of gathering the software entities that compose the system into meaningful and independent groups. Such an activity is also known as clustering. Generally, clustering-based approaches require the selection of the concerns to be used to group entities in clusters [25]. Similarity/dissimilarity measures should also be identified in order to compare the entities with respect to the considered concerns.

Figure 1 shows the general process we defined to group static and dynamic Web pages considered similar with respect to a given concern. The rounded rectangles represent process phases, while the rectangles represent intermediate artifacts produced at the end of each phase. The *Page Distance Computation* phase produces a distance matrix of the pages of a given Web application according to the concern to analyze. In our case this phase is refined by the phases *Page Transformation* and *Computing Distance Matrix*. In particular, suitable representations of the pages to consider in the identification of similar pages are produced in the Page Transformation phase. Successively, the Computing Distance Matrix phase uses the page representations to identify distances between the pairs of pages of a given Web application. The distances between pairs of pages are then used to build the distance matrix, which is provided as input to the phase *Clustering Web Pages*. This phase uses clustering algorithms [17,26] to group similar pages with respect to the considered distance measure.

This approach is general and enables the identification of groups of similar pages using suitable distance measures between pairs of pages and any clustering algorithm. In the following we show an instance of the process to identify groups of similar pages at the structural level that represents Web pages as strings of HTML tags and uses the Levensthein edit distance [20] to compute the distance matrix. Other process instances can be based on different Web page representations and different ways to compute the distance between them. For example, NLP processing and information retrieval techniques [27,28] can be adopted to identify groups of similar pages at content level [15,29]. Moreover, our approach can be customized with respect to the clustering algorithm used in the phase Clustering Web Pages. In our case, we adopt the WTA clustering algorithm [17,18].

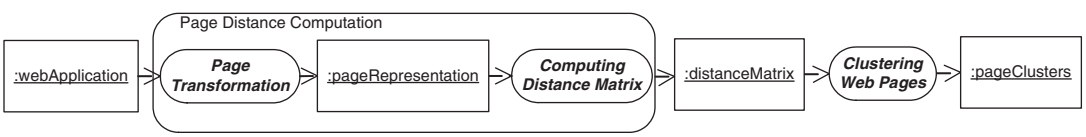

Figure 1. The overall clustering process.

DOI: 10.1002/smr

3.1. Computing the distance between the structure of Web pages

To cluster similar pages at the structural level, the Page Transformation phase produces string representations of static and dynamic pages. Indeed, we produce a representation of the structures through a depth-first traversal of the abstract syntax tree of both static and dynamic pages of a Web application. Each node of the syntax tree of a page is decorated with an HTML tag and with a set of attributes, such as text attribute, target source code, image attribute, etc. It is worth mentioning that the syntax tree of a dynamic page differs from the syntax tree of an HTML page only for the presence of the server-side scripting nodes.

The HTML tags of a Web page are encoded into symbols of an alphabet before being concatenated into the string representing its structure. This enables a more precise computation of the distance of two pages with respect to just concatenating the HTML tags into strings. Encoding the tags also improves the time required for the computation of the distance of two pages.

Once the strings encoding the page structures have been computed, the distance between the pages is computed using the Levenshtein edit distance algorithm [20], one of the most important algorithms for string matching. The Levenshtein model is based on the notion of edit operation and consists of a set of rules that transform a given string into a target string. In particular, given two strings x and y, the Levenshtein edit distance is defined as the minimum number of insert, delete, and replace operations required to turn x into y.

The distance matrix is then built using the Levenshtein distances computed between the Web pages and is provided as input to the phase Clustering Web Pages in order to identify groups of similar pages at the structural level.

3.2. Winner Takes All

In general, clustering algorithms are divided into hierarchical and partitional algorithms [26]. Hierarchical algorithms produce a nested series of partitions, while partitional algorithms produce only one partition. This classification has to be supplemented by a discussion of crosscutting issues that may affect both the hierarchical and partitional methods. Thus, we can further classify clustering algorithms in agglomerative and divisive. An agglomerative approach begins with each entity or observation (a page in our approach) in a distinct cluster, and successively merges clusters together until a stopping criterion is satisfied. Differently, divisive clustering begins with all observations in a single cluster and performs splitting operation until a stopping criterion is verified.

One of the widespread partitional approaches to cluster observations from the statistical recognition perspective is WTA. It is an Artificial Neural Network (ANN) [17,18] clustering algorithm aiming at grouping and representing similar observations by a singular unit also known as neuron. We adopt as ANN the Self-Organizing feature Map (SOM) that in the literature is sometimes called Kohonen self-organizing feature map [30].

