Mining Interaction Patterns in the Design of Web Applications for Improving User Experience

Vassiliki Gkantouna
University of Patras
gkantoun@ceid.upatras.gr

Athanasios Tsakalidis
University of Patras
tsak@ceid.upatras.gr

Giannis Tzimas
Technological Educational Institute of Western Greece
tzimas@teimes.gr

ABSTRACT
The key success factor for modern web applications is their acceptance by the end-users which heavily depends on the quality of the user experience that they offer to them. Users require applications designed in such a way that it enables them to learn the supported functionalities easily, so that they can quickly find the information that they are looking for. Therefore, the usability and the overall design quality of an application can determine its success or failure. In this paper, we analyze the conceptual model of CMS-based web applications in terms of the incorporated design fragments that support the various user interaction processes. We consider these fragments as recurrent interaction patterns occurring in the application model, consisting of a configuration of front-end interface components that interrelate each other and interact with the user to achieve certain functionality. We have developed a methodology that automatically extracts the conceptual model of a web application and subsequently performs a pattern-based analysis of the model in order to identify the occurrences of all the recurrent interaction patterns. Finally, we calculate evaluation metrics revealing whether these patterns are used consistently throughout the application design. By utilizing these patterns, developers can produce more consistent and predictable designs, improving the ease of use of web applications.

Keywords
Design Evaluation; Design Patterns; Interaction Patterns; CMS; Design Quality; Web Applications

1. INTRODUCTION
Modern web applications are progressively becoming more and more complex in order to support sophisticated functionalities and integrate advanced business logic. To assist developers facing the intrinsic complexity of web application development, a plethora of Model-Driven Web Engineering approaches [1] has proposed the use of web design patterns ([2], [11]) for producing successful designs. They provide designers with proven solutions to recurring design problems that can be reused in different contexts where the correspondence problem arises. At the same time, the adoption of design patterns can also improve both the usability and the quality of an application since their use enforces a coherent design style and makes it easier for the end-users to recognize typical patterns of user interactions (i.e. interaction patterns). Despite the fact that there are catalogues of available web design patterns ([21], [22]), developers have difficulties to apply them consistently throughout the design model of an application, often resulting in pattern variants which can cause serious design inconsistencies and disorientation of user's perception about the system. This highlights the need for tools supporting developers inspect the design of web applications in terms of consistency and usability, even at the conceptual level, in order to discover potential design problems in the way they support the various user interaction processes, as it is crucial for the quality of user experience.

Towards this end, we analyze and evaluate the conceptual model of web applications in terms of the incorporated design fragments that support the various user interaction processes within an application's context. At the conceptual level, we consider these fragments as recurrent interaction patterns occurring in the application model, consisting of a configuration of front-end interface components that interrelate each other and interact with the user to achieve certain behavior or functionality. To be able to inspect the consistent use of these patterns, we also consider pattern variants. More specifically, we consider that an interaction pattern consists of a core specification, i.e., an invariant composition of front-end design elements that characterizes the pattern and by a number of pattern variants which extend the core specification with all the valid modalities in which the pattern composition can start (starting variants) or terminate (termination variants). We have developed a methodology that automatically extracts the conceptual model of a web application and subsequently performs a pattern-based analysis of the model in order to identify the occurrences of all the incorporated recurrent patterns. To verify that the identified patterns actually define user interaction processes for performing certain functionality, we additionally inspect their occurrences to examine whether the recurrence of the design elements at the hypertext design goes with a recurrence at the data level i.e. the content they deliver to the end-users. This is done by utilizing a semantic similarity measurement technique. Finally, we calculate evaluation metrics revealing whether these patterns are used consistently throughout the application design. We argue that the identified patterns can benefit both sides of the user interaction, the software developers and the end-users. By utilizing the patterns, developers can produce more consistent and predictable designs improving the ease of use of an application.

