Journal of Web Engineering, , Vol. 5, No.2 (2006) 150-174

IDENTIFYING CLONED NAVIGATIONAL PATTERNS IN WEB APPLICATIONS

ANDREA DE LUCIA, RITA FRANCESE, GIUSEPPE SCANNIELLO, GENOVEFFA TORTORA

Dipartimento di Matematica e Informatica, University of Salerno, Italy
{adelucia, francese, gscanniello, tortora}@unisa.it

Received May 9, 2005
Revised July 25, 2005

Web Applications are subject to continuous and rapid evolution. Often programmers indiscriminately duplicate Web pages without considering systematic development and maintenance methods. This practice creates code clones that make Web Applications hard to maintain and reuse. We present an approach to identify duplicated functionalities in Web Applications through cloned navigational pattern analysis. Cloned patterns can be generalized in a reengineering process, thus to simplify the structure and future maintenance of the Web Applications. The proposed method first identifies pairs of cloned pages by analyzing similarity at structure, content, and scripting code. Two pages are considered clones if their similarity is greater than a given threshold. Cloned pages are then grouped into clusters and the links connecting pages of two clusters are grouped too. An interconnection metric has been defined on the links between two clusters to express the effort required to reengineer them as well as to select the patterns of interest. To further reduce the comprehension effort, we filter out links and nodes of the clustered navigational schema that do not contribute to the identification of cloned navigational patterns. A tool supporting the proposed approach has been developed and validated in a case study.

Key words: Clone Analysis, Cloned Patterns, Navigational Patterns, Web Application Reengineering
Communicated by: B White & L Olsina

1 Introduction

Web Applications (WAs) are playing an increasingly important role in today's society. They represent the engine allowing distributed organizations to communicate, to share information with other companies, customers and providers, and to manage production and distribution. Recently, the industry and academic researchers have been showing great interest in the engineering, maintenance, testing, reverse engineering, restructuring, and reuse of Web sites and applications [1, 8, 10, 13, 15, 16, 21, 30, 31]. Similarly to legacy systems, WAs are subject to continuous evolution [20, 37]: internal and external factors generate new or modified system requirements, so they inevitably changes. Moreover, WAs change much more quickly than traditional software systems; therefore, their maintenance absorbs considerable resources if developers do not use methodologies that anticipate change and evolution [2, 9, 10, 16, 30].

Although researchers of the Web engineering field have proposed several methodologies for the development of WAs, [6, 8, 12, 13, 15, 25, 26, 34], the current state of practice of Web development is far from using them. Thus, similarly to legacy systems, these applications result in very poor quality products, with a "spaghetti-like" structure and no documentation [1, 10, 16, 30]. Even worse, WAs are typically obtained by reusing the fragments of existing pages and without explicit documentation [3, 11, 17, 32]. This approach augments the code complexity and the effort to test, maintain and evolve these applications. It is worth noting that code cloning is not only a problem of WAs. Code cloning is

one of the factors that make software maintenance more difficult [5, 7, 27]. A code clone is a code portion in source files that is identical or similar to another. It is common opinion that code clones make the source files very hard to modify consistently. Clones are introduced for various reasons such as lack of a good design, changing requirements, undisciplined maintenance and evolution, lack of suitable reuse mechanisms, and reusing code by copy-and-paste. Various clone detection methods and tools have been proposed in the literature for traditional software systems, see e.g. [4, 5, 7, 24, 27]. These methods and techniques have been used to support different activities during software maintenance, such as reengineering and quality assessment [4, 24]. For the same reason, detecting code clones is bound to be an important issue to support quality improvement during WA maintenance and evolution.

Recently, researchers have extensively studied clone detection for static and dynamic Web pages [17, 11, 22, 32]. However, besides introducing duplication in Web pages, cloning practices also lead to the production of entirely duplicated navigational patterns in WAs representing duplicated functionalities in the application. This problem is not only common to static Web sites, where duplication of pages and navigational patterns cannot be avoided, but it can be also found in dynamic Web sites, as a result of undisciplined software evolution. Identifying cloned patterns (i.e., isomorphic sub-graphs) [14, 29, 35] is useful for software comprehension and WA reengineering: cloned patterns can be generalized thus simplifying the structure and the future maintenance of the application.

This paper presents a method and a tool to identify cloned navigational patterns in WAs. The method starts with the identification of pairs of cloned pages. Similarly to other approaches [17, 22, 32] the Levenshtein string edit distance [28] is used as basic metric to identify cloned pages. However, besides the distance at structural level, we also consider the distance of cloned pages at the content and scripting code levels. In particular, starting from the Levenshtein edit distance, we define similarity measures for structural, content, and scripting code of two dynamic pages and use thresholds on the similarity measures to establish whether two pages are clones. Cloned pages are then grouped into clusters and the links connecting pages of two clusters are grouped too. We propose an interconnection metric on the links between two clusters to express the effort required to reengineer them as well as to select and display only navigational patterns of interest. A threshold on the interconnection metric is used to filter out irrelevant details, such as nodes and edges corresponding to single pages and links, thus highlighting only cloned navigational patterns of interest. In this way the analysis and understanding of the cloned functionalities is further improved. The identified cloned functionalities can also be used in a reengineering process to simplify the structure and future maintenances of the WAs. The method and the tool have been validated in a case study.

The rest of the paper is organized as follows: Section 2 discusses related work, while Section 3 presents an overview of the approach. Section 4 details the clone analysis method for identifying static and dynamic cloned pages, while Sections 5 presents the approach to cluster cloned pages and links. The prototype tool implementing the proposed method is presented in Section 6, while Section 7 discusses the results of a case study and Section 8 concludes.

2 Related Work

Recently, researchers have extensively studied the problem of defining reverse engineering and analysis techniques for WAs [1, 10, 16, 18, 30]. In these applications, Web pages are usually written in HTML and/or other scripting languages, such as server side scripting languages.

Ricca and Tonella [30] propose ReWeb, a tool for analyzing the structure and the evolution of static Web sites. They 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. They also represent the evolution of a Web site by using colors as time indicators. This is useful to determine how the original structure of a Web site degrades during its life.

To fully understand the behavior of a WA it is necessary to detect the dynamic interactions among its components. Reverse engineering methods and tools have also been proposed by Di Lucca *et al.* [16, 18] that enable to extract the Conallen extension [13] of UML diagrams from existing WAs, by analyzing both static and dynamic contents.

Antoniol *et al*. [2] propose a methodology for reengineering static Web sites. The recovered design is based on the Relationship Management Data Model (RMDM) and the ER+ Model proposed within the Relationship Management Methodology (RMM) [25, 26]. The reverse engineering phase consists of abstracting the navigational structure of the Web site first and then identifying the entities of the application domain and the relationships among them. The recovered ER+ diagram is restructured and the forward engineering phase is performed by following the RMM methodology.

Other authors focus on the migration from static to dynamic Web sites by exploiting methods for the identification of cloned pages [10, 22, 32]. Boldyreff and Kewish [10] propose a reverse engineering approach for static Web sites that enables the extraction of duplicated styles and contents from Web pages. Content and styles are stored into the database [23] and the HTML pages are modified by using scripts that retrieve information from the database and then dynamically generate the page.

Girardi *et al.* [22] present a method for restructuring multilingual Web sites. The authors define a semiautomatic approach to identify and align static HTML pages whose structure is the same but whose content is written in different languages. The similarity of the structure of two static Web pages is based on the Levenshtein string edit distance [28]. The aligned multilingual pages are merged into MLHTML pages. Ricca and Tonella [32] enhance the approach based on Levenshtein distance with hierarchical clustering techniques to identify clusters of duplicated or similar pages to be generalized into a dynamic page. In particular, they propose a semi automatic process aimed at recognizing a common structure of Web pages using an agglomerative hierarchical clustering algorithm. 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 in the hierarchy. However, the method only deals with static pages and do not consider the problem of identifying cloned navigational patterns.

Other methods have been proposed in the literature for the identification of clones in dynamic Web sites. Calefato *et al.* [11] propose an approach for detecting cloned functions within scripting code of WAs for both client and server pages and extend the metric-based approach and the classification schema proposed by Balazinska *et al.* [4]. Their approach exploits a pattern matching algorithm to compare scripting code fragments and is based on two steps: automatic selection of potential function clones based on homonym functions and size measures and visual inspection of selected script functions. This approach does not consider the problem of identifying cloned pages, as it does not take into account neither the structure of Web pages nor their content. The problem of identifying cloned pages from a structural point of view is not considered.

Di Lucca *et al.* [17] present an approach to identify cloned pages in dynamic Web sites written in ASP. Like other approaches [22, 32], the sequence of tags of a Web page is encoded into a string and the Levenshtein edit distance [28] is used to identify cloned pages at structural level. Pages are considered perfect clones if their Levenshtein distance is zero. Our approach for the identification of cloned pages is also based on the Levenshtein distance. However, besides the distance at structural level, we also consider the distance of cloned pages at content and scripting code levels.

Clustering in WAs has also been proposed for the sake of program comprehension [19, 33]. Di Lucca *et al.* [19] propose a clustering method for decomposing WA 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 and simplify the navigational schema to make it more understandable. A different approach is proposed by Ricca *et al.* [33], where clustering is based on the similarity of keyword usage in the page contents. Like previous approaches, our method also simplifies the navigational schema of a WA and makes it more understandable. However, our clustering method is based on the clone relation between two pages and therefore it is more suitable for reengineering WAs by merging cloned pages.

3 Cloned Pattern Analysis Process

In this section we present an overview of our approach to identify cloned navigational patterns in WAs, which is particularly useful for WA reengineering. A cloned navigational pattern consists of groups of cloned pages and cloned links. Previous approaches to WA reengineering only focus on the identification of cloned pages and do not address the problem of cloned links. Suppose, for example, that we have to reengineer a digital library application and consider Figure 1(a). In this figure, the dynamic pages *Conference List* and *Workshop List* can be considered clones, as well as the dynamic pages *Conference Instance* and *Workshop Instance*. As a result, the two cloned patterns in Figure 1(a) can be identified that implement similar functionalities. To generalize these cloned patterns we need to generalize both the two pairs of cloned pages and cloned links. In particular, we have to generalize the entry link of the cloned patterns using a *Type* parameter and have to propagate it across the internal links, as shown in Figure 1(b).