The SOM network is based on a two-dimensional grid of neurons, whose weights are continuously adapted to the training set of the clustering algorithm, i.e., the vectors of the distances between the pages of a Web application. The number of neurons is provided as input to the SOM network and represents the number of clusters that the algorithm should identify. This number is *a priori* chosen by the software engineer and represents the tuning value of our approach. To assess the approach in our experiments (see Section 5), we used all configurations ranging from

DOI: 10.1002/smr

the number of Web pages (each page in a different cluster) to 1 (all pages in one cluster). To avoid analyzing the data set before running the algorithm, the initial positions of the neurons on the SOM lattice are randomly assigned. The neuron with weight vector most similar to the input vector is called the Best Matching Unit (BMU) or winner. The weights of the BMU and neurons close to it in the SOM lattice are adjusted toward the input vector. The magnitude of the change decreases with time and is smaller for neurons physically far away from the BMU. This process is also known as training process and is repeated over and over for each page in the input space. The training process is concluded when either the neurons do not change their position on the SOM lattice or a termination threshold (i.e., number of epochs) for the iteration has been reached. The network winds up associating the neurons to groups of similar pages in the input space. In particular, the pages in the input space are associated with the closest neuron (cluster).

It is possible that at the end of the process, some neurons do not have any page associated, i.e., they result in empty clusters. In other words, the number of achieved (non-empty) clusters can be lower than the number of neurons in the network (i.e., the number of expected clusters). Furthermore, we observed that if a network configuration achieves a number of (non-empty) clusters equal to the number of neurons, configurations with less neurons also do not result in empty clusters. Thus, we define as break-even configuration the configuration with a larger number of neurons that results in non-empty clusters. The identification of suitable configurations to detect clusters of similar pages in Web applications is generally expensive and time consuming [12,14–16]. Although the choice of the number of expected clusters (neurons of the network) is also critical, the results of the case studies in Section 5 suggest some guidelines based on the break-even configuration to support the software engineer in pruning bad configurations and quickly identifying the best one.

4. IMPLEMENTATION

The proposed approach has been implemented as a Java prototype. Figure 2 shows the architecture of the tool expressed in terms of a UML Class Diagram. To implement instances of the approach, we have adopted an Abstract Factory Pattern [31]. This pattern enables us to create a family of related objects to identify groups of similar pages with respect to the selected instance of the clustering process. In particular, *ClusteringMode* is the abstract class to be extended when different instances of the clustering process have to be implemented. This class instantiates the sub-classes of the *PageRepresentation* and *DistanceMatrix* abstract classes that implement the components of the selected process instance. Of course, the *PageRepresentation* and *DistanceMatrix* abstract classes should also be extended in case we want to extend the tool with components implementing a new process instance. The PageRepresentation abstract class uses the class *HTMLParser* to extract the needed information from the Web pages. To this end an open source HTML parser written in Java (HTML-Parser version 1.6), available under GPL license from http://sourceforge.net/projects/htmlparser, has been used. The class in charge of concatenating the HTML tags into strings representing the page structure is *HTMLTagRepresentation*. The class *LevenshteinDistanceMatrix* computes the distances between pairs of pages. Distances between pairs of pages are then used to obtain the distance matrix of the static/dynamic pages.

The distance matrices are used by the selected specialization of the class *Clustering Algorithm*, according to a Strategy Design Pattern. In our case, groups of similar pages are identified using

