algorithms

MDPI

Article

Mining Domain-Specific Design Patterns: A Case Study †

Vassiliki Gkantouna [1,*] and Giannis Tzimas [2]

[1] Department of Computer Engineering & Informatics, University of Patras, Patras 26504, Greece
[2] Computer & Informatics Engineering Department, Technological Educational Institute of Western Greece, Patras 26334, Greece; tzimas@teiwest.gr
* Correspondence: gkantoun@ceid.upatras.gr; Tel.: +30-2610-996-960

† This paper is an extended version of our paper published in Proceedings of Mining Humanistic Data Workshop (MHDW), 2016.

Academic Editors: Katia Lida Kermanidis, Christos Makris, Phivos Mylonas and Spyros Sioutas
Received: 16 November 2016; Accepted: 16 February 2017; Published: 21 February 2017

Abstract: Domain-specific design patterns provide developers with proven solutions to common design problems that arise, particularly in a target application domain, facilitating them to produce quality designs in the domain contexts. However, research in this area is not mature and there are no techniques to support their detection. Towards this end, we propose a methodology which, when applied on a collection of websites in a specific domain, facilitates the automated identification of domain-specific design patterns. The methodology automatically extracts the conceptual models of the websites, which are subsequently analyzed in terms of all of the reusable design fragments used in them for supporting common domain functionalities. At the conceptual level, we consider these fragments as recurrent patterns consisting of a configuration of front-end interface components that interrelate each other and interact with end-users to support certain functionality. By performing a pattern-based analysis of the models, we locate the occurrences of all the recurrent patterns in the various website designs which are then evaluated towards their consistent use. The detected patterns can be used as building blocks in future designs, assisting developers to produce consistent and quality designs in the target domain. To support our case, we present a case study for the educational domain.

Keywords: domain-specific design pattern; web mining; web design; design quality; content management system (CMS); conceptual modeling

1. Introduction

Design patterns [1] have emerged as a means to promote design reuse, providing designers with proven solutions to recurring design problems that can be reused in different contexts where the corresponding problem arises. By reusing such successful design solutions, developers can produce applications of high quality more rapidly, since they can rely on prior experiences and well-tested good practices. As a result, the development process of an application is accelerated and the development cost is reduced. At the same time, the adoption of design patterns can also bring many benefits, in terms of usability and overall application quality. When developers apply design patterns, they have the ability to enforce a coherent design style which can potentially result in more consistent and predictable designs. This makes it easier for the end-users to recognize typical patterns of interactions with the system for performing common tasks, thus improving the ease of use of an application. Furthermore, design patterns help improving the communication among interdisciplinary development teams, by providing them with a common design vocabulary to discuss the design alternatives and understand the various design decisions made throughout all phases of the development lifecycle.

While initially used mainly in the field of software engineering, over the last decades design patterns have also been enthusiastically embraced by the web engineering community for addressing common web design problems. A large number of web design patterns have been identified and organized into pattern catalogues [2–4], offering design guidelines to developers concerning several aspects of web design, e.g., navigation and user interface design. However, the available patterns are of general-purpose, i.e., too abstract and divorced from the context that an application is being developed, thus making it difficult for developers to understand the available context in which a pattern can be applied when the boundaries of its applicability are not clearly defined. Furthermore, the process of the pattern instantiation is a complex task, since it is difficult for developers, even for a certain application context, to determine in which part of the system a pattern can be used. In fact, in some cases, design patterns cannot be instantiated directly, due to natural constraints of the application context in which they are going to be used. As an example, consider the case of the popular carrousel pattern [5] used when developers need to display a large collection of items on a page, but there is not enough space. The carousel displays only a subset of the items at a time, arranged in a horizontal line where each item has a thumbnail image attached, thus, optimizing the screen space. Obviously, this pattern is a good design choice for domains in which the website content consists mainly of highly visual items, such as the various products in e-commerce websites. On the other hand, it can be a disaster in websites having as content text articles, PDF documents, etc. This raises the following questions: in what application domains can this pattern be used, except for the obvious e-commerce domain? Is it effective to use it on the domain of news sites? If yes, on which pages of a news site, and for what type of content, can this pattern be used? All of these questions cannot be answered by the pattern's definition in the available catalogues since they are expressed in natural language following a common structure of some typical elements, such as the pattern's name, the problem statement, the solution and its consequences, often resulting in prescriptive guidance which is vague and conflicting. As a result, developers have difficulties to effectively incorporate general-purpose patterns into their designs and, to make matters worse, they often attempt to force-fit them into use, causing serious design inconsistencies and other quality-related issues [6].