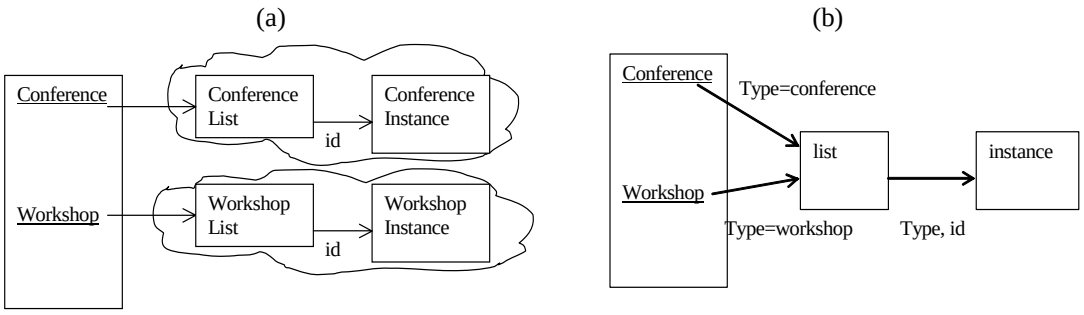


Figure 1 Example of cloned pattern generalization

Figure 2 illustrates the overall cloned pattern analysis process. The rounded rectangles represent process phases, whilst the rectangles represent the intermediate artefacts generated during the process phases. The phases of the process are briefly described in the following subsections. Although our approach is general and can be applied to WAs developed with different server side technologies, we have implemented it for Java Server Pages (JSP), as discussed in Section 6.

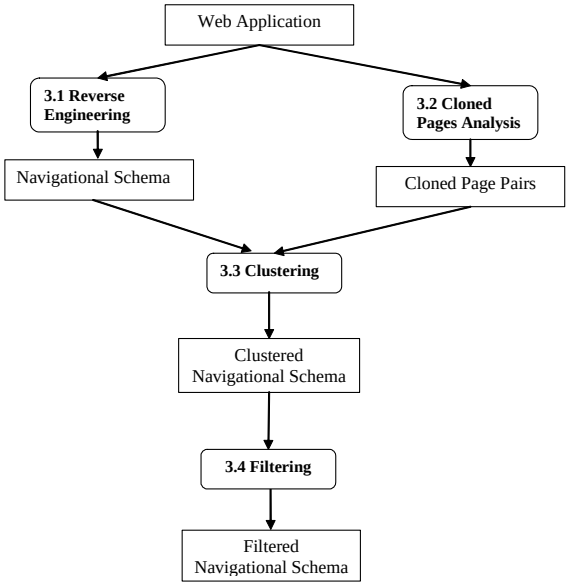


Figure 2 Cloned Pattern Analysis Process

3.1 Reverse Engineering

The reverse engineering phase consists of performing static and dynamic analysis to produce the navigational schema of the WA. The static analysis activity recovers the navigational schema of the given WA in terms of the static relationships between the WA components. Examples of relationships that can be detected are links between pages and submit relationships between a form and the server page that processes the data inserted into the form. The dynamic analysis activity is based on the static analysis results. The latter enables the identification of dynamic pages, but their content can only be determined by executing their code. In the dynamic analysis activity the WA is executed to recover details concerning the dynamic interactions among WA components, thus enriching the navigational schema produced during the static analysis activity. User interaction can improve the results of this phase.

3.2 Cloned Page Analysis

This phase consists of identifying all the possible pairs of clones. Clones are characterized by a similarity degree that ranges from 0%, for disjoint code, up to 100%, for identical copies. In particular, for each pair of HTML pages both the page structure (in terms of HTML tags) and the content is analyzed and compared. For dynamic pages the similarity degree of their scripting code is also considered. Clones are detected according to threshold values on the similarity degrees of Web pages. The choice of the similarity thresholds deeply influences the result of clone analysis process. The thresholds have to be appropriately tuned to work on the subject application. Details of this phase are discussed in Section 4.

3.3 Clustering

The aim of this phase is to group the detected cloned pages into clusters in such a way to be able to identify cloned navigational patterns. Two types of clusters can be identified, based on maximally connected sub-graphs and strongly connected sub-graphs that lead to the identification of different cloned navigational patterns. Once cloned pages are grouped into clusters, the links connecting two page clusters are also grouped. Like cloned pages, groups of links between two clusters (cloned links) will also have to be generalized within a reengineering process. An interconnection metric is computed for each cluster of cloned links to express the effort required to reengineer them. Details of this phase are discussed in Section 5.

3.4 Filtering

In this phase the software engineer can visualize the results of the process and identify parts of the application that implement duplicated or similar functionalities. This phase aimed at pruning the clustered navigational schema of irrelevant details that do not contribute to the comprehension of cloned navigational patterns. Indeed, although clustering greatly simplifies the navigational structure, for complex WAs this is generally not enough to improve the understanding of cloned navigational patterns.

For this reason, we have added a filtering step in the process that enables to highlight only the parts of interests of the navigational schema. In this phase of the process the software engineer interactively filter out uninteresting parts of the clustered navigational schema, i.e., nodes and links that do not contribute to the identification of cloned navigational patterns. In particular, links between single pages and unconnected nodes (corresponding to either single pages or clusters) that do not contribute to the identification of cloned navigational patterns can be filtered out. Moreover, the software engineer can filter out links from a cluster to a single static page, as these will likely not be generalized in a reengineering process. To identify more complex cloned navigational patterns, the software engineer can filter out the links between single pages and clusters, which should have a low reengineering effort. Also, the software engineer can filter out clustered links based on the value of the interconnection metric. Finally, the software engineer can selectively remove nodes and links from the graph.

It is worth noting that the filtering steps can be useful on one side to identify different kinds of cloned navigational patterns and on the other side to estimate the required effort to reengineer them.

4 Identifying cloned pages

In this section we present our approach for the identification of cloned pages in dynamic WAs. In this phase the pages of the WA are compared at three different levels:

Structure – the syntactic structure as marked by the HTML tags.

Content – the text information associated with HTML tags.

Scripting code – server-side scripts.

The first two levels also apply to HTML pages, while the scripting code level is only used for dynamic Web pages.

Figure 3 shows the clone analysis process. In the first phase the pages of the WA are parsed to produce string representations for structure, content, and scripting code (the last for dynamic pages

only). These strings are used to compute the similarity degree of two Web pages at structure, content and scripting code level. The similarity degree is based on the computation of the Levenshtein string edit distance [28] and its comparison with a threshold. The thresholds are dynamically computed based on the size of the two strings and can be separately tuned for the different clone analysis levels. The threshold values help the software engineer to prune the false positives that can be introduced in the page clone analysis. For instance, content level similarity is computed whenever the Web pages are very structured and are based on the same graphical layout that is predominant in the computation of the similarity at structural level. In this case, pages that should not be considered clones are considered clones at structural level, even using high similarity thresholds. In this way, content level similarity computation can prune pairs of cloned pages that are false positives. A similar approach can be used the thresholds of the scripting code. It is worth noting that style sheets that could be used to achieve a coherent page layout of WAs are not considered in the identification of clones at structural level, because they do not add any relevant information.

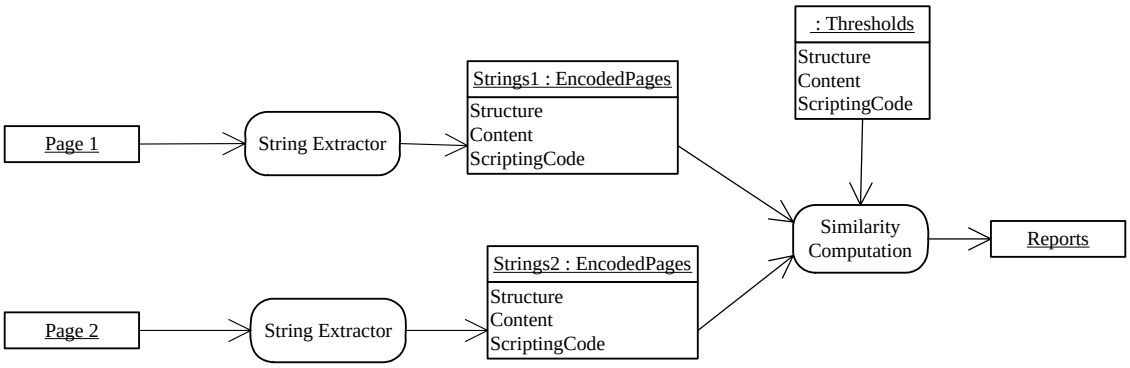


Figure 3 The clone identification process

4.1 Parsing Web pages and producing string representations

The static and dynamic pages are parsed and encoded into strings. In particular, these strings are achieved by means of a depth-first traversal of the abstract syntax tree of the Web page (see Figure 4). Each node of the syntax tree of a Web page is decorated with an HTML tag and with a set of attributes, such as text attribute, java source code, image attribute, etc. The syntax tree of a JSP page differs from the syntax tree of an HTML page only for the JSP node. An example of syntax tree for a JSP page belonging to the WA used as case study is shown in Figure 4. For sake of readability, only the HTML tags are associated to the nodes of the syntax tree. Figure 4 also shows the sequence of tags corresponding to the page structure.