DOI: 10.1002/smr

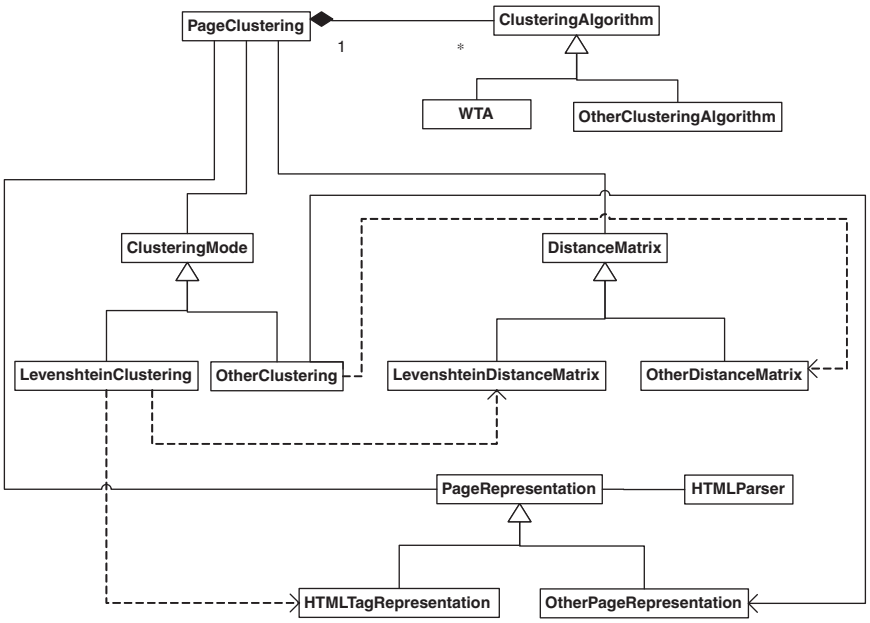

Figure 2. The prototype architecture.

the class *WTA*, which implements the WTA clustering algorithm. Indeed, this algorithm has been implemented in MATLAB and integrated within the prototype using the JMatLink engine, which is available under GPL license from http://www.held-mueller.de/JMatLink/.

5. CASE STUDIES

Although our approach is general, we applied it on two Web applications developed using JSP technology. In particular, we considered the Web application of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 02) and the Web application named SRA (Student Route Analysis)[‡]. SEKE 02 was used to support the organizers and the academic community for paper submission, refereeing, and conference registration. Two joint workshops on Web Engineering and Software Engineering Decision Support also used this application. The SEKE 02 Web application consisted of 157 pages (49 html static pages and 108 jsp dynamic pages) included in one directory without any meaningful classification. The site also contained some other files, such as images, a java applet, java classes, logos, etc., which were used by the Web application and were not considered in the analysis. The developers of SEKE 02 revealed that this Web application was developed without applying any development methodology. On the other hand, the SRA Web application was devised to provide students in Politics Science at University of Salerno

[‡]The pages of the Web applications are available at http://www.scienzemfn.unisa.it/scanniello/WTA/.

DOI: 10.1002/smr

Table I. Statistics on the analyzed Web applications.

	SEKE 02		SRA	
Page type	HTML	JSP	HTML	JSP
Number of pages	49	108	24	45
Actual pairs	19	87	48	15
Similarity density Measure (%)	1.6	1.5	17.4	1.5
Trade-off configuration	31	57	6	27
Break-even configuration	32	61	6	18

with statistics and information on their academic carrier. The information was also presented in English, thus supporting also visiting students. Accredited users could also exploit this application to obtain statistics on the students, visiting students, and graduates. The SRA Web application was composed of 380 files distributed in 45 folders according to a meaningful classification. For example, static pages containing Italian and English contents were grouped in different folders, while dynamic pages were grouped according to the provided functionalities. Overall, the SRA Web application was composed of 24 html static pages and 45 jsp dynamic pages. The developers of SRA stated that both a software development process and a Jakarta Struts development framework [32] were adopted. Some descriptive measures of the Web applications SEKE 02 and SRA are presented in Table I.

5.1. Assessment of the results

We adopted two well-known metrics, namely precision and recall, for the analysis of the results produced by the tool. To this aim, the actual groups of similar pages were manually produced by two Web development experts who were not in the development team of the proposed prototype. The two experts worked independently and iterated until they reached an agreement. The clusters manually identified were then turned into pairs of similar pages (or cloned pages). For example, if (P_1, P_2, P_3) is a cluster manually identified, the corresponding pairs of similar pages are (P1, P2), (P1, P3), (P2, P3). The clusters of similar pages identified by the tool were also turned into pairs of similar pages. The pairs of similar pages manually identified and those automatically identified by the tool were included in a gold matrix and in a result matrix, respectively. Both the gold and result matrices are symmetric with respect to their northwest–southeast diagonal due to the reflexivity and symmetricity of the clone relation.

In our case the recall is defined as the ratio between the number of actual pairs of similar pages identified by the tool over the total number of actual pairs of similar pages, while the precision is the number of actual pairs of similar pages identified by the tool over the total number of identified pairs. Therefore, if A is the set of pairs of similar pages identified by the tool and B is the set of actual pairs of similar pages, the recall and the precision are defined as follows:

$$\text{recall} = \left| \frac{A \cap B}{B} \right|, \quad \text{precision} = \left| \frac{A \cap B}{A} \right|$$

The evaluation of precision and recall is not straightforward. Thus, to better assess the achieved results, we also consider the trade-off between the precision and recall values. This trade-off measure

DOI: 10.1002/smr

enables us to better evaluate the results obtained when using different neural network configurations, i.e., neural networks with different number of neurons (expected clusters). In particular, we consider the F-measure, which is defined as harmonic mean of precision and recall. This measure is defined as follows:

$$2 * \frac{precision * recall}{precision + recall}$$

We define as trade-off configuration the neural network configuration that achieves the largest F-measure value. We considered all neural network configurations with a number of neurons ranging from the number of static or dynamic pages of the Web application to 1 and computed for each configuration the precision, the recall, and the F-measure. For space reasons, the tables in the following subsections show only the meaningful configurations.

