Mining Domain-Specific Design Patterns

Vassiliki Gkantouna, Giannis Tzimas, Basil Tampakas, John Tsaknakis

Mining Domain-Specific Design Patterns

Vassiliki Gkantouna[1], Giannis Tzimas[2], Basil Tampakas[2], and John Tsaknakis[2]

[1] Department of Computer Engineering and Informatics, University of Patras, Patras, Greece
gkantoun@ceid.upatras.gr
[2] Computer and Informatics Engineering Department, Technological Educational Institute of Western Greece, Patras, Greece
{tzimas, tampakas, jtsaknakis}@teiwest.gr

Abstract. Most catalogues of web design patterns contain patterns of general purpose, making it difficult for developers to properly apply them. This has led to the advent of domain-specific design patterns, encapsulating design experience which is in alignment with the natural constraints of a particular domain. Towards this end, we have developed a methodology which when applied on a collection of websites in a particular domain, leads to the automated identification of domain-specific design patterns. At the level of a single website, the methodology analyzes its conceptual model in terms of the incorporated recurrent patterns and evaluates their consistent use. The identified design patterns are stored in a central repository. By applying the methodology on a set of websites of the same application domain, we can populate a repository containing all the design patterns identified within the various websites designs, categorized towards various aspects such as the domain functionalities they perform. In this work, we focus on the domain of educational websites and present our preliminary results.

Keywords: Domain-specific, Design Pattern, Web application, CMS, Design Quality, Web Design

1 Introduction

Web design patterns support developers facing the intrinsic complexity of web application development by providing them with proven solutions to recurring design problems that can be reused in different contexts where the correspondence problem arises. However, due to the fact that most design patterns are of general purpose (i.e., too abstract and divorced from the context in which sites are being developed), developers often find it difficult to properly use them. This fact has led to the advent of domain-specific design patterns, encapsulating design experience which is in alignment with the natural constraints of a particular application domain. They offer developers solutions to common design problems that arise particularly in a specific application domain, enabling them to produce quality designs.

Towards this end, we have developed a methodology which when applied on a collection of websites in a particular domain facilitates the automated identification of domain-specific design patterns. In the case of a single website, the methodology analyzes

and inspects its conceptual model in terms of the incorporated design fragments that are used repeatedly for supporting the various functionalities within the application's context. At the conceptual level, we consider these fragments as recurrent patterns occurring in the application model, consisting of a configuration of front-end interface components that interrelate each other and interact with the end-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 a 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). The proposed methodology automatically extracts the conceptual model of a web application and subsequently performs a pattern-based analysis of it in order to identify the occurrences of all the incorporated recurrent patterns. Then, to verify that the identified patterns actually perform 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 on them, revealing whether these patterns are used consistently throughout the entire application design and store them on a central patterns repository.

To be able to mine domain-specific design patterns, the methodology must be applied on a collection of websites that belong to the same application domain. By applying the methodology on the websites, we can detect and store the design patterns used in the various websites designs in one single central repository. This way, we can identify the most typical design patterns that are commonly used by developers in this specific application domain for achieving certain application behavior or communication effects such as the checkout process in e-commerce applications. Furthermore, we can also categorize the identified patterns towards various aspect such as the application functionality they perform. In the context of this work, we focus on the domain of educational websites and present a number of design patterns that we have identified for this domain. These patterns can assist developers produce more consistent and predictable educational website designs, meeting users' expectations about the system and improving the ease of use of the application.

In order to automate the process of analyzing the design of a web application, we have to narrow 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. The remaining of this paper is organized as follows: Section 2 provides an overview of the related work. Section 3 presents in detail the methodology for analyzing and evaluating the conceptual model of a Web application, while section 4 presents some examples of domain-specific design patterns for educational websites. Finally, section 5 discusses conclusions and future work.