In the current implementation we produce separate string representations for structure, content, and scripting code of a Web page. In this way, the comparison of the different strings at the different levels could also be performed independently of the other levels. Concerning the structural level, the HTML tags of a Web page are encoded into symbols of an alphabet before being concatenated into the string corresponding to the page structure (for sake of simplicity, this mapping is not shown in Figure 4). This enables a more precise computation of the similarity degree of two pages with respect to just concatenating the HTML tags into the strings. For the content level, this means that the text associated with the HTML tags is concatenated into a single string, after deleting meaningless characters, such as multiple spaces, carriage returns and blank lines. More sophisticated analyses could be achieved by

also deleting stop words, similarly to information retrieval techniques. For the scripting code level, we also delete comments.

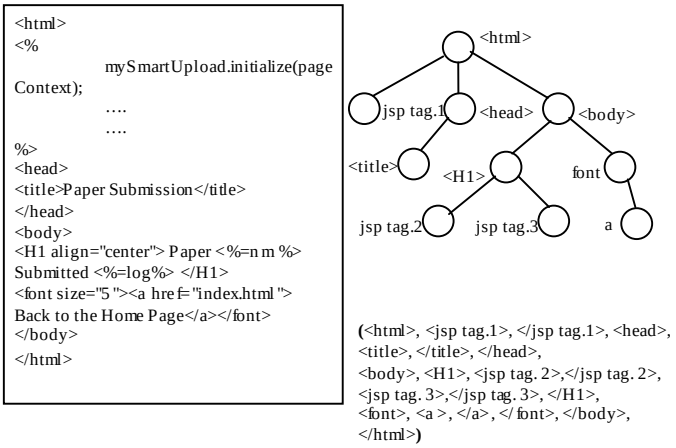


Figure 4 Abstract syntax tree and structure representation for a sample JSP page

An alternative to this approach consists of linking each content and java code string to the corresponding HTML/JSP tag. In this case, the content and java source code fragments of the corresponding tags in the structure could be compared while performing the structure level analysis. A disadvantage of this approach is that we would not be able to identify content and java source code similarity, whenever the similarity degree at the structure level of the two pages is low. For example, two Web pages could have the same java code distributed in different ways across the page structure.

4.2 Computing similarity of two pages

The similarity degree of the strings encoding structure, content and java code of a Web page is based on the Levenshtein edit distance model [28], which is one of most important models for string matching. Due to its flexibility, it is used in several fields, such as code theory, phonetic distance, molecular biology, speech recognition, etc. 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. The Levenshtein distance is defined as the minimum cost required aligning two strings. In particular, given two strings x and y the Levenshtein distance is defined as the minimum number of insert, delete, and replace operations required to transform x into y. Generally, the approach assumes that the insert and delete operations have cost 1, while the replacement has cost 2 (it is equivalent to a sequence of delete and insert operations). We use these costs as default values, although our approach can be parameterized with respect to the costs of the operations.

Whenever the Levenshtein distance is 0, the pair of strings are perfect or identical clones [17]. However, perfect clones in WAs are rare; more interesting is the case of nearly-identical or similar clones [32]. In our approach two Web pages can be considered clones if the distance of the corresponding strings is lower than a given threshold t. This threshold can be dynamically computed from the minimum percentage of similarity p required to consider the two strings as clones and the maximum of the length of the two strings. Therefore, given two strings x and y, let $D(x, y)$ be their distance and $S(x, y)$ be their similarity degree. The two strings are clones if $S(x, y) \geq p$, or equivalently $D(x, y) \leq t(x, y)$, where $t(x, y) = f(p, maxlength(x, y))$.

In the current implementation, the threshold is computed in the following way:

$$t(x, y) = 2\ maxlength(x, y)\ (1 - p)$$

It is worth noting that whenever p is 1 (100% minimum similarity required) the clone analysis will only identify perfect clones. In this case, the clone relation is reflexive (a string is a clone of itself), symmetric (if a string x is a clone of a string y, then y is a clone of x), and transitive (if a string x is a clone of a string y and y is a clone of a string z, then x is a clone of z). In general, whenever the threshold is greater than 0 (percentage of similarity required lower than 100%) the clone relation is not transitive. For example, with reference to Figure 5, string x is a clone of string y and string y is a clone of string z, while string x is not a clone of string z.

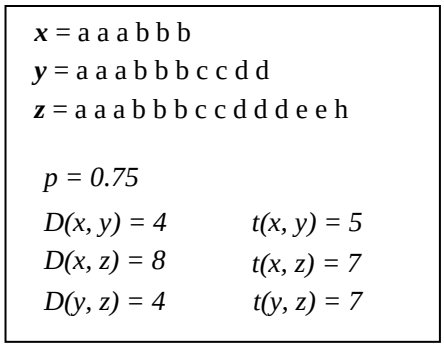


Figure 5 Sample strings, their thresholds and their distances

5 Clustering

Clustering techniques are often used in many areas for a wide range of problems including graph theory, business area analysis, information architecture, galaxy studies, chip design, pattern recognition, economics, statistics, biology, information retrieval, resource allocation, and image processing. In particular, clustering techniques have also been used for software modularization [1, 19, 32, 33, 36]. Clustering can be meant as the grouping of large amounts of things in such a way that the things in one group are closely related compared to the relationships between things in different groups. In cluster analysis such groups are called clusters.

The aim of clustering methods is to extract an existing natural cluster structure. It is worth noting that different methods may come up with different clustering, as well as different clustering requirements may come up with different clustering methods. There are various techniques used to cluster entities. Wiggerts [36] surveys different clustering methods for software modularization and classifies them in four categories, namely graph theoretical algorithms, construction algorithms, optimization algorithms, and hierarchical algorithms.

5.1 Page clustering

We use graph theoretical algorithms to identify clusters of cloned web pages. The clone relation between web pages can be expressed as a graph, where pages are nodes and edges link cloned pages. It is worth noting that the graph is undirected because the symmetric propriety is verified for each pair of cloned pages. In our case studies we used two different clustering techniques based on maximally connected sub-graph and strongly connected sub-graphs, respectively. We call weak clusters (Figure 6(a)) the clusters corresponding to maximally connect sub-graphs and strong clusters (Figure 6(b)) the

clusters corresponding to strongly connected sub-graphs. It is worth noting that the transitive property on the clone relation is not verified among all the pages of a weak cluster, while it is verified among the pages of a strong cluster. Moreover, using perfect clones (100% similarity) weak and strong clusters coincide.

Our approach identifies maximal clusters. In case of weak clusters a transitive closure of the clone relation is applied, while in case of strong clusters we need that the transitive property has to be verified. Unlike in weak clustering, in strong clustering it is possible that the same page belongs to more than one cluster, like in Figure 7. In this case, we assign the page to the larger cluster.

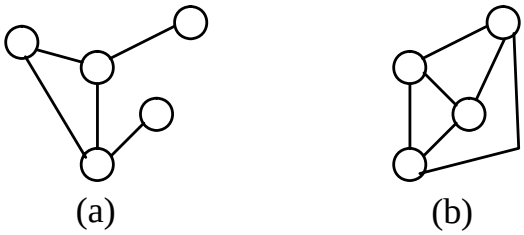


Figure 6 Weak (a) and Strong (b) clusters.

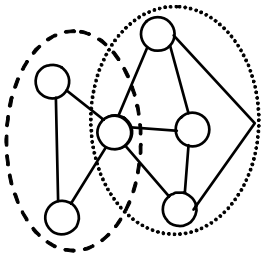


Figure 7 Two clusters with not null intersection

5.2 Link clustering

Once the clusters of cloned pages have been identified, the groups of links between two clusters (or between a cluster of pages and a single node) are clustered too. In this way, the navigational schema of the WA is restructured as shown in Figure 8, thus making the identification and understanding of the cloned navigational patterns easier.

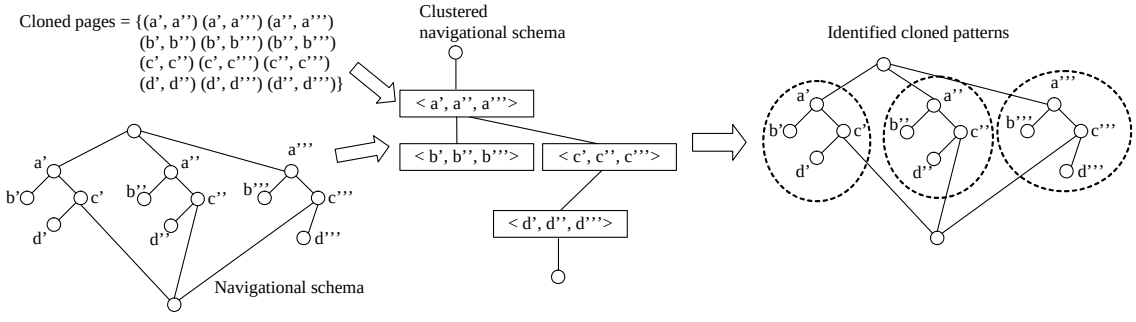


Figure 8 Perfect cloned patterns

Each group of links between two clusters has assigned an interconnection metric value, which takes into account the number of original links between the pages of the clusters. This metric can be considered as an indicator for the effort required to generalize cloned navigational patterns in a reengineering process. In addition, the detection of cloned patterns has also to take into account the lack of links. Figure 9 shows different cases of links between clusters. In particular, the best scenario for a simpler link generalization is shown in Figure 9(a), where for each page in the cluster C_1 there is one link towards only one page in the cluster C_2. Differently, when the number of links between the clusters is $|C_1|x|C_2|$ the link generalization effort should be maximum (see Figure 9(b)). Figure 9(c) shows a case that is more complex than the one depicted in Figure 9(a), from a reengineering point of view, because of the lack of some links. A generic scenario showing two connected clusters of different size is depicted in Figure 9(d).