The number and the density of the actual pairs of similar static and/or dynamic pages are also shown in Table I. We define the similarity density measure as the ratio between the number of actual pairs of similar pages over the number of possible pairs of similar pages in a Web application. Table I also presents the trade-off and the break-even configurations of the considered case studies.

5.2. Results of the SEKE 02 Web application

The results of the approach for the static and dynamic pages of the SEKE 02 Web application are presented in Tables II and III, respectively. The values of the trade-off configurations are in

Table II. Results on SEKE 02 static page structures.

Expected clusters	Identified clusters	Precision	Recall	Sum	F-measure
49	45	1	0.2105	1.211	0.3478
47	44	1	0.2632	1.263	0.4167
46	43	0.8333	0.2632	1.096	0.4
44	41	0.75	0.3158	1.066	0.4444
42	39	0.7	0.3684	1.068	0.4828
39	37	0.5833	0.3684	0.9518	0.4516
36	35	0.5333	0.4211	0.9544	0.4706
35	34	0.5625	0.4737	1.036	0.5143
34	33	0.5	0.4737	0.9737	0.4865
32	32	0.4737	0.4737	0.9474	0.4737
31	***31***	***0.5238***	***0.579***	***1.103***	***0.55***
29	29	0.3929	0.579	0.9718	0.4681
28	28	0.4	0.6316	1.032	0.4898
20	20	0.3	0.7895	1.089	0.4348
19	19	0.2885	0.7895	1.078	0.4225
18	18	0.2778	0.7895	1.067	0.411
17	17	0.2759	0.8421	1.118	0.4156
13	13	0.1977	0.8947	1.092	0.3238
12	12	0.1758	0.8421	1.018	0.2909
10	10	0.1126	0.8947	1.007	0.2
7	7	0.085	0.8947	0.9797	0.1553
6	6	0.06597	1	1.066	0.1238
1	1	0.01616	1	1.016	0.0318

DOI: 10.1002/smr

Table III. Results on SEKE 02 dynamic page structures.

Expected clusters	Identified clusters	Precision	Recall	Sum	F-measure
108	95	0.913	0.2414	1.154	0.3818
96	85	0.6176	0.2414	0.859	0.3471
95	84	0.6286	0.2529	0.8814	0.3607
92	82	0.5946	0.2529	0.8475	0.3548
91	81	0.6053	0.2644	0.8696	0.368
85	78	0.561	0.2644	0.8253	0.3594
84	77	0.5698	0.2816	0.8514	0.3769
83	77	0.5698	0.2816	0.8514	0.3769
82	76	0.5667	0.2931	0.8598	0.3864
81	75	0.5426	0.2931	0.8357	0.3806
80	74	0.5521	0.3046	0.8567	0.3926
63	62	0.363	0.3046	0.6676	0.3312
62	61	0.3654	0.3276	0.693	0.3455
61	61	0.3654	0.3276	0.693	0.3455
60	60	0.3563	0.3276	0.6838	0.3413
59	59	0.3598	0.3391	0.6988	0.3491
58	58	0.4415	0.477	0.9185	0.4586
57	***57***	***0.5227***	***0.6609***	***1.184***	***0.5838***
56	56	0.518	0.6609	1.179	0.5808
39	39	0.3267	0.6609	0.9876	0.4373
38	38	0.3104	0.6494	0.9599	0.4201
37	37	0.3087	0.6494	0.9582	0.4185
36	36	0.2964	0.6609	0.9573	0.4093
35	35	0.2985	0.6724	0.9709	0.4134
34	34	0.3128	0.7874	1.1	0.4477
23	23	0.2283	0.7874	1.016	0.354
22	22	0.2184	0.7759	0.9943	0.3409
14	14	0.1211	0.7644	0.8855	0.2091
13	13	0.1191	0.7874	0.9065	0.2069
12	12	0.1171	0.8103	0.9275	0.2046
11	11	0.1105	0.8103	0.9208	0.1945
10	10	0.1059	0.8103	0.9162	0.1873
1	1	0.01506	1	1.015	0.02967

bold-italic. The first column reports the number of expected clusters (neurons of the network), while the number of clusters identified by the tool is reported in the second column. The corresponding values of precision and recall are reported in the third and fourth columns, respectively. The sum of the precision and recall values is shown in the fifth column, while the last column shows the F-measure (i.e., the harmonic mean of the precision and recall values).