As a response to the problems of general-purpose design patterns, the web engineering community has also welcomed the advent of domain-specific design patterns, encapsulating design experience which is in alignment with the natural constraints of a particular application domain. In fact, domain-specific design patterns are design patterns defined particularly for the design of web applications in a specific application domain. They offer developers solutions to common design problems that arise particularly in the context of this specific domain. This makes it easier for developers to better understand the circumstances under which patterns can be used (i.e., the particular domain design problem they solve), thus facilitating them to effectively incorporate patterns into their designs, since they have more clear bounds of applicability. Moreover, they are specified in terms of domain-related objects, thus making it easier for developers to cope with the pattern instantiation process. To understand the main advantage of using domain-specific design patterns, consider the case of a domain-specific pattern that is based on the use of the previously described general-purpose carrousel pattern in the domain of news sites. The task of determining the context in which the carrousel pattern can be used in a news site is not simple since the content of these websites consists mainly of text articles. An example of a domain-specific pattern could be the one specifying the use of the carousel pattern on the pages of the websites corresponding to the news site editors' personal page for displaying their published books. The main difference now is that, in the case of the domain-specific pattern, the design problem addressed by the pattern is clearly defined, as well as the part of the website in which the pattern can be used (i.e., on the editors' personal pages). The rationale behind the concept of domain-specific design patterns is to capture reusable design fragments which can be used as design micro-architectures by developers for supporting common functionalities in the designs of websites in the considered domain. Contrary to general-purpose patterns, the number of the available domain-specific web design patterns is very limited mainly due to the absence of techniques

able to identify them. This happens because of the intrinsic complexity of these types of patterns, since they must encompass generality and variability for being able to be instantiated for various applications in a domain. As a result, there is a need for techniques able to facilitate the identification of domain-specific design patterns.

Towards this end, we present a methodology which, when applied on a collection of websites in a particular application domain, facilitates the automated identification of domain-specific design patterns. The methodology was originally introduced in [7]. In this paper, we present an important extension of our previous work in [7], which is based on the use of domain concepts in the process of identifying the reusable design solutions that can be considered as domain-specific patterns. The key idea is to analyze the designs of the various websites in order to detect all of the possible design commonalities among them that can be considered as partitions of the domain and, thus, can be used as building blocks for producing future designs. Based on this, the proposed methodology analyzes the conceptual models of the collected websites in terms of the reusable design fragments repeatedly used in them for supporting common functionalities within the domain context. At the conceptual level, we consider these fragments as recurrent patterns occurring in the conceptual models of the websites, consisting of a configuration of front-end interface components that interrelate with each other and interact with the end-user to achieve a certain functionality, i.e., support user navigation to certain information objects. 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). First, we create a collection of websites in a target application domain and capture their designs by automatically extracting their conceptual models. Subsequently, to identify the possible design commonalities among the website designs, we perform a pattern-based analysis of the recovered models which results into the identification of all the incorporated recurrent patterns occurring at hypertext level. To verify that the identified patterns (i.e., hypertext compositions) are actually used for supporting common functionality, we additionally examine the semantic aspect of their occurrences in the various websites, for determining whether the recurrence of the design elements at the hypertext design goes with a recurrence at the data level, i.e., the type of content they deliver to the end-users, which is captured by means of domain concepts. This is achieved by applying a semantic similarity measurement technique among the content of the pages involved in a pattern among its various occurrences on the websites which computes the degree of their semantic similarity and detects the common semantic concepts to which they refer. The computed common semantic concepts are considered as domain concepts representing the common information object types frequently appearing on the pages of the educational websites in the collection. We locate the pattern occurrences which are highly semantically related and deliver the same configuration of domain concepts to end-users, implying possible recurrence at the data level and, thus, increasing the possibility of performing common functionality. This way, the various domain functionalities are also expressed as configurations of domain concepts. Finally, we calculate evaluation metrics for the detected patterns, revealing whether they are used consistently throughout the models of the various websites, and store them on a central pattern repository. We argue that the detected patterns can assist developers produce more consistent and quality designs in the context of the target application domain, since they offer them design guidelines derived from the best design practices used in the considered domain. To exemplify the concepts behind our approach, we present the results of a case study in which the methodology has been applied in the domain of educational websites.

In order to automate the process of analyzing the design of a website, 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 proposed methodology is accompanied by a tool support available in [8], allowing developers to apply it on websites developed

by using the Joomla! [9] and Drupal [10] CMS platforms. Due to space limitations, in this work we present the methodology for the case of Joomla!-based websites. The remaininder of this paper is organized as follows: Section 2 provides an overview of the related work. Section 3 presents, in detail, the methodology for mining the domain-specific design patterns, while Section 4 presents the case study for the domain of educational websites. Finally, Section 5 discusses conclusions and future work.

2. Related Work

The main objective of this work is to facilitate the detection of domain-specific web design patterns. Unfortunately, research in the field of domain-specific design patterns is very limited and there is an absence of techniques for the patterns identification.