To express the effort required to generalize the cloned links between two clusters we propose a cluster interconnection metric:

$$\frac{\sum_{i \in C_1} \left| OutgoingLinks_i - 1 \right| + \sum_{j \in C_2} \left| IncomingLinks_j - 1 \right|}{2 \times \left(\left| C_1 \right| \times \left| C_2 \right| \right) - \left(\left| C_1 \right| + \left| C_2 \right| \right)}$$

$OutgoingLinks_i$ represents the number of outgoing links from page i of cluster C_1 towards pages of clusters C_2. Similarly, $IncomingLink_j$ represents the number of incoming links, from pages in cluster C_1, to the page j in the cluster C_2.

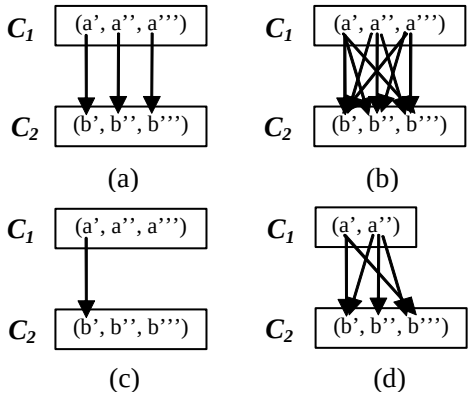


Figure 9 Examples of cloned links.

The value of the metric ranges between zero and one. The zero value represents the best case of cloned links and corresponds to the lowest effort to generalize them, as shown in Figure 9(a). The value of the metric will be 1 (worst case) whenever all possible links between the pages of the two clusters are in the WA (see Figure 9(b) for an example). It is worth noting that the metric also considers the lack of links between the pages of the two clusters, like in Figure 9(c). The value of the metric for the example in Figure 9(c) is 0.33. Indeed, it can be considered an anomalous case if number of links between two clusters is lower than the size of the clusters. Finally, it is worth nothing that this metric is not applied for groups of links connecting a single page to a cluster of pages; in this case the clustered link has assigned the interconnection metric value 0.0, corresponding to a moderate reengineering effort.

6 Implementation

We have implemented a Java prototype to support the software engineer in the detection of cloned pages and patterns. Figure 10 shows the layered architecture of the system that includes a Graphical User Interface, an HTML/JSP parser, a Repository, and an Application Logic layer containing a module for each of the process phases shown in Figure 2. The Repository contains the pages of the WA and all the intermediate representations produced by the parser and the modules of the application logic layer. The HTML/JSP Parser has been implemented by modifying an open source HTML Parser[1] available under GPL license. The parser is used by the cloned page analyzer to encode the static and dynamic pages into strings and by the reverse engineering module to extract the links between pages. The user can select the pages of the WA he/she want to analyze through the GUI (by default all pages of the WA are selected).

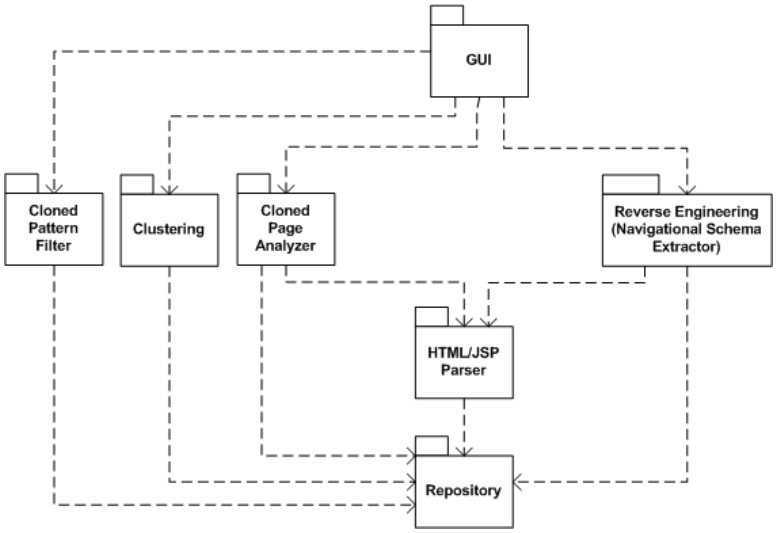


Figure 10 Overall System Architecture

The *Cloned Page Analyzer* detects basic clones on static and dynamic pages. The user can select the similarity thresholds through the GUI (see Figure 11). Basically, the analyzer first uses the HTM/JSP parser to produce the string representation of the Web pages. The Levenshtein algorithm is then used on these strings to compute the similarity degree of two pages at structure, content, and scripting code level (the latter for JSP pages only). The analyzer produces two XML files for the HTML pages and three for the JSP page. These files are stored in the *Repository* and represent the *Cloned Page Analysis* report. In particular, each file contains the list of the analyzed pages, the used threshold, and similarity degree for each pair of clones.

The *Reverse Engineering* module extracts the navigational schema of the WA using the parser and the list of pages selected by the user. It is worth noting that the extracted navigational schema contains also dynamic links (links jointing pages through a FORM element), which are considered useful to identify cloned patterns. The navigational schema can be analyzed through the *Graphical User Interface* (GUI) and the software engineer can edit it to add and delete nodes and links (see Figure 12).

[1] HTMLParser ver. 1.2 http://sourceforge.net/projects/htmlparser

The *Clustering* module groups both the cloned pages and links in a WA as discussed in Section 5. In particular, the *Clustering* module parses the cloned page reports stored in the repository and applies both weak and strong clustering. Once clusters of pages are identified, this module groups cloned links between pairs of page clusters and computes the cluster interconnection metric for each group of links. Figure 13 and 14 show the application of weak and strong clustering to the WA of the *14th International Conference on Software Engineering and Knowledge Engineering*[2] (*SEKE 02*), respectively. The original navigational schema of this WA is shown in Figure 12.

The *Cloned Pattern Filter* module allows filtering cloned patterns out taking into account the type of node and links, and the value of the *interconnection metric*, as discussed in Section 3.4.

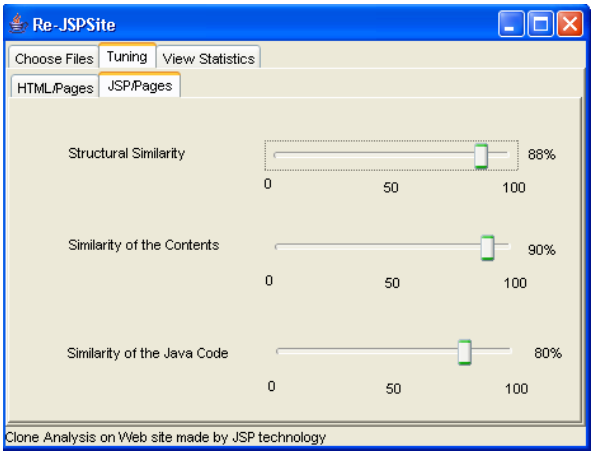


Figure 11 Similarity Thresholds for JSP Pages

Figure 12 SEKE 02 WA original navigational schema

[2] http://www.scienzemfn.unisa.it/seke/

Figure 13 Clustered navigational schema by Weak Clustering

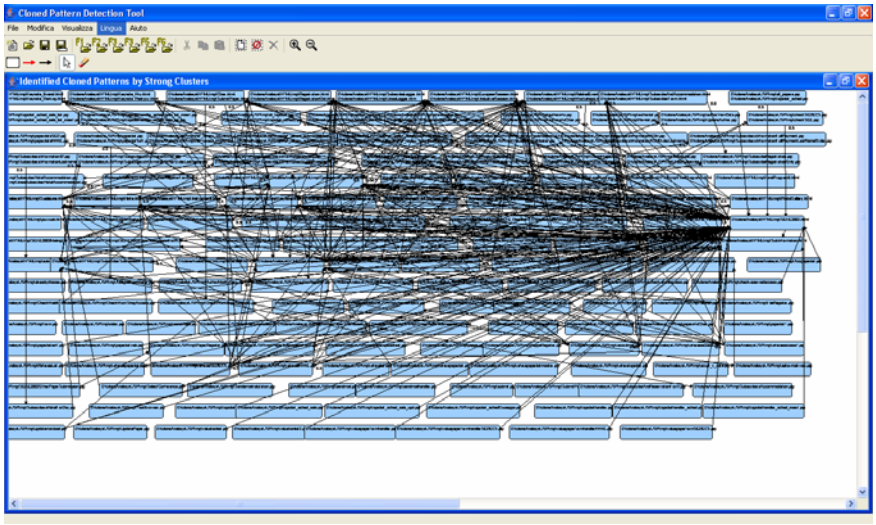


Figure 14 Clustered navigational schema by Strong Clustering

7 Case Study

The method and tool presented in this paper have been validated on three WAs, namely the Web sites of the *International School of Software Engineering*[3] (*ISSE 03*) the *3rd Workshop on Cooperative Support for Distributed Software Engineering Processes*[4] (*CSSE 04*), and the *14th International Conference on Software Engineering and Knowledge Engineering* (*SEKE 02*), all developed in JSP technology. Table 1 shows the descriptive information of these Web sites. The first row contains the number of files composing the application, while the second and third rows contain the number of static and dynamic pages, respectively.

[3] http://www.scienzemfn.unisa.it/seschool/
[4] http://www.scienzemfn.unisa.it/csse/

7.1 Overall Results

For each WA we experimented the approach using different thresholds for structure, content, and scripting code level during the cloned page analysis phase. The results obtained from the cloned page analysis have been evaluated using precision and recall, two well known metrics of the information retrieval and reverse engineering fields. In our case, the precision is the number of relevant pairs of cloned pages identified by the tool over the total number of identified pairs. The recall is the ratio between the numbers of relevant pairs of cloned pages identified by the tool over the total number of relevant pairs of cloned pages in the WA.

	ISSE03	***CSSE04***	***SEKE02***
Number of Component Files	66 files in 14 folders	56 files in 17 folders	1268 files in 63 folders
Number of HTML Pages	12	10	52
Number of JSP Pages	7	21	107

Table 1 Analyzed WAs

For each set of thresholds we performed weak and strong clustering and analyzed the resulting cloned patterns using the filtering functionality of the tool. We started with 80% as similarity threshold for structure, content, and scripting code and tuned these thresholds until we achieved 100% of recall, with respect to the results provided by the software engineers who developed the application.