In order to automate the process of analyzing the design of a web application, we narrowed down the methodology's scope to the domain of Content Management Systems (CMSs), since they provide a common base of source code which can be systematically processed for obtaining the automated extraction of the application's conceptual model. The proposed methodology is accompanied by a tool support available in [5], allowing developers to apply the methodology on websites developed by using the Joomla! [13] and Drupal [8] CMS platforms. Due to space limitations, in this work we present the methodology for the case of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
HT '16, July 10-13, 2016, Halifax, NS, Canada
© 2016 ACM. ISBN 978-1-4503-4247-6/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2914586.2914631

Joomla!-based websites. The remaining of this paper is organized as follows: Section 2 provides an overview of the related work and discusses the contribution of this work. Section 3 presents in detail the methodology for analyzing and evaluating the conceptual model of a Web application in terms of interaction patterns, while section 4 discusses conclusions and future work.

2. RELATED WORK AND CONTRIBUTION

Our primary goal in this work is to inspect the design of web applications in terms of the design fragments (i.e. recurrent patterns) that allow users to interact with the application for performing certain task. We consider two main types of interaction patterns: (i) the one which results from applying well-known design patterns for handling common interaction design problems and (ii) the other which has emerged as a result of the design decisions made by developers for adopting specific reusable design structures to support the various ways in which users can interact with the system (i.e., user interaction processes) for performing certain tasks within the application context. The latter type can result in either effective reusable design solutions consistently used into the application model for supporting predictable user behavior and making the application easy to use, or in problematic design cases causing inconsistencies that introduce unpredictability and make users find the interaction with the system an ineffective process.

The concept of interaction patterns appears in the field of Human Computer Interaction. In [16], [18], [19] and [20], there is a variety of UI patterns specifications which have derived from analyzing and reviewing the UIs of a large number of successful web applications. Related to UI patterns are the web interaction design patterns that provide solutions to recurring interaction problems of web applications users. Specific Web UI patterns can be found in [6], [12], [14] and [22]. Although the research on detecting software design patterns is mature, the research on the automated detection of design patterns in the conceptual model of an application is very limited. In [9], authors present the Web Quality Analyzer which analyzes the conceptual schema of WebML-based web applications and identifies the occurrences of a predefined set of WebML design patterns, allowing designers to automatically monitor the application's design consistency. In [10], authors present an approach for supporting the automatic identification of web interaction design patterns in a Web application. The approach is based on reverse engineering techniques which search the code of a website's pages for detecting a set of predefined features that characterize a pattern.

The key difference of our approach is that there is no limitation to focus strictly on the detection of predefined interaction patterns in the design of an application. On the contrary, we provide a methodology for supporting the automated detection of all the recurrent design structures occurring within the conceptual model for supporting the various user interaction processes. Such structures may be a well-known design pattern or they can also be reusable design compositions (effective or not) used by the developers to accomplish certain functionality. In this way, we can detect even patterns which may be hidden in a particular instantiation of a design problem, making it hard even for experienced designers to recognize them and come up with reusable design examples. These patterns can probably lead to: (i) the identification of new interaction design patterns for handling common interaction problems in the CMS domain which can be used as building blocks in future designs, offering a high quality user experience, or (ii) they can even stand as anti-patterns in case they are evaluated to cause serious design inconsistencies. To the best of our knowledge, this is the first effort to provide a mechanism for supporting the automated identification of interaction design patterns and anti-patterns for the CMS domain and for promoting the pattern-based CMS design and development.

3. METHODOLOGY

In this section, we present the methodology for automatically mining and evaluating the incorporated interaction patterns in the conceptual model of a web application. First, we extract the conceptual model of the application, at hypertext level, in the form of a directed graph which is then submitted to a pattern-based analysis with the aim of (i) identifying the occurrences of all the recurrent patterns lying within it, (ii) verifying which of them support a user interaction process for implementing certain functionality and (iii) calculate a set of evaluation metrics to assess if the identified patterns are used consistently throughout the application model. In order to explain the concepts and illustrate the potential of the proposed methodology, in what follows, we refer to various instances of a real website, called the AtticaBank[1], which has been developed on the Joomla! CMS platform.