Despite the small difference between the density of the actual pairs of similar static and dynamic pages (see Table I), the approach generally produced better results when applied on the structure of the dynamic Web pages. Indeed, the trade-off configuration resulting from clustering dynamic pages contains 57 clusters (see Table III). In this case we obtained 0.6609 as recall and 0.5227 as precision, respectively. On the other hand, when clustering static pages the trade-off configuration contains 31 clusters (see Table II) and in this case we obtained 0.5238 as precision and 0.579 as recall, respectively.

DOI: 10.1002/smr

Inspecting the static pages of the Web application, we observed that it included dynamic pages with a very structured layout and pages with no or very poor layout, which were developed to enable the members of the organizing committee to achieve statistics about the conference. The majority of the pages with very structured layout had a column including the same navigational menu on the left-hand side and the conference name and logo on the top. This resulted in a low Levenshtein distances between the strings encoding the page structure and in many false positives during the identification of the clusters of cloned pages.

5.3. The SRA Web application

The results obtained by applying the approach on the static and dynamic pages of the SRA Web application are shown in Tables IV and V, respectively. The approach generally produced better results when applied on the static Web pages. The trade-off configuration resulting from clustering static pages contains 6 clusters (see Table IV). In this case we obtained 0.9216 and 0.9792 as precision and recall values, respectively. On the other hand, when clustering dynamic pages the trade-off configuration contains 27 clusters (see Table V) and in this case we obtained 0.4 as recall and 0.2069 as precision, respectively. Let us note that the precision and recall values obtained by applying the approach on the static pages are much better than the results achieved on the dynamic pages. This result suggests that our approach produces better results when applied on Web applications with high values of density of actual pairs. In fact, the value of the similarity density of actual pairs is 17.4% for the static pages of the SRA Web application, while it is 1.5% for the dynamic page.

The better results obtained by applying the approach on the static pages were further confirmed by analyzing the sums of the precision and recall values. In fact, the smaller sum of the precision and recall values obtained on the structure of the static pages is larger than the larger sum of the precision and recall values obtained on the dynamic pages of the SRA Web application.

We inspected the pages of SRA to comprehend the results obtained by applying the approach on its static and dynamic Web pages. Indeed, the inspection process revealed a lot of static pages nearly identical at the structural level. This was due to the fact that each static page with Italian content had at least a corresponding static page with content in English and with the same structure. Differently, only few dynamic pages had similar structure, as also the results achieved by applying the approach revealed (i.e., the majority of the clusters were composed of a single page).

Table IV. Results on SRA static page structures.

Expected clusters	Identified clusters	Precision	Recall	Sum	F-measure
24	9	1	0.8125	1.813	0.8966
15	8	0.9512	0.8125	1.764	0.8764
10	7	0.8298	0.8125	1.642	0.821
7	6	0.7959	0.8125	1.608	0.8041
6	***6***	***0.9216***	***0.9792***	***1.901***	***0.9495***
5	5	0.8246	0.9792	1.804	0.8952
4	4	0.6438	0.9792	1.623	0.7769
3	3	0.4563	0.9792	1.435	0.6225
2	2	0.3333	0.9792	1.313	0.4974
1	1	0.1739	1	1.174	0.2963

DOI: 10.1002/smr

Table V. Results on SRA dynamic page structures.

Expected clusters	Identified clusters	Precision	Recall	Sum	*F*-measure
45	44	1	0.06667	1.067	0.125
43	42	0.6667	0.1333	0.8	0.2222
42	41	0.5	0.1333	0.6333	0.2105
37	36	0.1667	0.1333	0.3	0.1481
35	34	0.2	0.2	0.4	0.2
31	30	0.15	0.2	0.35	0.1714
30	29	0.1905	0.2667	0.4571	0.2222
29	28	0.16	0.2667	0.4267	0.2
28	27	0.1923	0.3333	0.5256	0.2439
27	***26***	***0.2069***	***0.4***	***0.6069***	***0.2727***
26	25	0.1875	0.4	0.5875	0.2553
25	24	0.1765	0.4	0.5765	0.2449
24	23	0.1667	0.4	0.5667	0.2353
23	22	0.15	0.4	0.55	0.2182
22	21	0.1463	0.4	0.5463	0.2143
21	20	0.1667	0.4667	0.6333	0.2456
20	19	0.1321	0.4667	0.5987	0.2059
18	18	0.1273	0.4667	0.5939	0.2
15	15	0.1475	0.6	0.7475	0.2368
13	13	0.1304	0.6	0.7304	0.2143
12	12	0.1176	0.6667	0.7843	0.2
5	5	0.03834	0.8	0.8383	0.07317

6. DISCUSSION

The identification of appropriate cluster decompositions and the assessment of the achieved results are generally complex. Of course, the larger the size of a Web application, the more expensive and time consuming the assessment of the obtained results is. This is essentially due to the fact that results of the tool have to be manually analyzed and validated. Thus, methods to automatically filter out surely bad tuning configurations and to get more quickly good configurations should be devised in order to support software engineers.