Table 2 shows the results achieved with the initial thresholds and the best results achieved in terms of precision with a recall of 100%. The first two rows contain the thresholds used at structure, content, and scripting code similarity levels for the page clone analysis, respectively. The third row contains the numbers of identified pairs of static cloned pages achieved with the adopted thresholds, while the numbers of pairs of dynamic cloned pages is contained in the fourth row. Finally, the fifth and sixth rows contain the number of weak and strong clusters, respectively.

The first WA that we present as case study is the ISSE03 Web site. The page clone analysis phase produced 6 pairs of cloned static pages (100% of recall and 50% of precision) and one pair of cloned dynamic pages (100% recall and precision), using the initial thresholds. The best results for the static pages were achieved at the third iteration using as thresholds 87% for structure and 86% for content. In this way, one of the three false positives included with the initial thresholds was eliminated (60% precision) while keeping 100% recall. The initial thresholds used for JSP pages resulted in 100% of both precision and recall. The similarity at scripting code level of most pairs of dynamic pages was less than 55.8% and this explain why they were not selected using the initial thresholds. Differently from the static pages, the best results of precisions and recall were achieved at the first iteration. The only pair of cloned pages selected by the tool can be considered a pair of nearly identical clones. Indeed, the similarity of these pages is 100% at structure level, 99.8% at content level, and 100% at scripting code level. These pages were used by the scientific committee members and the registered students, respectively, to access the reserved area of the WA. Weak and strong clustering produced different results also in terms of the identified cloned patterns. In both cases one of the clusters was represented by the pair of JSP pages. Moreover, weak clustering grouped all HTML cloned pages into one cluster, thus correctly restructuring the navigational schema, while strong clustering produced two

clusters of HTML pages, which resulted in incorrect restructuring. It is worth noting that all the generalizable links identified by the tool had 0.0 as interconnection metric value. This was due to the fact that the two identified weak clusters were linked to single nodes in the clustered navigational schema.

	ISSE03		***CSSE04***		***SEKE02***	
	Initial Results	**Best Results**	**Initial Results**	**Best Results**	**Initial Results**	**Best Results**
HTML Pages Thresholds	80% structure 80% content	87% structure 86% content	80% structure 80% content	80% structure 65% content	80% structure 80% content	97% structure 80% content
JSP Pages Thresholds	80% structure 80% content 80% scripting code	80% structure 80% content 80% scripting code	80% structure 80% content 80% scripting code	99% structure 100% content 88% scripting code	80% structure 80% content 80% scripting code	70% structure 70% content 70% scripting code
Number of static clone pairs	6	5	1	8	26	10
Number of dynamic clone pairs	1	1	2	0	73	127
Number of Weak Clusters	2	2	3	3	23	29
Number of Strong Clusters	3	3	3	4	30	42

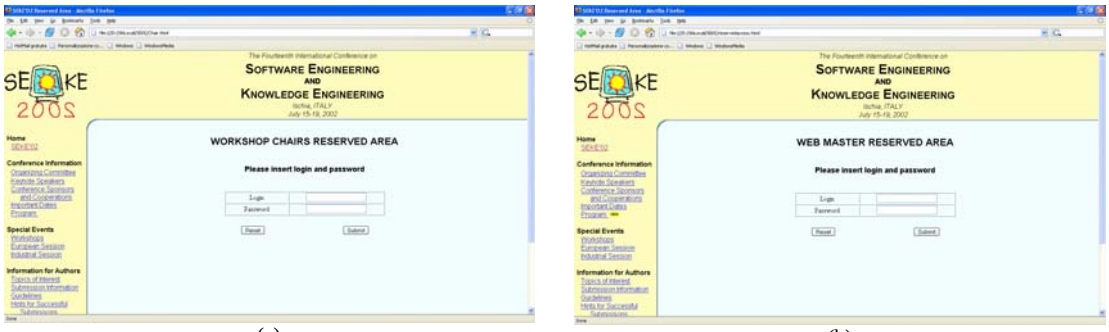
Table 2 Results achieved with the initial thresholds and the best results

The second WA used as case study is the CSSE04 Web site. This application has been used both to provide information about the workshop and for paper submission and refereeing. The manual analysis of this application revealed only seven pairs of cloned static pages and no cloned dynamic pages. The initial threshold values resulted in 100% of precision and a low recall value. For this reason we decided to try with a lower value for the content level threshold. The best results (100% of precision and recall) were achieved using as thresholds 80% at structure level and 65% at content level, respectively. We achieved these results in two iterations. The highest similarity values for pairs of JSP pages were 98.4% at structure level, 100% at content level, and 87% at scripting code level. Therefore, to avoid the inclusion of false positives we had to use higher threshold values, as shown in Table 2. In spite of the low number of dynamic pages, six iterations were required to tune the thresholds.

As in the case of ISSE03, weak clustering produced better results than strong clustering. The three weak clusters belong to different navigational patterns. The first cluster included two versions of the CSSE 04 home page, which slightly differ in the structure and content: in particular, the early version also includes the topics of the workshop and the guideline for the paper submission. These pages contained the same outgoing and incoming links that suggested us to restructure them into a single dynamic page, whose behavior depends on the date. The second cluster involved the pages used to notify error or warning messages to the user, e.g. unauthorized access, wrong login or password, and so on. These pages can also be restructured in one dynamic page, as they only differ for the displayed

message. The third cluster includes two pages, i.e. the page containing the list of accepted papers and the schedule of the workshop program.

The third application used as case study is the SEKE02 Web site. This WA has been used to provide main information about the venue as well as for conference registration and for paper submission and refereeing. The functionalities of the WA were used for the main conference and for two joint workshops on *Web Engineering* and *Software Engineering Decision Support*, respectively. The overall size of the application was 4.14 MB. Besides the 159 Web pages, the Web site also contained many other files, such as images, Java applets, and Java classes, as well as copies of the analyzed Web pages. This is the most meaningful case study for the size and the complexity of the application. The original navigational schema of the SEKE02 WA is shown in Figure 12. The initial threshold values for the static pages produced many false positives (100% of recall and low precision). The best results (100% of recall and 100% of precision) were achieved using 97% and 80% as minimal structural and content similarity thresholds, respectively. The pairs of relevant cloned pages were 10. To achieve these results, we tried eight pairs of threshold values. It is worth noting that a larger number of attempts were needed, differently from the case studies proposed so far. This was due to the large number of static pages as well as the low quality and high complexity of the considered WA. An example of cloned static pages with 97.3% similarity at structure level and 94.6% similarity at content level is shown in Figure 15.



(a) (b)

Figure 15 A pair of nearly identical HTML pages

Concerning the dynamic pages the initial thresholds did not result in 100% of recall, (the number of relevant pairs of clones are 94) so we tried with lower thresholds. The best value of recall (100%) was achieved using 70% for all similarity thresholds, with a precision of 74%. In this case, the pairs of identified cloned pages were 127 and the best results were achieved at the fourth iteration.

The weak and strong clustering methods gave different results. Again, weak clustering gave better results than strong clustering. The reason was that strong clustering produced very small clusters, most of them composed of only a pair of pages, thus splitting groups of pages identified by weak clustering that had indeed to be reengineered into a single dynamic page. However, also the results of weak clustering had to be analyzed and revised, because of the presence of false positives. In particular, most of the clusters were identified correctly, but we noticed the presence of few clusters that were either false positives or included too many pages and needed to be split into two different clusters. This was due to the false positives cloned pairs of pages identified during the page clone analysis. The analysis of the clusters helped us to identify and remove the false positive pairs of clones. The false positive clusters were just composed of two pages, thus immediately suggesting the pair of clones to

remove. In the case of larger clusters, we analyzed the pairs of pages involved in the cluster and identified and removed the pairs of false positives clones. After removing all the false positive pairs from the cloned page analysis report (thus having all and only the actual pairs of clones), we performed again weak and strong clustering, to achieve the results in Table 3(a) and 3(b), respectively.

Cluster 1	all_papers.jsp; topiclist_sched.jsp; topiclist_sched_sala_list.jsp; topiclist_sched_sala_list_insert.jsp; topiclisthandler_sched_sala.jsp;
Cluster 2	all_topiclist.jsp; topiclist.jsp; topiclist_sched2.jsp; topiclist_sched_sala.jsp;
Cluster 3	assign_conf.jsp; AssignValuation.jsp; refereassign.jsp; topiclisthandler_sched_sala_list.jsp;
Cluster 4	authorinfo.jsp; authorinfoChair.jsp; PaperRegister.jsp;
Cluster 5	authorlist.jsp; Rewlist.jsp;
Cluster 6	AuthorViewData.jsp; UpdateAuthor.jsp
Cluster 7	checkSEDECS.jsp; checkSEE.jsp; checkWWE.jsp;
Cluster 8	paperdataSED.jsp; paperdataWWE.jsp;
Cluster 9	paperlist.jsp; paperlistSoft.jsp; paperlistWeb.jsp;
Cluster 10	Ref_review_SEDECS.jsp; Ref_review_WWE.jsp;
Cluster 11	Ref_review_submission.jsp; Ref_review_submissionWWE.jsp;
Cluster 12	Ref_review_submissionSEDECS.jsp; SecondRef_review_submissionWWE.jsp;
Cluster 13	Ref_review_submissionSEKE.jsp; ValuationViewMaster.jsp;
Cluster 14	SubscribersInformationACM.jsp; SubscribersInformationACMOrderLastname.jsp; SubscribersInformationFullPayment.jsp; SubscribersInformationFullPaymentLastNameOrder.jsp; SubscribersInformationID.jsp; SubscribersInformationLastname.jsp; SubscribersInformationOther.jsp; SubscribersInformationOtherOrderLastname.jsp; SubscribersInformationStudent.jsp; SubscribersInformationStudentLastName.jsp;
Cluster 15	topichandler.jsp; valuepaperform.jsp; valuepaperformSEDECS.jsp;
Cluster 16	valuepaper.jsp; valuepaperformWWE.jsp;
Cluster 17	AuthorAccess.jsp; CheckWebM.jsp; CheckWorkShopAccessB.jsp;
Cluster 18	Subscribe.jsp; UnisaSubscribe.jsp
Cluster 19	SubscribeRegistration.jsp; SubscribeRegistrationOnly.jsp; SubscriberHotelAccommodation.jsp; UnisaSubscriberHotelAccommodation.jsp
Cluster 20	Keynote_Briand.html; Keynote_Dantzig.html Keynote_Pedrycz.html; Keynote_Wu.html
Cluster 21	Chair.html; SubmitPaper.html; reservedaccess.html
Cluster 22	Registration.html; RegistrationOld.html
Cluster 23	SubscribeLogger.html; UnisaLogger.html
Cluster 24	EuropeanSession.html, IndustrialSession.html RelatedEvents.html; Workshops.html, Keynote.html
Cluster 25	SubscribeAndAccommodation.html; SubscriberForm.html
Cluster 26	MailingList.html; SEKE2002MailingList.html