By searching the literature for studies related to domain-specific design patterns in the web engineering field, one can find out that there are only very few domain-specific pattern catalogues, such as the ones in [11,12]. These catalogues include design guidelines for designing websites in a specific application domain, e.g., e-commerce, news sites, travel sites, etc., expressed in natural language. The available guidelines have been derived empirically by experienced designers after studying and manually analyzing the designs of successful websites in a target application domain. Some other works focused on the automated detection of design patterns in the conceptual model of data-intensive web applications can be found in [13,14]. In [13], authors present the Web Quality Analyzer (WQA) which is able to automatically analyze the conceptual schemas of WebML-based web applications with the aim of identifying the occurrences of a predefined set of WebML design patterns specifying typical ways of content publishing and management. By calculating metrics about the coherent use of the identified occurrences of design patterns throughout the application schema, the WQA allows designers to automatically monitor the design consistency of the applications. In [14], authors present an approach for supporting the automatic identification of web interaction design patterns implemented in a Web application. The approach is based on reverse engineering techniques aiming to search the code of a website's pages to detect those features that characterize a pattern. The authors have selected the characteristic features for a number of patterns by analyzing different typical pattern implementations. Such features include UI model fragments (e.g., forms, tables, etc.) and lexical terms (e.g., login, poll, etc.). To identify a given pattern, the approach requires that its characteristic features are specified a priori. However, these approaches are focused on general-purpose web design patterns. Only few approaches have been proposed to assist developers identify domain-specific patterns, but they all refer to the field of software engineering.

In [15,16], the authors present a method for constructing and reusing domain-specific design patterns, applicable to the domain of real-time applications. The proposed method guides developers to construct the patterns by applying a set of comparison criteria, which are defined by experts after manually performing domain analysis, on various applications in the real-time domain in order to identify the possible commonalities and differences between the applications models and, thus, derive the fundamental, optional, and extensible elements of a pattern. This method is based on the UML-profile language. However, this method is applicable merely for the real-time domain and it is possible that it can result in patterns which their instantiation may not correspond to reality.

In [17], the authors highlight the need to introduce a formalism to describe domain-specific patterns accurately and to allow a rigorous reasoning process to assist developers retrieve the most appropriate pattern for handling a specific instantiation of a design problem in a particular application domain. To this end, they propose a semantic representation for domain-specific design patterns which can be used as an underlying armature for complementing the informal textual pattern description by means of semantic annotations. More specifically, they propose an ontology-based approach that allows designers to enrich a set of predefined domain-specific patterns with semantic annotations using domain concepts and terms defined by domain experts. The pattern format and the domain knowledge concerning the pattern are kept separately in a representational vocabulary using ontologies. This representation framework keeps the pattern template and the domain knowledge separate, since

each domain, depending on its needs, uses more or fewer fields to describe its patterns and the later ontology changes according to the domain knowledge that the pattern captures. In [18], authors present a UML-based language for specifying domain-specific design patterns. They present a pattern specification notation called the Role-Based Metamodeling Language (RBML) and shows how it can be used to express domain-specific patterns. An RBML pattern defines a domain-specific sublanguage of the Unified Modeling Language (UML). Developers can use the sublanguage to create UML diagrams for applications in the domain addressed by the pattern.

All of the aforementioned approaches assume the existence of a predefined set of domain-specific design patterns which are manually devised by domain experts after analyzing the designs of successful websites in a domain. The key difference of our approach is that it is not based on this assumption. Our aim is not to devise domain-specific design patterns, but rather to discover and mine these patterns by detecting their recurrent use in a set of real domain-specific websites. To this end, we provide a methodology for the automated identification of the recurrent domain-specific design patterns lying in the designs of a collection of websites in a specific application domain. 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. The identified patterns can empower less-experienced designers to produce designs of high quality in a specific application domain on the grounds that these patterns provide them with well-tested solutions, guiding them on how to organize the various sections of a domain-specific website.

3. The Methodology

In this section, we present the methodology for mining domain-specific design patterns from a set of concrete designs of various websites in a target application domain. It is comprised by three main phases (Figure 1). First, we utilize a web crawler to create a collection of websites, and then we extract their conceptual models at hypertext level. Secondly, the recovered models are submitted to a pattern-based analysis with the aim of (i) identifying the occurrences of all the recurrent patterns lying within them; and (ii) verifying which of them can be considered as candidate domain-specific design patterns (i.e., supporting the realization of common functionalities). Finally, we calculate a set of evaluation metrics to assess if they are used consistently throughout the models of the websites. The identified patterns are stored in a repository available at [8].