3.1 Extracting the Conceptual Model

At the hypertext level, the conceptual model of a web application specifies its composition and navigation. The composition of a website defines which pages compose its hypertext and their internal organization in terms of design elements for publishing content and allowing users to interact with the system. The navigation defines the different ways in which the various design elements and pages can be connected through links to specify the allowed navigation paths, the selections offered to the end-users as well as the sequence of pages presented to them when they interact with the content displayed in a page. In other words, the conceptual model of a Web application specifies the organization of its front-end interfaces in terms of pages, made of design elements which are linked to support the user's navigation and interaction. Thus, the main task for automatically extracting the conceptual model of a website is to identify the organization of the front-end design elements that compose the hypertext of its HTML pages. In the context of a Joomla!-based website, such design elements are called components and modules. A page is composed by one component specifying the organization of content in its main part, and by a set

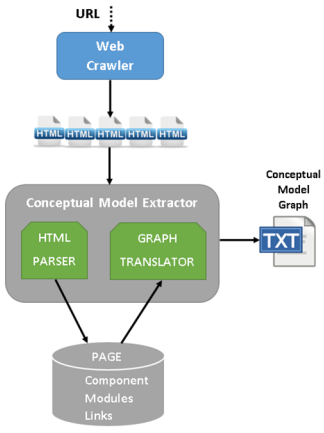

Figure 1. The conceptual model extraction process.

[1] Available at http://www.atticabank.gr/en/

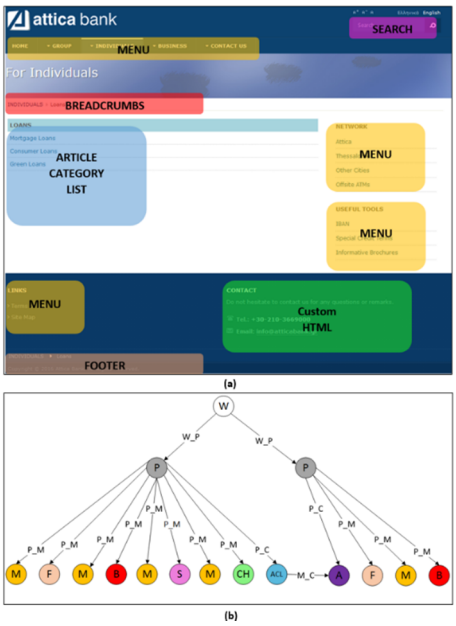

Figure 2. (a) The organization of a page in terms of Joomla! design elements (component and modules). (b) The equivalent graph representation of the page's hypertext.

of modules specifying the organization of content in its peripheral positions. There is a variety of components and modules categories, each one containing various types for interacting with the system and supporting alternative ways of arranging the content delivered to the end-users. The content that they publish is extracted from the tables of the website's underlying database. To specify the hypertext organization for all the pages of a website, we have developed a set of tools as depicted in Figure 1.

By given the URL of a Joomla!-based website, the Web Crawler crawls its pages which are then parsed by the Conceptual Model Extractor to identify their organization as a set of components and modules. In the HTML code of a page, components and modules can be found as elements having an HTML class attribute value (i.e.) which specifies its style i.e. the exact type of the component-module it represents (available in [5]). Thus, by parsing the HTML code of a page and locating the occurrences of these characteristic values within it, we can recover the page's organization as a set of Joomla! design elements. The identified components and modules within a page are stored in the "Page" repository along with additional information about the incoming and outgoing links of the page. For example, Figure 2(a) presents the Joomla! design elements identified for the "Loans" page of the AtticaBank website. As we can see, the page consists of the "Article Category List" component which displays a list with the articles of the "Loans" category and a set of modules such as menus, footer, etc. Once this is done for all the pages of the website, we manage to capture its composition and navigation, i.e. to capture its conceptual model. Then, we employ the Graph Translator which represents the recovered model as a directed graph. An example can be found in Figure 2(b) which presents an instance of the conceptual model graph for the AtticaBank website (root node W), consisting of two connected pages. The left page node (P) represents the equivalent graph representation of the "Loans" page depicted in Figure 2(a). By selecting one of the available list items (i.e. the various loans types) published by the ACL node of the left page node, the user can navigate to the right page node consisting of an A (Article) component which publishes information for the

Figure 3. An example of two isomorphic subgraphs. A stands for the Article, ACL for the Article Category List, LA for the Latest Articles and S for the Search box.

selected loan type. Finally, the tool produces a TXT file as an output, containing the graph representation of the conceptual model, which is going to be the input for the graph mining algorithm of the next phase.

3.2 Identification of the Recurrent Interaction Patterns

After the extraction of the application's conceptual model, the next step is to inspect and analyze it in terms of the recurrent patterns (i.e., design fragments) that support user interaction processes for achieving certain tasks. These interaction patterns are configurations of Joomla! components and modules which when located in a particular layout may serve a certain application purpose. To this end, we perform a pattern-based analysis of the recovered model aiming to identify and evaluate the occurrences of all the incorporated recurrent interaction patterns.