2 Related Work

Research in the field of domain-specific design patterns is not mature. In [7], authors present an ontology-based approach that allows designers and domain experts to enrich their domain-specific patterns with semantic annotations using their domain concepts and terms. The representation framework keeps the pattern template and the domain specific knowledge separate, since each domain depending on its needs uses more or less fields to describe its patterns and the later ontology changes according to the domain knowledge that the pattern captures. In [8], authors present an approach of using design patterns at PIM (platform independent model) level, which combines the benefits of model-driven software engineering and design patterns directly for the domain expert. In [9] and [10], authors present an integrated pattern definition process that supports both the top-down and bottom-up design approaches. It is distinguished from existing methods in two ways: (i) it is based on UML-profile language that visually distinguishes between the fundamental and variable elements of a pattern, and (ii) it defines the different steps that must be taken to obtain a domain-specific design pattern as well as a precise set of unification rules that identifies the commonalities and differences between applications in the domain. In all the aforementioned works, the domain-specific design patterns are manually devised by human experts after analyzing the designs of successful applications. The key difference of our approach is that we provide a methodology for the automated identification of the recurrent design patterns lying in the designs of the websites in a specific application domain. We can detect even patterns which may be hidden in a particular instantiation of a design problem in a specific website design, making it hard even for experienced designers to recognize them and come up with reusable design examples. By applying the methodology on their websites, developers can compare their designs, the design patterns they have used for supporting certain application functionalities, and see how their design choices differentiate from the most common design patterns used in an application domain.

3 Methodology Overview

In this section, before presenting the design patterns that we have identified for the educational websites domain, we present a brief overview of the methodology that we have used to detect them. At the level of a single website, the application of the methodology results in the automated identification and evaluation (towards consistency) of all the recurrent design patterns lying within its conceptual model. It is comprised by three main phases (**Fig. 1**). First, we extract the conceptual model of the website, at hypertext level, 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 can actually be considered as design patterns (i.e. support the realization of a certain functionality). Then, we calculate a set of evaluation metrics to assess if they are used consistently throughout the application model. Finally, the identified design patterns are stored in a repository available at [1]. A detailed description of the methodology can be found in [2].

3.1 Phase 1: Extracting the Conceptual Model

At the hypertext level, the conceptual model of a web application specifies its composition and navigation, i.e. 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, the hypertext of its pages is composed by assembling a number of predefined structural and navigational design elements which 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 of modules, specifying the organization of content in the peripheral positions. There is a variety of categories for components and modules, each one providing various types for interacting with the system (such as forms, confirmation buttons, etc.) and supporting alternative ways of arranging the content delivered to the end-users (e.g., blogs, lists, etc.). 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 the first phase of **Fig. 1**.

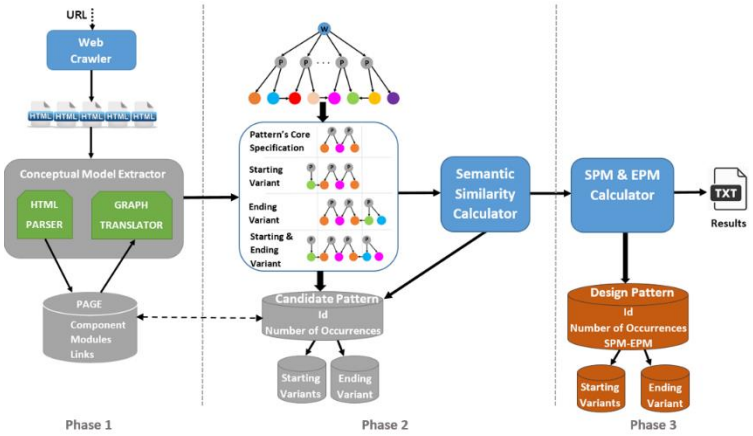

Fig. 1. The tree main phases of the methodology.

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 characterizes them and specifies the exact type of the component-module they represent (the complete list with these values is available in [1]). 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. For example, **Fig. 2**(a) presents the Joomla! design elements identified

within a page of the MMLAB[1] educational website. As we can see, the page consists of the "Article" component 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. its conceptual model. Then, we employ the Graph Translator for representing the recovered model as a directed graph (**Fig. 2**(b)), which is going to be the input for the graph mining algorithm of the next phase.