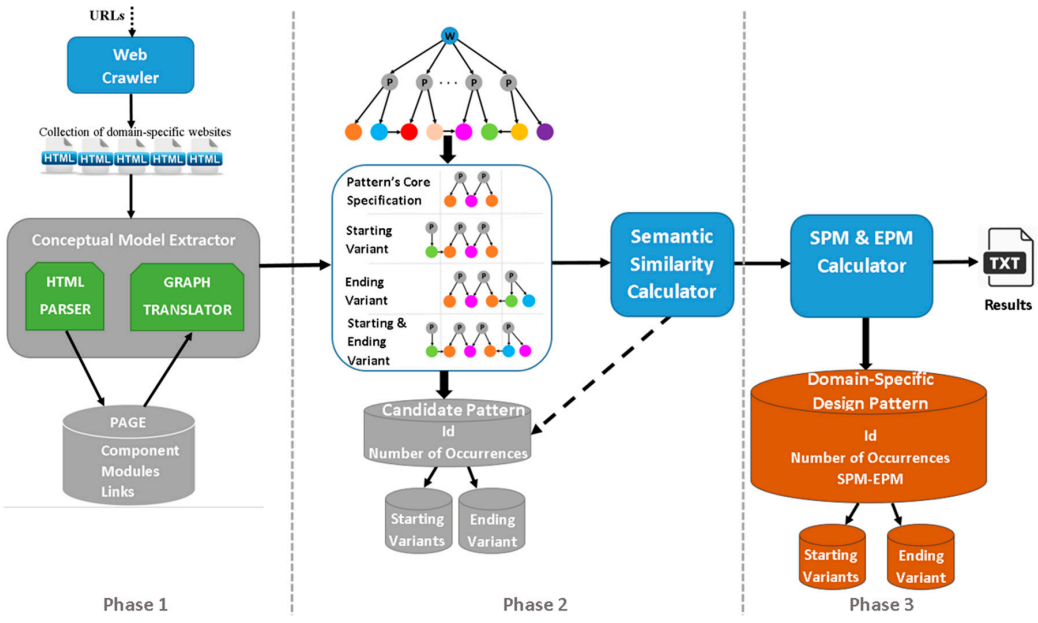


Figure 1. The three main phases of the methodology.

3.1. Phase 1: Extracting the Conceptual Models of the Websites

In the first phase, we utilize a web crawler which crawls all of the pages of a set of websites in a target application domain and locally stores them on the user's computer, thus creating a collection of domain-specific websites. Then, we utilize the "Conceptual Model Extractor" tool to capture their designs by extracting their conceptual models. At the hypertext level, the conceptual model of a website 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 the "Conceptual Model Extractor" tool, as depicted in the first phase of Figure 1. This tool parses all of the locally-stored HTML pages of every website in the collection and identifies 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 a specific HTML class attribute value (i.e.,) which characterizes them and specifies the exact type of the component-module they represent. The complete list with all the categories of Joomla! components and modules, along with their characteristic HTML class attribute values are available in [8]. 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, Figure 2a presents the Joomla! design elements identified within a page of the MMLAB educational website [19], consisting of the "Article" component and a set of modules such as menus, footer, etc. Once this is done for all of the pages of the websites, we manage to capture their composition and navigation, i.e., to extract their conceptual models. Then, the next step is to represent them as graphs, required for facilitating the detection of the recurrent patterns among them in the next phase.

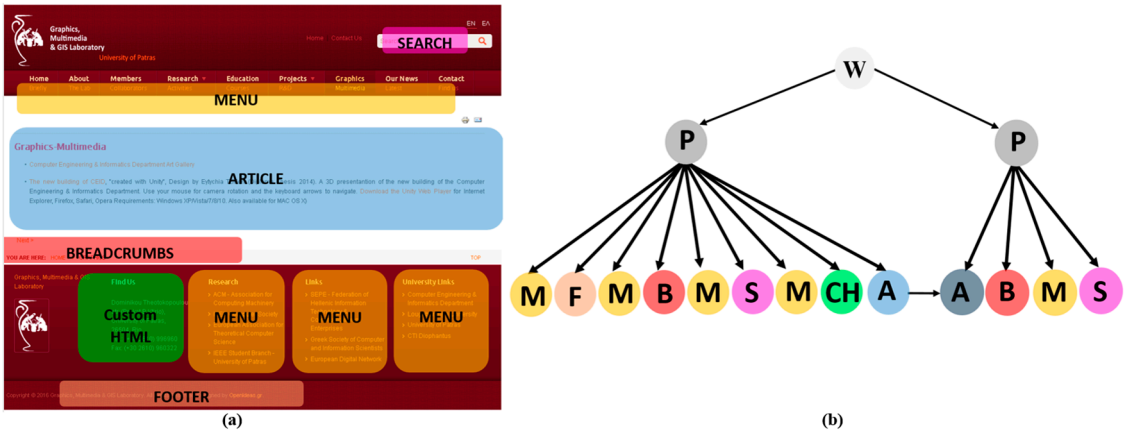


Figure 2. (**a**) The organization of a page in terms of Joomla! design elements (component and modules); and (**b**) The equivalent graph representation of the page's hypertext.