Although we did not aim at proposing a general strategy to select a neural network configuration allowing to achieve the best (or at least a good) trade-off between the values of precision and recall, the results of the case studies suggest some directions to support the software engineer in the identification of such a configuration by restricting the analysis to a subset of possible configurations. To this aim, we have analyzed the trade-off and break-even configurations and corresponding results in terms of precision and recall values.

We observed that in most cases the trade-off configuration is very close to the break-even configuration (see Table I). It is worth noting that the break-even configuration can be easily identified by trying the approach with all possible neural network configurations and looking for the configuration where the number of empty clusters becomes 0 for the first time. Once the break-even configuration is achieved, the software engineer can analyze the configurations only with more or less neurons than the break-even configuration within a given range to approach the trade-off configuration.

DOI: 10.1002/smr

In this way, only a relatively small number of configurations are analyzed, thus reducing the effort required to approach the trade-off configuration.

We also observed that when the density of actual pairs of similar pages is high the approach generally produced better results in terms of precision and recall values. For instance, high values of precision and recall were obtained on the structures of the static pages of the SRA Web application. As shown in Table I, this Web application presented a high value of the similarity density measure on the static pages (17.4%) and in this case the break-even and the trade-off configurations coincide. A similar result was also achieved by applying the approach on the structures of the static pages of the SEKE 02 Web application, where the distance between the break-even and trade-off configurations is 1. However, in this case the value of the similarity density measure is only 1.6%.

The analyzed Web applications also revealed that in case the trade-off and the break-even configurations are different and the Web applications have a large number of pages similar at the structural level, the trade-off and the break-even configurations produced similar results. For example, on the static pages of the SEKE 02 Web application, we achieved similar values of precision and recall by selecting as tuning values both the break-even configuration and trade-off configuration (see Table II). It is worth noting that this result is always verified except for the dynamic pages of the SEKE 02 Web application. Nevertheless, on the structure of the dynamic pages of this application, the difference between the break-even and trade-off configurations is low (see Table I), thus enabling the software engineer to quickly filter out the bad tuning values around the break-even configuration.

Generally, we also observed the case when the Web applications present only few static or dynamic pages similar at the structural level, the break-even configuration and the trade-off configuration are different and the obtained results are in general bad. For example, the SRA Web application has a huge number of dynamic Web pages with different structures. In this case, the break-even and the trade-off configurations are different and in general the results in terms of precision and recall are quite bad and the sum of precision and recall corresponding to the trade-off and break-even configurations is 0.6069 and 0.5939, respectively.

The obtained results enable us to believe that the approach should produce appreciable results also on different Web applications. In fact, the majority of the problems that can be found in the comprehension of legacy static and dynamic Web sites are proposed by the selected case studies. For example, as mentioned above the SEKE 02 Web application was developed without using systematic development methodology and its requirements were unclear and changing. Both the static and dynamic pages were stored without a meaningful classification on the file system of the Web server. Differently, SRA was a multi-language Web application and was developed using a systematic development process and a Jakarta Struts [32] as development framework. This application contained a considerable number of cloned static pages at the structural level.

7. CONCLUSION

In this paper we have presented an approach based on a competitive clustering algorithm, namely Winner Takes All (WTA), to comprehend legacy Web applications. We have also shown an instance of the approach to identify similar static and dynamic pages at the structural level. Page structure is implemented by specific sequences of HTML tags and compared using the Levenshtein algorithm

DOI: 10.1002/smr

to obtain the corresponding distance measure. WTA uses the Levenshtein distances of the page representations to group similar pages at the structural level.

A prototype to automate the identification of groups of similar pages has also been proposed. We also reported the results of applying our approach and prototype on two Web applications with different sizes and characteristics, all implemented using J2EE technology. The analysis of the results of these case studies enabled us to identify some guidelines to reduce the number of configurations to analyze to approach the configuration providing the best partition of clusters of similar pages.

To further confirm such a result, we also plan to apply the approach on Web applications implemented using different technologies and coming from different application fields. Future work will also be devoted to customize our approach for the identification of clusters of similar pages at content level and apply it on different case studies. Finally, we also plan to extend our tool to experiment and compare different clustering algorithms.