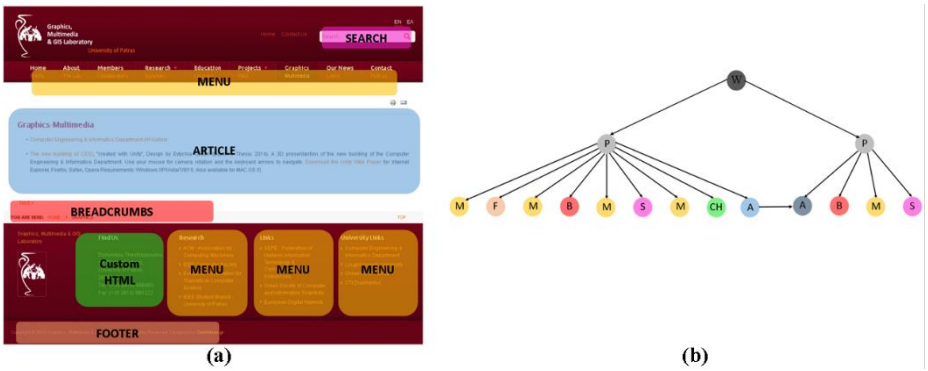

Fig. 2. (a) The organization of a page in terms of Joomla! design elements (component and modules). (b) The equivalent graph representation of the page's hypertext.

3.2 Phase 2: Identification of the Recurrent Design 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 the various application's functionalities for achieving certain purposes. These patterns are configurations of Joomla! components and modules which when located in a particular layout, they may serve a certain application purpose. To this end, we perform a pattern-based analysis of the recovered model to identify and evaluate the occurrences of all the incorporated recurrent patterns (**Fig. 1**, Phase 2).

Mining the Recurrent Patterns.

To identify the recurrent patterns (their core specifications along with their starting and termination variants), we have used an approach which is based on the subgraph isomorphism problem, synopsized into finding whether the isomorphic image of a subgraph exists in a larger graph. As presented in detail in [2], 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 applied the gSpan algorithm [4] on the conceptual model graph G, which identifies the occurrences of all the recurrent patterns lying in the conceptual model, by locating all the isomorphic subgraphs images within it. In this way, the graph G is analyzed in terms of its frequent subgraphs

[1] Available at http://mmlab.ceid.upatras.gr/en/graphicsmultimedia/

(representing recurrent patterns). In this way, we manage to identify the core specifications as well as the starting and ending variants of the recurrent subgraph-pattern.

Design Patterns Selection.

In order to verify that the identified recurrent patterns can actually specify design patterns, i.e. they perform a certain functionality, we additionally inspect their occurrences on the website to examine whether the recurrence of the design elements occurring at the hypertext design goes with a recurrence at the data level, i.e., the content they deliver to the end-users. To achieve this, we have used a semantic similarity measurement technique [5] among the patterns occurrences. The content published by the various components and modules is extracted from the tables of the website's database. Thus, to examine for a given recurrent pattern whether there is a coexisting recurrence at the data level, we have to examine from which database tables the corresponding Joomla! components and modules (that make up the pattern) among its occurrences extract content. If they publish content from the same tables, then there is a high possibility of identifying a reusable design pattern for implementing a certain task. However, in this work we assume that we do not have access to the database of a website, since this is the common scenario in real-life websites. 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. So, if the pattern's occurrences are semantically close, we can assume that they could probably derive content from the same database tables, and infer that the pattern is used for certain purpose, i.e., to provide certain information object to end-users.

To 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. This is why there are empty cells for the "SemSimScore" computations in Table 1, when it comes to measure semantic similarity among content published by modules. Given two contents, the "SemSimScore" metric determines how similar the meaning of two contents is. The higher the score, the more similar the meaning of the two contents is, increasing the possibility that there is also a recurrence at the content displayed by the components of a pattern between its occurrences. Then, the "AverageSemSimScore" computes the average value of the individual "SemSimScore" values between the pattern occurrences. In Table 1, we can see an example of a pattern occurring on the MMLAB website. To verify that this pattern is actually used in the website for supporting a functionality, 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 1, we can see three occurrences of the pattern Occ.1, Occ.2 and Occ.3. By comparing the semantic similarity of the content published by the pattern's corresponding components for Occ.1 and Occ.2, they have an AverageSemSimScore of 85%

which means that they are semantically very close. Similarly, the AverageSemSim-Score for Occ.1 and Occ.3 is 0.04% implying that the content published in these two occurrences is irrelevant. As a result, we can assume that the pattern in Table 1 is used for supporting the user's navigation among the various categories of a specific information object, i.e., the various fellowships categories. In this way, 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 and store in a "Candidate Patterns Repository" only the ones having an AverageSemSiScore over 70%.