We define the conceptual model of a website as a directed graph of the form G (V, E, f_V, f_E), comprising a set of nodes V, a set of edges E, a node labelling function f_V: V $\rightarrow$ Σ_V and an edge labelling function f_E: E $\rightarrow$ Σ_E. The function f_V assigns labels to the nodes in V from the alphabet Σ_V which includes all of the different types of the Joomla! components and modules available in [8]. Similarly, function f_E assign labels to the edges in E from the alphabet Σ_E = {W_P, P_M, P_C, M_P, C_P, M_C}. The label W_P denotes the containment of a page in a website, the labels P_M and P_C denotes the containment of modules and components, respectively, within a page, the labels M_P and C_P denotes the link from modules and components respectively to a page, and finally the label M_C denotes the link between modules and/or components. To represent the conceptual models of the websites as directed graphs, we have developed the "Graph Translator" tool. For example, Figure 2b presents an instance of a conceptual model graph and particularly the equivalent graph representation of the page depicted in Figure 2a. These graphs are going to be the input for the graph mining algorithm of the next phase.

3.2. Phase 2: Mining the Candidate Domain-Specific Design Patterns

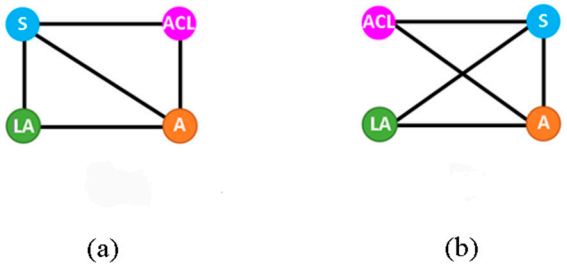
The main objective of this phase is to detect the possible design commonalities lying among the previously-extracted conceptual models, which can probably indicate the existence of domain-specific design patterns. To this end, we analyze these models in terms of the reusable design fragments used recurrently in them for supporting common functionality. These fragments are considered as recurrent patterns consisting of a configuration of Joomla! components and modules which, when located in a particular layout, may deliver a certain set of information objects to end-users in the context of supporting a certain task or domain functionality. To capture these fragments, first we perform a pattern-based analysis of the recovered models in order to identify all the recurrent patterns occurring in them at hypertext design. Then, we inspect their occurrences in the various websites of the collection to examine if there is a coexisting recurrence at data level, i.e., if the identified patterns are used to deliver the same set of information objects types, specified in terms of domain concepts, to end-users (Figure 1, Phase 2).

3.2.1. Mining the Recurrent Patterns at Hypertext Level

To locate the recurrent patterns (their core specifications along with their starting and termination variants) occurring among the set of conceptual models, we have reduced the problem of their identification into the domain of graph mining, and particularly to the subgraph isomorphism problem. The latter is synopsized in its general form into finding whether the isomorphic image of a subgraph exists in a larger graph. An example of an isomorphic image of a subgraph is depicted in Figure 3. The subgraph in Figure 3b is isomorphic to the subgraph in Figure 3a. As we can see, despite the different configuration of nodes in the two subgraphs, the edges connecting the nodes of the same color remain the same. The table in Figure 3c contains some sequences of the nodes that are connected in the subgraphs of Figure 3a,b. For example, a sequence can start from node S, from which one can navigate to node article category list (ACL) and then navigate to node A, and so on. By observing the node sequences, one can notice that they can actually reveal the recurrent patterns occurring 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 attempt to identify the recurrent patterns lying in the set of the recovered conceptual models, by locating all the isomorphic subgraphs within their equivalent graph representations. To achieve this, we employ a graph mining algorithm, namely gSpan [20], which identifies the occurrences of all the recurrent isomorphic subgraph patterns within the conceptual model graphs. More specifically, gSpan addresses the problem of frequent subgraph mining. Due to space limitations, we cannot provide a detailed description of how the gSpan works, but we present an explanatory overview. Intuitively, gSpan traverses the set G of the conceptual model graphs and

finds all the smaller subgraphs g in G that occur frequently. A subgraph g is frequent if its occurrence frequency in G, denoted as support(g), is no less than a minimum support threshold (minSup). In a more formal definition, the problem of frequent subgraph mining is to find any subgraph g into G so that support(g) $\geq$ minSup. When looking for the occurrences of a subgraph in G, the algorithm encounters except for its identical occurrences, its isomorphic images, too. In this way, the conceptual model graphs are analyzed in terms of the frequent subgraphs (representing recurrent patterns) occurring among them.



(a) (b)

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

(c)

Figure 3. (**a**,**b**) An example of two isomorphic graphs; (**c**) The table of node sequences connected in the isomorphic graphs.