(a)

Cluster 1	topiclist_sched.jsp; all_papers.jsp
Cluster 2	topiclist_sched_sala_list.jsp; topiclist_sched_sala_list_insert.jsp
Cluster 3	topiclisthandler_sched_sala_list.jsp; topiclisthandler_sched_sala_list_insert.jsp
Cluster 4	all_topiclist.jsp; topiclist_sched2.jsp
Cluster 5	AssignValutation.jsp; assign_conf.jsp
Cluster 6	refereassign.jsp; topiclisthandler_sched_sala.jsp
Cluster 7	authorinfoChair.jsp; authorinfo.jsp
Cluster 8	authorlist.jsp; Rewlist.jsp
Cluster 9	UpdateAuthor.jsp; authorViewData.jsp
Cluster 10	checkSEDECS.jsp; checkSEE.jsp
Cluster 11	paperdataWWE.jsp; paperdataSED.jsp
Cluster 12	paperlist.jsp; paperlistSoft.jsp
Cluster 13	Ref_review_WWE.jsp; Ref_review_SEDECS.jsp
Cluster 14	Ref_review_submission.jsp; Ref_review_submissionWWE.jsp
Cluster 15	SecondRef_review_submissionWWE.jsp; Ref_review_submissionSEDECS.jsp
Cluster 16	Ref_review_submissionSEKE.jsp; ValuationViewMaster.jsp
Cluster 17	SubscribersInformationACMOrderLastname.jsp; SubscribersInformationACM.jsp
Cluster 18	SubscribersInformationFullPayment.jsp; SubscribersInformationFullPaymentLastNameOrder.jsp
Cluster 19	SubscribersInformationLastname.jsp; SubscribersInformationID.jsp
Cluster 20	SubscribersInformationOther.jsp; SubscribersInformationOtherOrderLastname.jsp
Cluster 21	SubscribersInformationStudentLastName.jsp; SubscribersInformationStudent.jsp
Cluster 22	topichandler.jsp; valuepaperform.jsp
Cluster 23	valuepaperformWWE.jsp; valuepaper.jsp
Cluster 24	AuthorAccess.jsp; ChechWebM.jsp
Cluster 25	Subscribe.jsp; UnisaSubscribe.jsp
Cluster 26	SubscribeRegistration.jsp; SubscribeRegistrationOnly.jsp
Cluster 27	UnisaSubscriberHotelAccommodation.jsp; SubscriberHotelAccommodation.jsp
Cluster 28	Keynote_Briand.html; Keynote_Dantzig.html
Cluster 29	Keynote_Pedrycz.html; Keynote_Wu.html
Cluster 30	Chair.html; SubmitPaper.html
Cluster 31	Registration.html; RegistrationOld.html
Cluster 32	UnisaLogger.htm; SubscribeLogger.html
Cluster 33	EuropeanSession.html; IndustrialSession.html
Cluster 34	Workshops.html; RelatedEvents.html
Cluster 35	SubscribeAndAccommodation.html; SubscriberForm.html
Cluster 36	MailingList.html; SEKE2002MailingList.html

(b)

Table 3 Weak (a) and Strong (b) Clusters

As expected, the number of strong clusters is larger than the number of weak clusters. While weak clustering correctly identified the clusters, strong clustering again did not produce good results, thus

confirming our previous findings. It is worth noting that all strong clusters have size two, thus meaning that all weak clusters with size greater than two are not strong clusters (in other words, the transitive property on the clone relation is not verified in these clusters). The result of strong clustering for the SEKE case study is a loss of clone information. For example the page *valuepaperformSEDECS.jsp* in the cluster 15 in Table 3(a) does not belong at any strong cluster in Table 3(b). Figure 13 shows the clustered navigational schema obtained by weak clustering. It contains 108 nodes (26 nodes represent clusters of pages), while the number of links is 406. Of these, only 122 were actually groups of links that can be generalized, while the remaining are single links between nodes representing single pages.

7.2 Patterns Analysis in SEKE02 Web Application

In this section we show examples of navigational patterns discovered in the SEKE 02 WA. We used the results of the weak clustering method shown in Table 3(a) because the results of strong clustering were not accurate. The clustered navigational schema of the SEKE 02 WA was simplified using the filtering functionality of the tool in different steps.

The first filtering operation consisted of pruning the links between two pages (non clustered links) and the resulting unconnected nodes corresponding to single static or dynamic pages. Indeed, this part of the navigational schema would not be involved in the identification, understanding, and reengineering of cloned patterns. In a second step we also filtered out the clustered links having a single static page as target node. The reason was that such a link stands for a group of static links that likely will not be generalized in a reengineering process. Again, the resulting unconnected nodes corresponding to single pages were filtered out too.

As a result of these two filtering steps we obtained the navigational schema in Figure 16, which contains five isolated sub-graphs representing cloned navigational patterns of the WA, as well as the unconnected nodes corresponding to clusters of cloned pages. Four of the isolated sub-graphs (shown in the left-hand side of the screen-shot) were composed of only two nodes, one corresponding to a single node and the other one to a cluster of cloned pages. The pages involved in these sub-graphs were concerned with duplicated functionalities about the visualization and update of author and session chair information (clusters 4 and 6), paper review assignment that were duplicated in the main conference and the two joint workshops (cluster 9), and computation of statistics about the conference attendees (cluster 14). This cloned pattern is composed of a single HTML page, which is linked to all ten JSP pages of the cluster 14. As defined in Section 5, the cluster *interconnection metric* value is 0. It is worth noting that 12 isolated clusters of pages were also identified, which might be easily generalized in a reengineering process.

The larger sub-graph contained 38 nodes, 10 of which were clusters of cloned pages. The pages of these clusters implemented duplicated functionalities about the conference registration and hotel reservation, paper submission, and visualization of keynote speeches and conference sessions. Most of the nodes corresponding to single pages in the larger sub-graph were linked to three clusters, namely, cluster 22, cluster 23, and cluster 24 (nodes on the right-hand side of the graph).

In the third step we filtered out all the nodes corresponding to single pages and all the unconnected nodes. The goal was to highlight only the more complex cloned navigational patterns to generalize in the reengineering process. As a result, all the smaller sub-graphs were filtered out (one of the two nodes in all these sub-graphs was a single page). Concerning the larger sub-graph, all nodes were filtered out, except the nine clusters enclosed in the dashed line in Figure 16. Only one cluster (cluster 7 in Table 3(a)) of this sub-graph was filtered out, as it was only connected to single pages.

The resulting clustered navigational schema is shown in Figure 17. It contains 14 generalizable links with values of the interconnection metric between 0.0 and 0.7. It is worth noting that to better comprehend this clustered navigational schema the nodes have been labeled with the corresponding cluster in Table 3(a).