To identify these patterns (their core specifications along with their starting and termination variants), we have used an approach which is based on the subgraph isomorphism problem. The latter is synopsized to finding whether the isomorphic image of a subgraph exists in a larger graph. An example of two isomorphic subgraphs (having components and modules on the nodes) is depicted in Figure 3. As we can see, despite the different configuration of nodes in the two subgraphs, the edges connecting the nodes of the same color/label remain the same. Table 1 contains some sequences of the nodes that are connected in the subgraphs of Figure 3. By observing these sequences, one can notice that they actually reveal the recurrent patterns within a graph, both their core specifications and their starting and termination variants. Clearly, the identification of the isomorphic subgraphs within a graph is an alternative way to obtain the identification of the incorporated recurrent patterns. Based on this, we have employed a graph mining algorithm which identifies the occurrences of all the recurrent patterns lying in the conceptual model, by locating all the isomorphic subgraphs within its equivalent graph representation. Then, in order to verify that the identified recurrent patterns are actually interaction patterns, i.e. they define a user interaction process for performing a certain functionality, we additionally inspect their occurrences to examine whether the recurrence of the design elements occurring at the hypertext design goes with a recurrence in the content-data they deliver to the end-users. To achieve this, we have used a semantic similarity measurement

Table 1. The sequences of the connected nodes.

Starting Variant	Pattern's Core Specification			Ending Variant
	S	ACL	A	
	S	ACL	A	S
LA	S	ACL	A	
	ACL	S	LA	
A	ACL	S	LA	
	ACL	S	LA	A

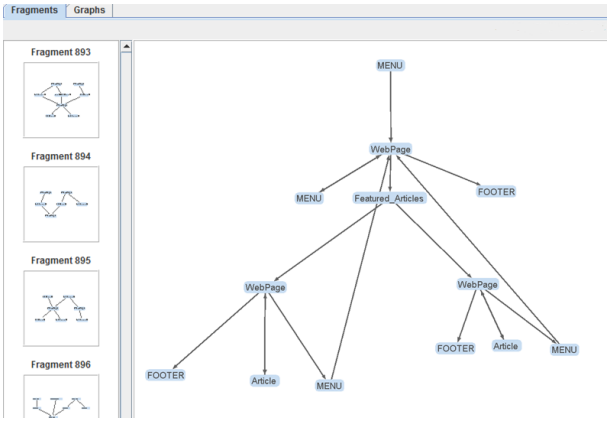

Figure 4. A recurrent subgraph detected in the conceptual model of the AtticaBank website.

technique among the patterns occurrences. The following subsections present the aforementioned tasks in detail.

3.2.1 Mining the Recurrent Patterns

In order to identify the recurrent patterns within the conceptual model graph G, we apply the gSpan algorithm [23] on the graph for detecting the isomorphic subgraphs images. To apply the gSpan, we use the Parsemis project [15] which supports an implementation of the algorithm within a graphical environment for visualizing the identified frequent subgraphs. When looking for the occurrences of a subgraph in G, gSpan encounters except for its identical occurrences, its isomorphic images too. In this way, the graph G is analyzed in terms of its frequent subgraphs (representing recurrent patterns). An example can be found in Figure 4 which presents an identified subgraph-pattern for the AtticaBank website. The subgraph consists of a composition of Joomla! design elements (components and modules) involving three different pages of the website. The Parsemis tool provides the identified subgraphs in a TXT file containing the configuration of the Joomla! design elements that compose each subgraph as well as their occurrences in the graph. Then, we process this file in a way similar with the one presented in the example of Table 1 (based on the sequences of the connected nodes in the subgraphs) and identify the core specifications and the starting and ending variants of each identified pattern.

3.2.2 Interaction Patterns Selection

In this task, we inspect the occurrences of the identified recurrent patterns to examine whether the recurrence of the design elements at the hypertext design goes with a recurrence in the content-data they deliver to the end-users. The content published by the various components and modules is extracted from the tables of the website's database. Thus, to examine if there is a coexisting recurrence at the data level, we have to examine from which tables of the database the Joomla! components and modules (that make up a pattern) extract content. If the corresponding components and modules in a pattern's occurrences publish content from the same data tables, then there is a high possibility of identifying a reusable interaction pattern for implementing a certain task.