To apply the gSpan algorithm on the set of conceptual model graphs, we have used the Parsemis project [21], which supports an implementation of the gSpan algorithm within a graphical environment for visualizing the identified frequent subgraphs. An example can be found in Figure 4 which presents an identified subgraph pattern representing one of the most typical ways of browsing large hierarchical content structures, i.e., the hierarchy of categories and subcategories of an information object, used in the designs of many educational websites. The pattern consists of a composition of Joomla! hypertext design elements (components and modules) involving four different pages of an educational website. More specifically, it consists of an article page publishing content about an information object, from which users can navigate to a second page which is based on the ACL component and allows users to quickly scan a list of information items. By selecting a certain list item, users can navigate to a third page of the same component type (ACL), allowing them to have access to a second list of information items. Finally, by selecting a specific list item, users can have access to an article page presenting content about the selected list item.

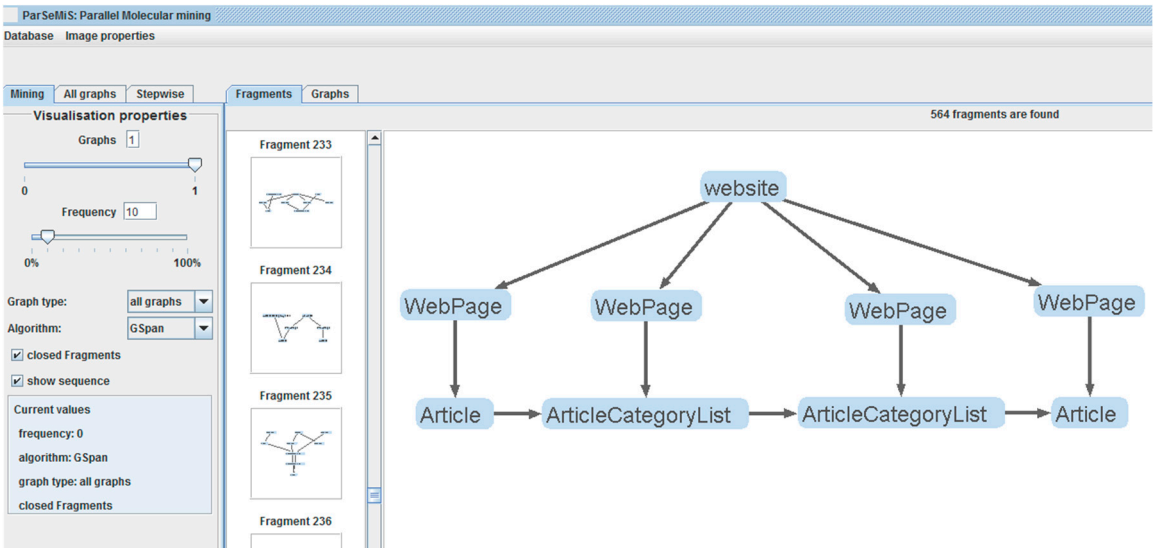


Figure 4. A frequent subgraph detected in the conceptual models of educational websites.

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 set of graphs. We process this file in a way similar with the one presented in the example of Figure 3c (based on the sequences of the connected nodes in the subgraphs) and identify the core specifications and the starting and ending variants of each pattern.

3.2.2. Inspecting Data Level: Identifying Patterns Supporting Common Functionality

In order to verify that the identified recurrent patterns are used in the various websites for supporting common domain functionality (so that they can be considered as candidate domain-specific patterns), we additionally have to examine whether the recurrence of the design elements at the hypertext level of the websites goes with a coexisting recurrence at their data level. To achieve this, it is necessary to inspect all of the occurrences of these patterns on the websites in the collection in order to examine whether a recurrent hypertext pattern, i.e., a certain configuration of Joomla! components and modules, is used particularly (intentionally) for delivering a certain set of information objects types to the end-users and, thus, performing a certain domain functionality.

To achieve this, we apply a WordNet-based semantic similarity measurement technique [22] on the contents published by the pages involved in a pattern among its occurrences, which computes the degree of their semantic similarity and detects the common semantic concepts (i.e., information objects) to which they refer. The content published by the various components and modules of a page 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, ideally, 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 common functionality. 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 similarity 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 among the various websites in the collection 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 supporting common functionality, which can be specified by capturing the common semantic concepts to which these occurrences refer.