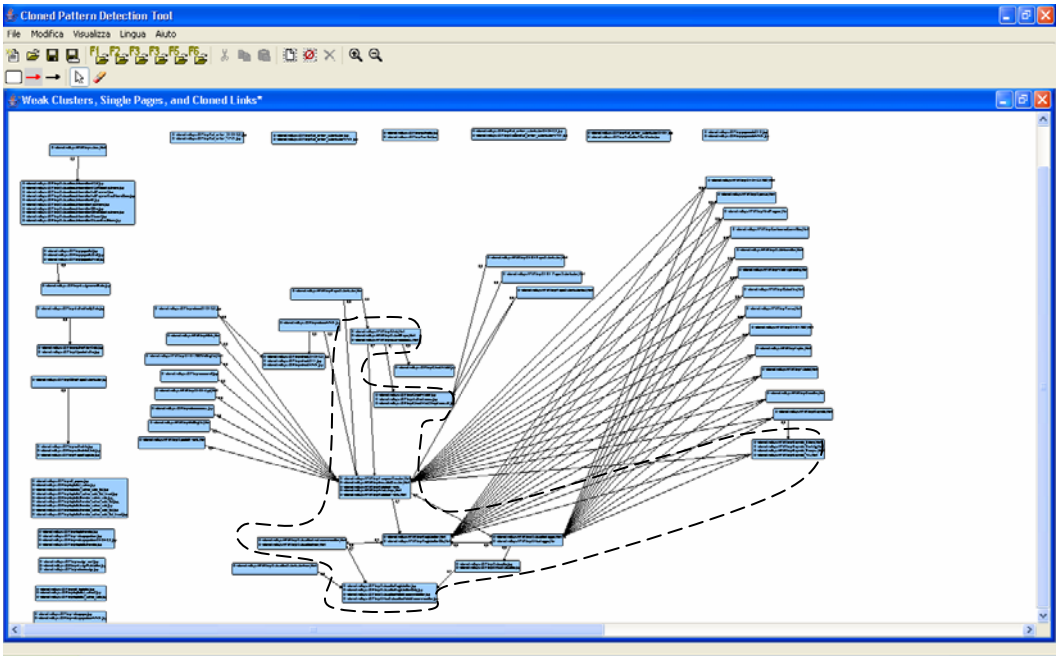


Figure 16 Filtered navigational schema

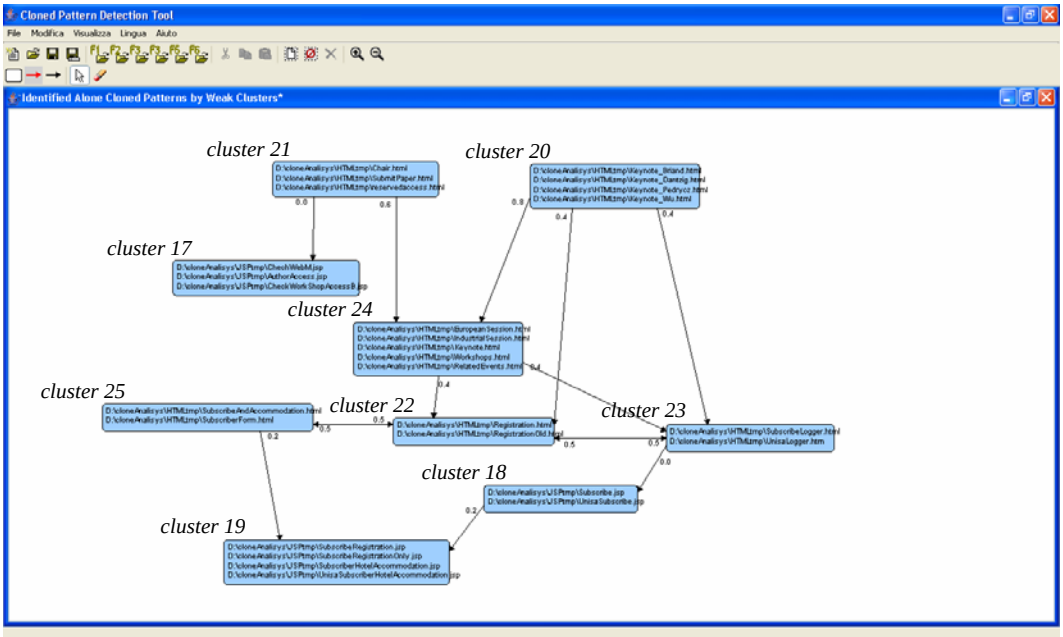


Figure 17 Main clustered navigational schema

Figure 18 shows the sub-graph of SEKE 02 navigational schema used to identify the clustered navigational schema above. This clustered navigational schema has been identified using the pages of the clusters 17, 18, 19, 20, 21, 22, 23, and 25 of Table 3(a), and the links jointing these pages.

The cluster 21 contains static pages for the login of the users of SEKE 02 conference and its workshops. Each page of this group is linked to a dynamic page of the cluster 17 in Table 3(a), namely *AuthorAccess.jsp, CheckWorkshopAccessB.jsp*, and *CheckWebM.jsp*. The page *AuthorAccess.jsp* was used by the SEKE 02 users to see and modify personal data as well as to view the submitted paper, while the latter two pages were used to manage the activities common to the conference and workshops. In particular, the conference committee used *CheckWebM.jsp*, while the page *CheckWorkshopAccessB.jsp* was used by the workshop organizers.

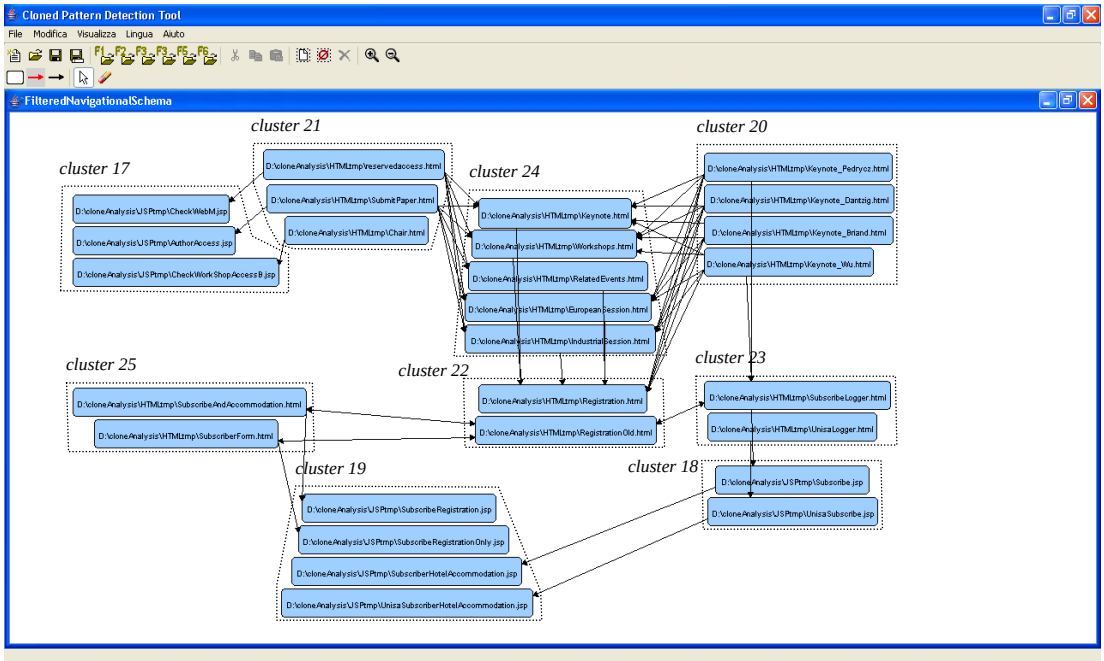


Figure 18 Sub-graph of the navigational schema used to identify the main clustered navigational schema

The pages of cluster 22 are *RegistrationOld.html* and *Registration.html*. The WA developers used these pages to publish information on the conference and workshops registration. Thus, the contents of both the pages are very similar. In particular, *RegistrationOld.html* was connected to pages for the conference registration and hotel reservation. The pages implementing these features were available until one week before the beginning of the conference; afterward *Registration.html* was the published page. These pages represent an example of evolution of the services provided by a WA conceived for academic conferences [37]. Unfortunately, we did not find other examples, as the developers did not systematically maintain the different versions of the Web pages. These two pages in the reengineering phase could be turned into a single dynamic page, whose behaviour depends on the date. The pages of the cluster 23 are *UnisaLogger.html* and *SubscribeLogger.html*. These pages were used to login SEKE 02 staff and pre-registered conference attendees, respectively. In particular, the former page was linked to *UnisaSubscribe.jsp*, while the latter to *Subscribe.jsp*. These pages were in the cluster 18 and were aimed at identifying the users in the system database. For the hotel reservation the staff and conference attendees used *UnisaSubscriberHotelAccommodation.jsp* and *SubscriberHotelAccommodation.jsp*

(cluster 19), respectively. It is worth noting that the SEKE 02 WA developers implemented two different pages for the staff and conference attendees because the type of reservation was different. The cluster 19 also contained the pages *SubscribeRegistration.jsp* and *SubscribeRegistrationOnly.jsp*. *SubscribeAndAccommodation.html* and *SubscribeRegistration.jsp* allowed the conference registration and the hotel reservation, while for the conference registration only the user could use the pages *SubscriberForm.html* and *SubscribeRegistrationOnly.jsp*. The pages *SubscribeAndAccommodation.html* and *SubscriberForm.html* are both in the cluster 25.

8 Conclusion and Discussion

In this paper we have presented an approach based on clone analysis to understand duplicated functionalities in WAs as well as a prototype to support the software engineer in the detection of cloned pages and patterns. The approach first detects cloned pages by analyzing their similarity from structural, content and scripting points of view. A threshold is used to decide whether two pages have to be considered clones. Cloned pages are then grouped into clusters by using two different clustering techniques, namely weak and strong clustering. Cloned patterns are identified by grouping links connecting clusters. Filtering functionalities are defined to improve the readability of the clustered navigational graph and identify relevant cloned navigational patterns. An interconnection metric is also used to filter out cloned links in cloned patterns. This metric can also be considered as an indicator for the effort required to generalize cloned navigational patterns in a reengineering process. Our method has been applied to WAs based on JSP technology. However, it can be also used on WAs developed using other server side technologies.

Both the approach and the tool have been assessed in three case studies. The most meaningful case study was the SEKE 02 WA, which allowed us to perform several considerations. To identify cloned pages for both the static and dynamic pages the best result of precision and recall were obtained with high threshold values because of the analyzed pages had almost the same graphical layout. Indeed, in our case this was due to the fact that all pages have a column including the same navigational menu on the left side and the conference name and logo on the top (see the two sample pages in Figure 15). Moreover, the tool identified clones representing different releases of the same page as shown in Tables 3(a) and (b). In some cases they represent the evolution of the functionality provided by a page, at different stage of the conference organization, as for example *Registration.html* and *RegistrationOld.html*. In other cases, they are just improvements of previous pages or perfectly duplicated pages both from a structural, content, and scripting code (for JSP pages) point of view. In the first case the cloned pages were involved in the cloned pattern identification, while in the latter they were not considered.

Case studies also revealed that when using a high threshold in the basic clone analysis, weak clustering produces better results than strong clustering. This is due to the fact that high similarity threshold in the basic clone analysis results in small weak clusters that cannot be further decomposed in meaningful strong clusters. It is possible that whenever a lower threshold is required in the basic clone analysis, larger weak clusters are achieved that can be further refined using strong clustering. We also noticed that most groups of cloned links between clusters have an interconnection metric value higher than 0.5, and then most of the cloned patterns are not perfect cloned patterns. This means that the WA does not include perfect cloned patterns. This is mainly due to the fact that we considered a dynamic web site, while it is likely that static web sites include many perfect cloned patterns. In general, we observed that the values of the interconnection metric of the clustered links can be used to make a preliminary rough understanding of the type of links between two clusters. For example, a link

between two clusters each composed of two pages can have only the interconnection metric values 0.0, 0.5, and 1.0. The value 0.0 can be achieved in case of perfect cloned patterns, while the value 1.0 was achieved in case of maximally connected cloned patterns. In all the other cases the value of the interconnection metric is 0.5. Similar considerations can be made for clusters of different size.