However, we assume that we do not have access to the database of the website under study. Based on this, we attempt to examine if there is a recurrence at the data level, by computing the semantic closeness of the content published by the pattern's corresponding Joomla! design elements among its occurrences. The rationale behind this is that the contents of the pages that come from the same database's table usually have a very close semantic relation. To

Table 2. Measuring semantic similarity among occurrences.

PATTERN	MENU [Module]	CATEGORY LIST [Component]	CATEGORY LIST [Component]	ARTICLE [Component]
Occ.1	Individuals Top Menu	Individuals Deposits Categories Page	Individuals Deposits Time Accounts Category Page	Individuals SingleTerm Deposits Details
Occ.2	Business Top Menu	Business Deposits Categories Page	Business Deposits Time Accounts Category Page	Business SingleTerm Deposits Details
SemSim Score		72%	100%	100%
AverageSemSimScore Occ1.-Occ.2				90%
Occ. 3	Individuals Top Menu	Individuals Loans Categories Page	Individuals Mortgage Loans Category Page	Individuals Housing Mortgage Loan Details
SemSim Score		30%	30%	20%
AverageSemSimScore Occ1.-Occ.3				27%

compute the semantic closeness of the published content between two occurrences of a pattern, we have defined two metrics, the "SemSimScore" and the "AverageSemSimScore". On the grounds that the main content of a page, published by components, is indicative of the page's semantics, the "SemSimScore" metric addresses the semantic similarity measurement of the content published by the Joomla! components occurring in a pattern. Then, the "AverageSemSimScore" computes the average value of the individual "SemSimScore" values between the pattern occurrences. To compute the "SemSimScore" metric, we have used the methodology proposed in [17] for WordNet-based semantic similarity measurement. In Table 2, we can see an example of a pattern supporting the user's navigation among the various categories of an object type. To verify that the pattern is actually used in the AtticaBank website, we inspect its occurrences to examine if there is also a recurrence at the content that the pattern's components deliver to the end-users. In Table 2, we can see three occurrences of the pattern Occ.1, Occ.2 and Occ.3. In Occ.1, all the categories and the subcategories of the deposits for Individuals customers are displayed. In Occ.2, the same happens except for the fact that it is intended for Business customers. By comparing the semantic similarity of the content published by the pattern's corresponding components for these two occurrences, they have an AverageSemSimScore of 90% which means that they are semantically very close. Occ.3 refers to the case of displaying all the categories and the subcategories of the loans for Individuals customers. In the same way, we can compute the AverageSemSimScore for Occ.1 and Occ.3 which is 27%, implying that these occurrences are not semantically close. This is actually expected, since their only common base is that they refer to the customer of type Individuals, the rest of the content displayed on them is irrelevant. By using the AverageSemSimScore metric, we can obtain a safe estimation about the recurrence at the data level among the occurrences of the identified patterns. We compute the AverageSemSimScore metric for all the occurrences of the identified patterns (core specification and variants) and we select

Table 3. The measurement scale for the SPM metric.

SPM range	Measurement scale value
0 ≤ SPM < 0.2	Insufficient
0.2 ≤ SPM < 0.4	Weak
0.4 ≤ SPM < 0.6	Discrete
0.6 ≤ SPM < 0.8	Good
0.8 ≤ SPM ≤ 1	Optimum

and store in a "Interaction Patterns Repository" only the ones having an AverageSemSimScore over 70%.

3.3 Evaluation of Interaction Pattern Variants Consistent Use

In this final step, we focus on evaluating the consistent use of the identified interaction patterns for determining their impact on the overall application's quality. Patterns which are consistently used throughout the conceptual model of an application enhance the ease of use of the application, since they facilitate users identify typical sequences of interactions with the system for performing common tasks. This results in foreseeable navigation behavior, enhancing the quality of the user experience. On the other hand, patterns which are not use consistently cause design inconsistencies and make the user feel disoriented. To this end, we calculate some metrics to evaluate whether the patterns stored in the "Interaction Patterns Repository" are used consistently throughout the conceptual model. In [9], authors have introduced a methodology for evaluating the consistent application of predefined WebML design patterns within the conceptual schema of a WebML-based applications. We utilize this methodology in order to introduce metrics computing the consistent application of an interaction pattern's variants throughout the application model. Assuming a pattern can have N starting and M termination variants, we have defined two metrics that compute the statistical variance of the occurrences of the N starting and the M termination variants of the pattern, normalized according to the best-case variance. These metrics are called Start-Point Metric (SPM) and End-Point Metric (EPM) respectively. SPM is defined as (EPM is defined in an analogous way):