PATTERN	MENU [Module]	CATEGORY LIST [Component]	ARTICLE [Component]
Occ.1	Top Menu	Postgraduate Fellowships List	Fellowship Description
Occ.2	Top Menu	Undergraduate Fellowships List	Fellowship Description
SemSimScore Occ.1-2		90%	80%
AverageSemSimScore Occ1.-Occ.2			85%
Occ.3	TopMenu	R&D Projects	Project Description
SemSimScore Occ.1-3		5%	2%
AverageSemSimScore Occ1.-Occ.3			0.04%

Table 1. Measuring semantic similarity among pattern's occurrences.

3.3 Evaluation of Pattern Variants Consistent Use

In this final step, we focus on evaluating the consistent use of the identified patterns for determining their impact on the overall application's quality. Patterns which are used consistently throughout the conceptual model of an application result in high design quality, facilitating end-users identify typical sequences of interactions with the system for performing common tasks. This results in foreseeable navigation behavior, and thus high design quality. On the other hand, patterns which are not used consistently may cause serious design inconsistencies. To this end, we calculate some metrics to evaluate whether the patterns stored in the "Candidate Patterns Repository" are used consistently throughout the conceptual model. These metrics are called Start-Point Metric (SPM) and End-Point Metric (EPM) and intuitively they compute the statistical variance of the occurrences of the starting and termination variants of a pattern throughout the application model. 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):

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

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. More details about the metrics definition can be found in [2]. We have also specified a measurement scale which defines a mapping between the numerical results obtained through the calculus method of the SPM-EPM metrics and a set of (predefined) meaningful and discrete values, expressing different consistency levels (Table 2).

EPM-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

Table 2. The measurement scale for the EPM and SPM metrics.

We compute these metrics for all the occurrences of all the patterns' starting and ending variants and store the results in the "Design Patterns Repository", which are also provided in a TXT file ("Results"). By applying the methodology on a website, developers can gain important information regarding its design quality and particularly about the various design fragments that they have chosen to use for realizing the various tasks in the application's context. On one side, the methodology can detect effective reusable design solutions consistently used throughout the application for supporting certain functionality. Such reusable solutions 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 lowering the application's quality.

4 Domain-Specific Design Patterns - A case study: the educational websites domain

In the context of this work, we focused on the domain of educational web-sites and we applied the previously described methodology on a collection of such type of websites. This way, we have identified a set of design patterns for this specific domain, stored in a central repository available at [1], a number of which are presented in the current section. Our dataset is composed by a number of educational websites that have been developed by our team, such as the MMLAB website, and a set of websites from educational websites category of the official Joomla! Community Showcase website catalogue [6]. In the following sections, we provide a short description of this domain and a number of the design patterns we have identified for this area.

4.1 Domain description

College and university websites are part of institutions' identity, having an important role in their marketing practices. They provide useful information to prospective students and they have a strong impact on forming user's opinion during the college search process. Thus, educational websites must be designed in such a way, that it enables end-users to recognize typical interaction patterns with the system in order to quickly find the information that they are looking for. In the context of educational websites, the common types of end-users are the prospective students, the current undergraduate and postgraduate students, and the alumni. Furthermore, the common types of information objects that can be found in such websites include: courses catalogues, degree programs for graduate diplomas, master and doctoral degrees, research and publications, admissions and financial aid, faculty, students' life, news and events, etc.

4.2 Design patterns for educational websites

In this section, we present some of the design patterns that we have identified within the various website designs. We have classified these patterns into three main categories, as listed below.