Although the approach and the system prototype fit their purpose, the time required to identify basic clones in some circumstances may be considerable. This is mainly due to the time required to compare web pages using the Levenshtein string edit distance algorithm [28]. In particular, the execution of the cloned page analysis on the SEKE 02 WA that resulted in the best values of precision and recall took 38 minutes[5]. On the other hand, less than two minutes were employed by the system to identify cloned pages in the ISSE 03 and the CSSE 03 WAs. Such a difference is mainly due to the larger size of the SEKE 02 WA in terms of both number and length of web pages. However, the Cloned Page Analysis phase does not require the interaction of the software engineer and can be carried out in background. Conversely, the identification of the cloned navigational patterns is interactive, and the time to carry out this activity depends on the software engineer background and his/her tool expertise. The software engineer has to analyze the results of the system and if these are not acceptable, he/she can tune the thresholds to achieve better results. Also, cloned navigational patterns are better identified by the software engineer when he/she is able to suitably use the filtering operations provided by the system. In general, the process of reengineering existing systems cannot be completely automated. The interaction of the software engineer is required to achieve good results. However, the effort spent for reengineering a software application in general and a WA in particular is largely compensated by the effort saved during the maintenance of the application.

Future work will be devoted to further experiment our tool in larger case studies to better validate the proposed approach. Furthermore, we are going to extend our approach to systems based on servlets and on other server-side scripting code. Moreover, we plan to investigate other metrics to assess the effort for reengineering WAs.

References

1. Anquetil N. and Lethbridge T. C. Experiments with clustering as a software remodularization method. In Proc. of the 6th Working Conference on Reverse Engineering, Atlanta, Georgia, USA, October1999. IEEE Computer Society, pp. 235–255.
2. Antoniol G., Canfora G., Casazza G., and De Lucia A. Web Site Reengineering using RMM. In Proc. of International Workshop on Web Site Evolution, Zurich, Switzerland, 2000, pp. 9-16.
3. Aversano L., Canfora G., De Lucia A., and Gallucci P. Web Site Reuse: Cloning and Adapting. In Proc. of 3rd IEEE International Workshop on Web Site Evolution, Florence, Italy IEEE CS Press, 2001, pp.107-111.
4. Balazinska M., Merlo E., Dangenais M., Lague B., and Kontogiannis K. Measuring Clone Based Reengineering Opportunities. In Proc. of the 6th IEEE International Symposium on Software Metrics, 1999, Boca Raton, Florida, IEEE CS Press, pp.292-303.
5. Baker B. S. On Finding Duplication and Near Duplication in Large Software Systems. In Proc. of the 2nd IEEE Working Conference on Reverse Engineering, Toronto, Canada, IEEE CS Press, 1995, pp 86-95.
6. Baresi L., Garzotto F., and Paolini P. Extending UML for Modeling Web Applications. In Proc. of 34th Annual Hawaii International Conference on System Sciences (HICSS-34), IEEE CS Press, 2001, pp. 1-10.

[5] A Laptop equipped with a 1,4 GHz Intel Pentium M processor and 512 MB of RAM Memory was used

7. Baxter D., Yahin A., Moura L., Sant'Anna M., and Bier L. Clone Detection Using Abstract Syntax Trees. In Proc. of IEEE International Conference on Software Maintenance, Bethesda, Maryland, USA, IEEE CS Press, 1998, pp. 368-377.
8. Bieber M. and Isakowitz T. (guest editors), Special issue on Designing Hypermedia Applications, Communications of the ACM, vol. 38, no. 8, 1995.
9. Boldyreff C., Munro M., and Warren P. The evolution of websites. In Proc. of 7th IEEE International Workshop on Program Comprehension, Pittsburgh, Pennsylvania, USA, IEEE CS Press, 1999, pp. 178–185.
10. Boldyreff C. and Kewish R. Reverse Engineering to Achieve Maintainable WWW Sites. In Proc. of 8th IEEE Working Conference on Reverse Engineering, Suttgart, Germany, IEEE CS Press, 2001, pp. 249 – 257.
11. Calefato F., Lanubile F., and Mallardo T. Function Clone Detection in Web Applications: A Semiautomated Approach. In International Journal of Web Engineering, vol.3, no.1, May 2004, pp. 3-21.
12. Ceri S., Fraternali P., Bongio A. Web Modeling Language (WebML): a modeling language for designing Web sites. In Computer Networks, 9th World Wide Web Conference, vol. 33, 2000, pp. 137 – 157.
13. Conallen J. Building Web application with UML. Addison Wesley, 2000.
14. Cormen T. H., Leiserson C. E., and Rivest R. L. Introductions to Algorithms, MIT Press, 1990.
15. Costagliola G., Ferrucci F., and Francese R. Web Engineering: Models and Methodologies for the Design of Hypermedia Applications. In Handbook of Software Engineering and Knowledge Engineering, S.K. Chang (editor), World Scientific Publishing Co., pp. 181- 199.
16. Di Lucca G. A., Fasolino A. R., and Tramontana P. Reverse engineering Web applications: the WARE approach. In Journal of Software Maintenance and Evolution: Research and Practice, vol. 16, no. 1-2, 2004, pp. 71-101.
17. Di Lucca G. A., Di Penta M., and Fasolino A. R. An Approach to Identify Duplicated Web Pages. In Proc. of 26th IEEE Annual International Computer Software and Application Conference, Oxford, UK, IEEE CS Press, 2002, pp. 481-486.
18. Di Lucca G. A., Fasolino A.., De Carlini U., and Tramontana P. Abstracting Business Level UML Diagrams from Web Applications. In Proc. of 5th IEEE International Workshop on Web Site Evolution, Amsterdam, The Netherlands, IEEE CS Press, 2003, pp. 12-19.
19. Di Lucca G. A., Fasolino A. R., De Carlini U., Pace F., and Tramontana P., Comprehending web applications by a clustering based approach. In Proc. of the 10th International Workshop on Program Comprehension, Paris, France, IEEE CS Press, 2002, pp 261-270.
20. Eichmann D. Evolving an Engineered Web. In Proc. International Workshop Web Site Evolution, Atlanta, GA, 1999, pp 12-16.
21. Ginige A. and S. Murugesan (guest editors), Special issue on Web Engineering, IEEE Multimedia, vol. 8, no. 1-2, 2001.
22. Girardi C., Pianta E., Ricca F., and Tonella P. Restructuring Multilingual Web Sites. In Proc. of 4th IEEE International Workshop on Web Site Evolution, Montreal, Canada, 2002, IEEE CS Press, pp. 290-299.
23. Hainaut J. L., Chandelon M., Tonneau C., and Joris M. Contribution to a Theory of Database Reverse Engineering. In Proc. of the 1st IEEE Working Conference on Reverse Engineering, Baltimore, MA, USA, IEEE CS Press, 1993, pp. 161-170.
24. Higo Y., Ueda T., Kamiya Y., Kusumoto S., and Inoue K. On software maintenance process improvement based on code clone analysis. In Proc. of the 4th International Conference on Product Focused Software Process Improvement, 2002, Rovaniemi, Finland, pp 185-197.
25. Isakowitz T., Stohr E. A., and Balasubramanian P. RMM: a Methodology for Structured Hypermedia Design. In Communications of the ACM, vol. 38, no. 8, 1995, pp. 34–44.
26. Isakowitz T., Kamis A., and Koufaris M. Extending the Capabilities of RMM: Russian Dolls and Hypertext. In Proc. of 30th Hawaii International Conference on System Science, Maui, Hawaii, USA, IEEE CS Press, 1997, pp. 177-186.

27. Kamiya T., Kusumoto S., and Inoue K. CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code. In IEEE Transactions on Software Engineering, 2002, vol. 28, no. 7.
28. Levenshtein V. L. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Cybernetics and Control Theory, vol. 10, 1966, pp. 707-710.
29. Messmer B. T. and Bunke H. A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection. In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 5, n. 20, 1998, pp. 493-503.
30. Ricca F. and Tonella P. Understanding and Restructuring Web Sites with ReWeb. In IEEE Multimedia, vol. 8, no. 2, 2001, pp. 40-51.
31. Ricca F. and Tonella P. Analysis and Testing of Web Application. In Proc. of International Conference on Software Engineering, Toronto, Ontario, Canada, 2001, pp. 25-34.
32. Ricca F. and Tonella P. Using Clustering to Support the Migration from Static to Dynamic Web Pages. In Proc. of 11th IEEE International Workshop on Program Comprehension, Portland, Oregon, 2003, pp. 207-216.
33. Ricca F., Tonella P., Girardi C., and Pianta E. An Empirical Study on Keyword-based Web Site Clustering. In Proc of 12th International Workshop on Program Comprehension, Bari, Italy, IEEE CS Press, 2004, pp 204-213.
34. Schwabe D. and Rossi G. Developing hypermedia applications using OOHDM. In Proceedings of Workshop on Hypermedia development Process, Methods and Models, Hypertext 98, 1998.
35. Ullman J. R. An Algorithm for Subgraph Isomorphism. In Journal of the Association of Computer Machinary, vol. 1, n. 23, 1976, pp. 31-42.
36. Wiggerts T. A. Using clustering algorithms in legacy systems remodularization. In Proc of 4th Working Conference on Reverse Engineering, Amsterdam Netherlands, 1997, pp. 33-43.
37. Wong K. Toward Reusable and Evolvable Web Sites. In Proc. of 1st International Workshop on Web Site Evolution, Atlanta, GA, USA, 1999.