$$SPM = \sigma^2 / \sigma_{BC}^2 \qquad (1)$$

σ^2 is the statistical variance of the N starting variants occurrences which is calculated according to the formula (2):

$$\sigma^2 = \frac{1}{N} \sum_{i=0}^{N} \left(p_i - \frac{1}{N} \right)^2 \qquad (2)$$

where p_i is the percentage of occurrences for the i-th pattern variant. σ_{BC}^2 is instead the best case variance and it is calculated by the formula (2) assuming that only one variant has been coherently used throughout the application. The last step in the metrics definition is the creation of a measurement scale (Table 3) which defines a mapping between the numerical results obtained through the calculus method and a set of (predefined) meaningful and discrete values, expressing a consistency level. We compute these metrics for all the occurrences of all the interaction patterns' starting and ending variants and store the results in the "Results Repository", which are also provided in a TXT file ("Results"). For example, in Table 4, we consider the core specification of an interaction pattern for allowing users browse the hierarchy of Deposits categories and subcategories. In Table 4, there are two occurrences of the two starting variants of the pattern. In fact, in these occurrences, the pattern variants perform the same functionality, except for the fact that they differ on the starting point they provide to users for browsing the hierarchy of categories. The value of the SPM metric for the first variant is about 0.68 ('Good') since it is a typical way for browsing the hierarchy of categories for various object types whereas the value for the second variant is 0.39 ('Weak') due to the fact that it has a limited number of occurrences. By observing the low value of the second variant on the "Results" file, developers can inspect the identified occurrences of the variant on the AtticaBank website in order to verify if it is actually a design inconsistency. It is worth noting that in some cases it could happen that detected inconsistencies can be caused by conscious design choices in order to respond to specific application constraints. This is just a simple example of showing how the proposed methodology can be used to assist the design process and support designer to improve the quality of a website by highlighting design fragments that need to be inspected in order to make interaction with the application an easy and effortless process.

Table 4. Pattern variants.

VAR. 1	MENU [Module]	CATEGORY LIST [Component]	CATEGORY LIST [Component]	ARTICLE [Component]
Occ. 1	Individuals Top Menu	Individuals Deposits Categories Page	Individuals Deposits Time Accounts Category Page	Individuals Single Term Deposits Details
VAR. 2	BANNER [Module]	CATEGORY LIST [Component]	CATEGORY LIST [Component]	ARTICLE [Component]
Occ. 4	Home Page Banner	Individuals Deposits Categories Page	Individuals Deposits Time Accounts Category Page	Individuals Single Term Deposits Details

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have illustrated a model-driven approach for mining interaction patterns in the conceptual model of websites built on the Joomla! CMS. To achieve this, we provide an automated way for extracting the conceptual model of a website which is then submitted to a pattern-based analysis for the identification of all the interaction patterns lying within it. Finally, the identified patterns variants are evaluated towards their consistent use throughout the application. Most of the work presented here can be applied also to web applications built by using other CMSs with slight straightforward modifications. By applying the methodology on a website, developers can gain important information regarding its design quality and particularly about the various design fragments that allow the users to interact with it for performing certain task. On one side, the methodology can detect effective reusable design solutions consistently used throughout the application for supporting predictable user navigation behavior in order to perform certain task. Such reusable solutions can be used as building blocks for producing predictable future designs. They also facilitate the discovery of new interaction design patterns for the CMS domain. On the other side, the methodology can also detect recurrent design constructs indicating design inconsistencies making the user feel disoriented.

In the future, we plan to apply the methodology to a very large number of domain-specific websites for populating a central interaction patterns repository. In this way, it is possible to come up with useful design guidelines for building successful Joomla!-based websites.