Layout: this category includes patterns that provide developers with solutions for standard page types within the context of educational websites, in terms of the design components that specify the page's layout. These page types include:

Homepage: specifies a template for the design of the website's homepage. For example, **Fig. 3** depicts one of the most prevalent design patterns for the design of the homepage that we found in our websites dataset. It consists of a welcome message based on the single article component (usually having a link to the About us page) and a number of modules such as banners which provide quick links usually to undergraduate and postgraduate information pages. Additionally, they use the most read module for direct access to the available educational programs pages and the latest articles module for publishing their news.

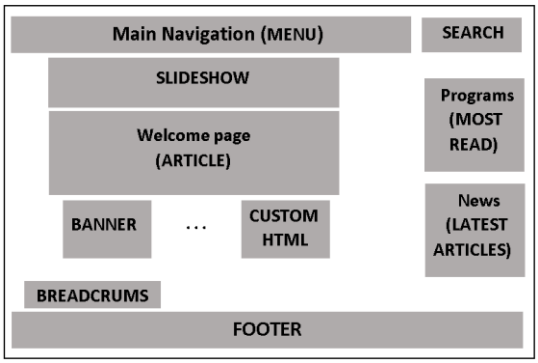

Fig. 3. Design patterns for the design of the Homepage.

Browsing Information Object's Categories Hierarchy: specifies a template for supporting the user's navigation among the various categories and subcategories of a specific information object. In, **Fig. 4**, we can see two of the most common patterns for browsing the categories hierarchy of typical object information types.

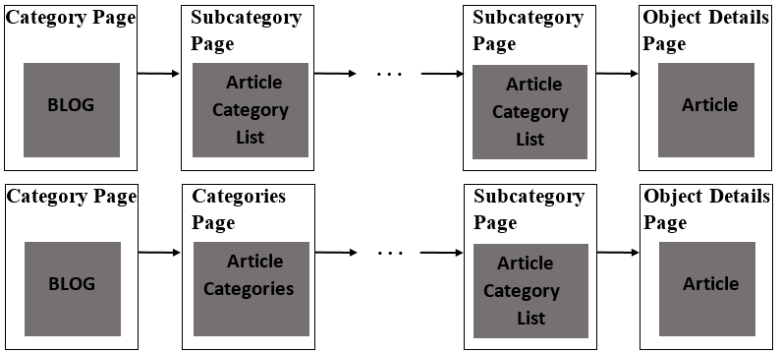

Fig. 4. The patterns for browsing the categories hierarchy of an information object type.

The top pattern is mostly used to browse the categories hierarchy of the courses whereas the other pattern is mainly used for browsing the educational programs, since the module Article Categories component provides a short description of each published list item (i.e. for each educational program). Similarly, we have also identify additional layout patterns such as the Search Results Page, the About Institution Page, the Information Object List Page, the Information Object Details Page, etc. The description of these patterns is available in [1]. Generally, these patterns support the consistent use of standard page structures, composed by a specific combination of Joomla! components and modules, used as templates for assisting designers in the predictability of user navigation. After all, consistency across pages and page design elements enforces a coherent design style improving the ease of use of a website while at the same time the users can enjoy a more pleasant user experience by reducing the number of unexpected variations in the page layout.

Navigation: this category includes patterns (such as navigation bars, menus, pagination, module tabs) for supporting user's navigation for locating content or accomplishing certain task. An example can be the use of a "fat" footer (**Fig. 5**) for providing users with a mechanism for quick access to specific sections of the website, bypassing the navigational structure. An example can be found on the website of the Graduate School of Arts and Sciences[2] (GSAS) in which the footer contains links to pages frequently used by users.

[2] Available at https://www.gsas.harvard.edu/

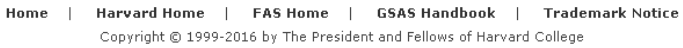

Fig. 5. An example of a fat footer.

Domain functionality: this category includes patterns that we identified to be used for supporting basic activities within the context of educational websites domain. For example, in **Fig. 6**, we can see two of the prevalent recurrent patterns that we identified for supporting the presentation of the degree programs to the users. In **Fig. 6**(a), there is a blog page displaying an overview of the various degree programs categories from which the user can navigate to another blog page providing information about the selected degree program category which includes a menu with links to article pages, corresponding to its subcategories, presenting the individual degree programs belonging to this specific category. In the other case **Fig. 6**(b), the second blog page with the menu has been replaced by two other pages, an article categories page providing information about the selected degree program category from which the user can navigate to article category list page enlisting all the subcategories of the selected degree program category. These are two pat-terns for supporting the user navigation to the hierarchy of categories and subcategories of the various degree programs.