ACKNOWLEDGEMENTS

The authors would like to thank Angelo Ciaramella, Rocco Oliveto, and Michele Risi for the stimulating discussion and their precious suggestions. Special thanks are also due to Francesco Casertano, who implemented some software components of the system prototype.

The work described in this paper is supported by the project METAMORPHOS (MEthods and Tools for migrAting software systeMs towards Web and service Oriented aRchitectures: exPerimental evaluation, usability, and technology tranSfer), funded by MiUR (Ministero dell'Università e della Ricerca) under grant PRIN-2006-2006098097.

REFERENCES

1. Ceri S, Fraternali P, Bongio A. Web modeling language (WebML): A modeling language for designing Web sites. *Computer Networks*, *9th World Wide Web Conference*, 2000; 137–157.
2. Conallen J. *Building Web Application with UML*. Addison Wesley: Reading MA, 2000.
3. Ginige A, Murugesan S (guest eds). Special issue on Web engineering. *IEEE Multimedia* 2000; **18**(1–2):14–18.
4. Isakowitz T, Stohr EA, Balasubramanian P. RMM: A methodology for structured hypermedia design. *Communications of the ACM* 1995; **38**(8):34–44.
5. Schwabe D, Rossi G. Developing hypermedia applications using OOHDM. *Proceedings of the Workshop on Hypermedia Development Process*, *Methods and Models*, Hypertext 98, 1998.
6. Boldyreff C, Tonella P. Web site evolution. *Journal of Software Maintenance* 2004; **16**(1–2):1–4.
7. Antoniol G, Canfora G, Casazza G, De Lucia A. Web Site Reengineering using RMM. *Proceedings of the International Workshop on Web Site Evolution*, 2001. IEEE CS Press: Silver Spring MD, 2000; 9–16.
8. Boldyreff C, Kewish R. Reverse engineering to achieve maintainable WWW sites. *Proceedings 8th IEEE Working Conference on Reverse Engineering*, 2001. IEEE CS Press: Silver Spring MD, 2001; 249–257.
9. Di Lucca GA, Fasolino AR, Tramontana P. Reverse engineering Web applications: The WARE approach. *Journal of Software Maintenance and Evolution*: *Research and Practice* 2004; **16**(1–2):71–101.
10. Di Lucca GA, Fasolino A, De Carlini U, Tramontana P. Abstracting business level UML diagrams from Web applications. *Proceedings 5th IEEE International Workshop on Web Site Evolution*, 2003. IEEE CS Press: Silver Spring MD, 2003; 12–19.
11. Ricca F, Tonella P. Understanding, restructuring Web sites with reWeb. *IEEE Multimedia* 2001; **8**(2):40–51.
12. De Lucia A, Francese R, Scanniello G, Tortora G. Identifying cloned navigational patterns in Web applications. *International Journal of Web Engineering* 2006; **5**(2):150–174.
13. Di Lucca GA, Fasolino AR, De Carlini U, Pace F, Tramontana P. Comprehending Web applications by a clustering based approach. *Proceedings of the 10th International Workshop on Program Comprehension*, 2002. IEEE CS Press: Silver Spring MD, 2002; 261–270.