To compute the semantic similarity of the published contents between two pattern occurrences, we have defined two metrics, the "SemSimScore" and the "AverageSemSimScore". On the grounds that the main content of a page, indicative of its semantics, is published by the page's underlying component, the "SemSimScore" metric addresses the semantic similarity measurement of the content published by the Joomla! components that are involved 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 pattern between its occurrences. Then, the "AverageSemSimScore" computes the average value of the individual "SemSimScore" values between the pattern occurrences. For every computation of the "SemSimScore" metric among two given contents, we also detect the common semantic concept to which these contents refer and we consider it as a domain concept representing the common information object that they deliver to end-users. This way, we can express the common functionality supported by the two pattern occurrences as the configuration that is formed by the individual domain concepts computed for these occurrences. By computing the pairwise computations among all of the occurrences of a given pattern, we can determine which is the most recurrent configuration of domain concepts among its

occurrences and consider it as the common domain functionality supported by the pattern. In Table 1, we can see an example of computing the semantic similarity between the occurrences of the pattern presented in Figure 4 for browsing the hierarchy of categories and subcategories of information items in educational websites. To verify that this pattern is actually used for supporting common functionality, we inspect its occurrences to examine if there is also a recurrence at the content it delivers to the end-users. In Table 1, we can see three occurrences of the pattern Occ.1, Occ.2, and Occ.3 in two different educational websites. 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 75%, which means that they are semantically close. Furthermore, the common functionality supported by these two pattern occurrences is to support user navigation to the following configuration of domain concepts: {University, Departments, Staff, Professor}. Similarly, the AverageSemSimScore for Occ.1 and Occ.3 is 58%, implying that the content published in these two occurrences is not semantically highly related and, thus, that there is not a recurrence at the data level. By computing all the pairwise computations among the occurrences of this pattern in the various websites, we can identify that the previous identified configuration of domain concepts is one of the most recurrent ones and, thus, we can assume that the domain functionality supported by this pattern is to allow user navigation to this specific set of domain concepts.

Table 1. Measuring the semantic similarity among pattern's occurrences in the various websites of the collection.

PATTERN	MENU [Module]	ARTICLE [Component]	CATEGORY LIST [Component]	CATEGORY LIST [Component]	ARTICLE [Component]
Occ.1	Top Menu	AUTH University	Departments in AUTH	Academic staff—Department of Informatics	Professor Mr. Papadopoulos
Occ.2	Main Menu	Piraeus University	Undergraduate Studies—Departments	Academic Staff—Department of Electronics Engineering	Professor Mrs. Rammou
SemSimScore Occ.1–2		85%	75%	70%	70%
Common Semantic Concept		**University**	**Departments**	**Staff**	**Professor**
AverageSemSimScore Occ1.–Occ.2					75%
Occ.3	Top Menu	AUTH University	Departments in AUTH	Undergraduate Studies	Databases—Course Description
SemSimScore Occ.1-3		100%	100%	10%	23%
Common Semantic Concept		**University**	**Departments**	**Education**	**Course**
AverageSemSimScore Occ1.–Occ.3					58%

In this way, we can obtain a safe estimation about the recurrence at the organization of information objects in the pages of the websites 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 AverageSemSimScore over 70%.

The use of domain concepts for identifying the reusable design solutions lying among the designs of the various websites in the collection is the main contribution of this work to the methodology presented in [7]. Previously in [7], we solely used the "SemSimScore" and the "AverageSemSimScore" metrics for measuring whether the contents of the pages involved in a pattern among its occurrences are semantically close and, thus, assume that they possibly deliver the same information object types to end-users, i.e., support common functionality. However, we were not able to capture which of these are information object types so that we can understand the purpose behind the use of the patterns in the

context of the target domain. In other words, we could capture cases in which the use of a pattern in the designs of the various websites was implying common functionality, but we could not express this functionality in terms of domain-specific objects so that to make it easier for developers to understand the context in which the identified patterns can be applied. To achieve this in [7], we had to manually inspect the pattern occurrences in order to be able to find out for which type of content (i.e., information objects) they are used to deliver to end-users in the educational domain.

To overcome this problem, in the current work, we came up with the idea of the domain concepts allowing us to automatically identify the common information objects to which the pattern refers among its various occurrences, and thus understand the domain functionality supported by the pattern. This way, we can express the various domain functionalities as a set of domain concepts reflecting the common information objects of the domain.

3.3. Evaluation of Pattern Variants Consistent Use

In this final step, we focus on evaluating the consistent use of the identified patterns throughout the conceptual models of the various websites. Patterns which are consistently used probably result in high design quality, facilitating end-users to identify typical patterns of interactions with the system for performing common functionalities. This results in predictable 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 repository are used consistently throughout the set of conceptual models. 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 conceptual models. SPM is defined as follows, and EPM is defined in an analogous way:

$$SPM = \sigma^2 / \sigma^2_{BC} \tag{1}$$

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

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

where p_i is the percentage of occurrences for the i-th pattern variant. σ^2_{BC} is, instead, the best case variance and it is calculated by Equation (2), assuming that only one variant has been coherently used throughout the models. More details about the metric's definition can be found in [13]. 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).