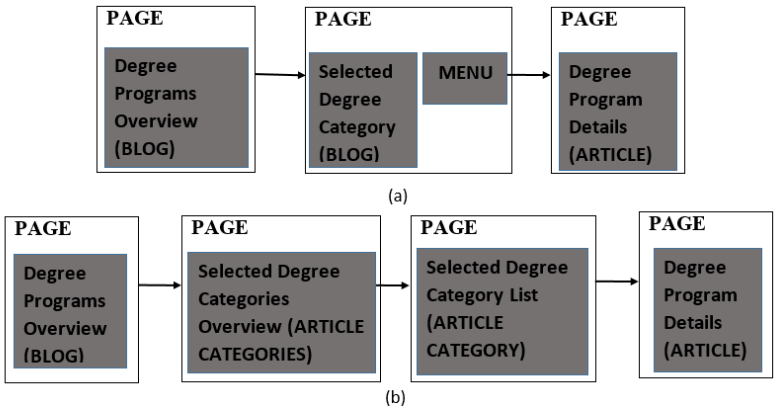

Fig. 6. An example of a domain functionality design pattern.

Miscellaneous: this category includes patterns that we have not found their main category yet. In this case, we have detected certain configurations of Joomla! component and modules occurring frequently in the hypertext design of educational websites but we can-not find a clear semantic connection with the domain's concepts.

The benefits of using the above domain-specific design patterns will be increased reuse in future designs. Furthermore, in this way, we can derive website design practices and design guidelines (i.e., the prevalent design patterns) for the domain of educational websites, and also identify commonalities and differences in the design of such kind of applications.

5 Conclusion and Future Work

In this paper, we have illustrated a model-driven approach for mining domain-specific design patterns. We focus on the domain of educational websites and present our preliminary results, i.e. the design patterns which we identified for this particular domain. 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 submit the patterns identified in their designs to the central repository. In the future, we plan to explore other application domains and thus, come up with useful design guidelines for building successful Joomla!-based websites.

6 References

1. CMS Patterns, (2015) Available at: http://alkistis.ceid.upatras.gr/research/modeling/patterns/ (Accessed: 22 February 2016).
2. Gkantouna V., Tsakalidis A., Tzimas G., Mining Interaction Patterns in the Design of Web Applications for Improving User Experience, ACM Hypertext (2016)
3. Yan, X., Han, J., 2002. gSpan: Graph-based substructure pattern mining. In ICDM '02, page 721, Washington,DC, USA, 2002. IEEE Computer Society
4. 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).
5. 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).
6. Joomla! Community Showcase website. Available at: http://community.joomla.org/showcase/sites/educational.html (Accessed: 22 February 2016).
7. Montero, S., Dıaz, P., Aedo, I.: A semantic representation for domain-specific patterns. In: Wiil, U.K. (ed.) MIS 2004. LNCS, vol. 3511, pp. 129–140. Springer, Heidelberg (2005)
8. Wolfgang Herzner, Gyorgy Csertan and Andras Balogh, Design Patterns for Domain-specific Application Modelling. Available at http://static.inf.mit.bme.hu/decoscd/papers/DECOS_paper_Design_Patterns_for_Domain_specific_Application.pdf
9. Saoussen Rekhis, Nadia Bouassida, Claude Duvallet, Rafik Bouaziz, Bruno Sadeg, 2010. A Process to derive Domain-Specific Patterns: Application to the Real Time Domain, Advances in Databases and Information Systems, pp. 475-489
10. Saoussen Rekhisa, Nadia Bouassidaa, Rafik Bouaziza, Claude Duvalletb, Bruno Sadegb, 2016. A new method for constructing and reusing domain specific design patterns: Application to RT domain, Journal of King Saud University - Computer and Information Sciences