5. REFERENCES

[1] Aragón, G., Escalona, M. J., Lang, M., Hilera, J. R., 2013. An Analysis of Model-Driven Web Engineering Methodologies, In International Journal of Innovative Computing, Information and Control, Vol. 9, no. 1, pp. 413-436.

[2] Bernstein, M., 1998. Patterns of Hypertext. In Hypertext 98 - Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia, Pittsburgh, PA, USA, June 20-24, 1998, pp. 21-29.

[3] Brambilla, M., Mauri, A., 2012. Model-Driven Development of Social Network Enabled Applications with WebML and Social Primitives, In Grossniklaus, M., Wimmer, M., (Eds.), Current Trends in Web Engineering, LNCS, Springer, Berlin, 2012, Vol. 7703, pp. 41-55.

[4] Ceri S., Fraternali P., Bongio A., 2000. Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. In the Proceedings of WWW Conference. Amsterdam, NL, May 2000, pp. 137-157.

[5] CMS Patterns, (2015) Available at: http://alkistis.ceid.upatras.gr/research/modeling/patterns/ (Accessed: 22 February 2016).

[6] D.K. van Duyne, J.A. Landay, J. Hong, 2002. The design of sites, Addison-Wesley, Boston, US.

[7] Díaz, P., Rosson, M.B., Aedo, I., Carroll, J.M., 2009. Web design patterns: Investigating user goals and browsing strategies. In: Pipek, V., Rosson, M.B., de Ruyter, B., Wulf, V. (eds.) IS-EUD 2009. LNCS, vol. 5435, pp. 186–204. Springer, Heidelberg.

[8] Drupal CMS (2015). Available at: https://www.drupal.org/ (Accessed: 22 February 2016).

[9] Fraternali, P., Matera, M., Maurino A., 2002. WQA: an XSL Framework for Analyzing the Quality of Web Applications. In the Proceedings of the 2nd International Workshop on Web-Oriented Software Technologies – IWWOST'02. Malaga, Spain, June 10–14, 2002, pp. 46–61.

[10] G.A. Lucca, A.R. Fasolino, and P. Tramontana, 2005. Recovering interaction design patterns in web applications, Proceedings of the Ninth European Conference on Software Maintenance and Reengineering, March 2005, pp. 366-374.

[11] Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

[12] Graham, 2003. A pattern language for Web Usability, Addison-Wesley, Boston, US.

[13] Joomla! CMS (2015). Available at: http://www.joomla.org/ (Accessed: 22 February 2016).

[14] M. van Welie, G. C. van der Veer, 2003. Pattern Languages in Interaction Design: Structure and Organization, Proceedings of Ninth International Conference on Human-Computer Interaction, Interact 2003, Zürich, Switzerland, pp. 527-534.

[15] Philippsen, M., 2011. ParSeMiS - the Parallel and Sequential Mining Suite. Available at: https://www2.cs.fau.de/EN/research/zold/ParSeMiS/index.html (Accessed: 22 February 2016).

[16] Scott, B., Neil, T., 2009. Designing Web Interfaces: Principles and Patterns for Rich Interactions. O'Reilly, Sebastopol.

[17] Simpson, T., Dao, T., 2010. WordNet-based semantic similarity measurement. Available at: http://www.codeproject.com/Articles/11835/WordNet-based-semantic-similarity-measurement (Accessed: 22 February 2016).

[18] Tidwell, J. 2006. Designing Interfaces. O'Reilly, Sebastopol.

[19] Toxboe, A., 2010. Pattern library. Available at: http://ui-patterns.com (Accessed: 22 February 2016).

[20] Valverde, F., Pastor, O., 2008. Applying interaction patterns: Towards a model-driven approach for rich internet applications development. In Proceedings of the International Workshop on Web Oriented Software Technology (IWWOST). CEUR Workshop. CEUR-WS.org, RWTH Aachen, 13–18.

[21] Website patterns (2015). Available at: http://c2.com/cgi/wiki?HypermediaDesignPatternsRepository (Accessed: 22 February 2016).

[22] Welie, M. v. (2008): Interaction Design Patterns. Available at http://www.welie.com/patterns/ (Accessed: 22 February 2016).

[23] Yan, X., Han, J., 2002. gSpan: Graph-based substructure pattern mining. In ICDM '02, page 721, Washington,DC, USA, 2002. IEEE Computer Society