DOI: 10.1002/smr

14. Girardi C, Pianta E, Ricca F, Tonella P. Restructuring multilingual Web sites. *Proceedings 4th IEEE International Workshop on Web Site Evolution*, Montreal, Canada, 2002. IEEE CS Press: Silver Spring MD, 2002; 290–299.
15. Ricca F, Tonella P, Girardi C, Pianta E. An empirical study on keyword-based Web site clustering. *Proceedings of the 12th International Workshop on Program Comprehension*, 2004. IEEE CS Press: Silver Spring MD, 2004; 204–213.
16. Ricca F, Tonella P. Using clustering to support the migration from static to dynamic Web pages. *Proceedings of the International Workshop on Program Comprehension*, 2003. IEEE CS Press: Silver Spring MD, 2003; 207–216.
17. Duda RO, Hart PE, Stork DG. *Pattern Classification*. Wiley: New York, 2001; 576–581.
18. Lampinen J, Laaksonen J, Oja E. *Neural Network Systems Techniques and Applications in Pattern Recognition*, 1997. http://www.lce.hut.fi/publications/ps/b1_nnsystems.ps [15 April 2007].
19. Di Lucca GA, Di Penta M, Fasolino AR. An approach to identify duplicated Web pages. *Proceedings of the 26th Annual International Computer Software and Application Conference*, 2002. IEEE CS Press: Silver Spring MD, 2002; 481–486.
20. Levenshtein VL. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory* 1966; **10**:707–710.
21. Aversano L, Canfora G, De Lucia A, Gallucci P. Web site reuse: Cloning and adapting. *Proceedings of the 3rd International Workshop on Web Site Evolution*, 2001. IEEE CS Press: Silver Spring MD, 2001; 107–111.
22. Isakowitz T, Kamis A, Koufaris M. Extending the capabilities of RMM: Russian dolls and hypertext. *Proceedings of the 30th Hawaii International Conference on System Science*, 1997. IEEE CS Press: Sliver Spring MD, 1997; 177–186.
23. Calefato F, Lanubile F, Mallardo T. Function clone detection in Web applications: A semiautomated approach. *International Journal of Web Engineering* 2004; **G**(1):3–21.
24. Rajapakse DC, Jarzabek S. An investigation of cloning in Web applications. *Proceedings of the 5th International Conference on Web Engineering*, 2005; 252–262.
25. Wiggerts TA. Using clustering algorithms in legacy systems remodularization. *Proceedings of the 4th Working Conference on Reverse Engineering*, 1997. IEEE CS Press: Silver Spring MD, 1997; 33–43.
26. Flynn PJ, Jain AK, Murty MN. Data clustering: A review. *ACM Computing Surveys* 1999; **31**(3):264–323.
27. Baeza-Yates R, Ribeiro-Neto B. *Modern Information Retrieval*. Addison-Wesley: Reading MA, 1999.
28. Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 1990; **41**(6):391–407.
29. De Lucia A, Scanniello G, Tortora G. Using a competitive clustering algorithm to comprehend Web applications. *Proceedings 8th IEEE International Symposium on Web Site Evolution*, 2006. IEEE CS Press: Silver Spring MD, 2006; 33–40.
30. Kohonen T. Self-organizing formation of topologically correct feature maps. *Biological Cybernetics* 1982; **43**:59–69.
31. Gamma E, Helm R, Johnson R, Vlissides J. *Design Patterns*: *Elements of Reusable Object-Oriented Software*. Addison-Wesley: Menlo Park CA, 1995.
32. Hightower R. *Jakarta Struts Live*. SourceBeat, LLC: Highlands Ranch, CO, 2004.

AUTHORS' BIOGRAPHIES

Andrea De Lucia received the Laurea degree in Computer Science from the University of Salerno, Italy, in 1991, the MSc degree in Computer Science from the University of Durham, U.K., in 1996, and the PhD in Electronic Engineering and Computer Science from the University of Naples 'Federico II', Italy, in 1996. He is a full professor of Software Engineering and the Director of the International Summer School on Software Engineering at the Department of Mathematics and Informatics of the University of Salerno, Italy. Previously he was at the Refsearch Centre on Software Technology (RCOST) of the University of Sanio, Italy. His research interests include software maintenance, reverse engineering, re-engineering, global software engineering, traceability management, configuration management, document management, workflow management, Web engineering, visual languages, and e-learning. Prof. De Lucia has edited international books and special issues of international journals on these topics and has published more than 100 papers in international journals, books, and conference proceedings. He serves on the organizing and program committees of several international conferences in the field of software engineering. He is a member of the IEEE and the IEEE Computer Society.

DOI: 10.1002/smr

Giuseppe Scanniello received the Laurea degree in Computer Science from the University of Salerno, Italy, in 2001, where in 2003 also received the PhD in Computer Science. In 2006, he joined the Department of Mathematics and Computer Science of the University of Basilicata, Potenza (Italy), where he is currently an assistant professor. His research interests include reverse engineering, re-engineering, workflow automation, migration of legacy systems, wrapping, integration, e-learning, cooperative supports for software engineering, and visual languages. Dr Giuseppe Scanniello is a member of the IEEE Computer Society.

Genoveffa Tortora received the Laurea degree in Computer Science from the University of Salerno, Italy, in 1978. Since 1990, she has been a full professor at University of Salerno, Italy, where she teaches database systems and fundamentals of computer science. In 1998, she was a founding member of the Department of Mathematics and Computer Science, acting as chair until October 2000. Since November 2000, she has been the dean of the Faculty of Mathematical, Natural, and Physical Sciences. She is author and coauthor of several papers published in scientific journals, books, and proceedings of refereed conferences, and is coeditor of two books. She is an associate editor and reviewer for international scientific journals. She has been program chair and program committee member in a number of international conferences. Her research interests include software engineering, visual languages, geographical information systems, and pictorial information systems. She is a senior member of the IEEE Computer Society.

DOI: 10.1002/smr