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

EPM-SPM Range	Measurement Scale Value
$0 \leq$ SPM < 0.2	Insufficient
$0.2 \leq$ SPM < 0.4	Weak
$0.4 \leq$ SPM < 0.6	Discrete
$0.6 \leq$ SPM < 0.8	Good
$0.8 \leq$ SPM ≤ 1	Optimum

4. The Case Study

In the context of this work, we have focused on the domain of educational websites and, thus, we have applied the proposed methodology on a collection of websites in this domain. This resulted in the identification of a set of domain-specific design patterns specifically tailored for the educational domain, which are stored in a central pattern repository available at [8]. Here, we present a number

of these patterns in order to illustrate the potential of our approach and particularly the way in which the detected patterns can help developers during the design of educational websites. 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 educational websites derived from the official Joomla! Community Showcase website catalogue. The list of the websites that we have included in our dataset is available at [8]. In the following sections, we provide a short description of the educational domain and some examples of the patterns that we have identified for this area.

4.1. Domain Description

The educational websites have an increasingly important role to the admissions and marketing practices of colleges and universities, due to their ability to rapidly communicate a significant amount of content to a vast audience. They tend to be large, complex, and content-heavy websites that serve diverse audiences and, thus, the organization of their navigation and content can be a difficult task. In fact, educational websites must be designed in such a way that it enables end-users to recognize typical navigational patterns for quickly and effectively locating the information that they are looking for.

In the context of educational websites, the most common types of end-users include prospective, current, and international students along with the institution's staff and faculty, as well as alumni, donors, etc. Furthermore, the most common types of information objects appearing on the pages of educational websites include studies, courses, research, publications, admissions, academic staff, university campus, etc. The patterns that we have detected by applying the methodology on a collection of educational websites represent the navigational architectures that are commonly used in their designs for supporting various user tasks.

4.2. Domain-Specific Design Patterns for Educational Websites

In this section, we present some of the domain-specific design patterns that we have identified in this case study. We have classified these patterns into three main categories based on their common design characteristics concerning their layout, navigation structure, and functionality. Examples of patterns included in these categories along with a short category description are provided below.

Layout Category: this category includes patterns that provide developers with standard compositions of Joomla! components and modules which are widely used in educational websites for browsing the various information objects appearing in educational websites. Additionally, in many cases, these compositions specify page templates, capturing the most common ways of organizing the various Joomla! front-end interface elements in order to form a page. For example, Figure 5 presents two of the most commonly used templates in the website collection for specifying the layout of the homepage.

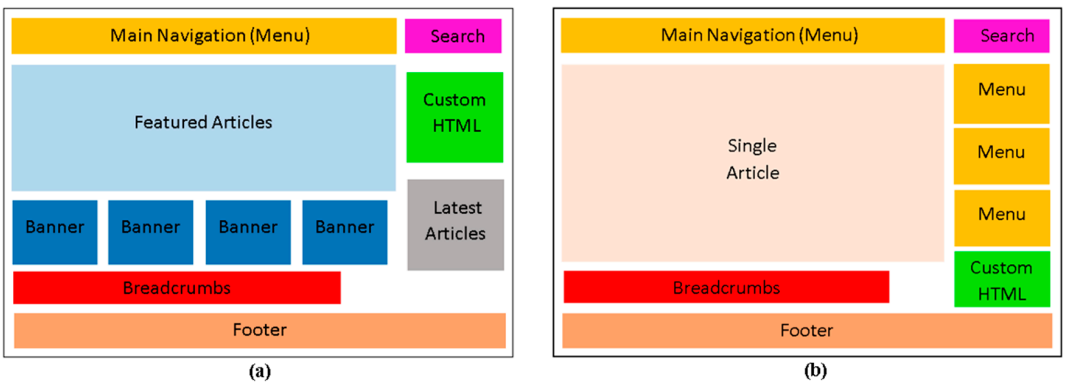


Figure 5. Templates for the homepage of educational websites. (**a**) A template for publishing the main content of the homepage in blog layout format; (**b**) A template for publishing the main content of the homepage in the form of an article.

The first template in Figure 5a consists of the featured articles component for publishing the main content of the homepage in the form of a blog presenting an overview of a number of articles, and a set of modules, such as banners, menu, latest articles module, etc., for supporting the navigation to other pages of the website. As you can see, except for the main navigation supported by the top menu, the use of banners on the homepage is a very popular way for supporting common user tasks, such as to provide direct navigation to pages which are of common interest, e.g., the available undergraduate and postgraduate studies in a university website, etc. Furthermore, many educational websites include custom HTML modules in their homepage, usually for displaying logos acting as links to external websites and, additionally, a latest articles modules for allowing users to quickly browse the most recently published articles. The second page template in Figure 5b consists of an article page usually displaying a welcome message to visitors and a number of modules, such as custom HTML, menus, etc., for accessing other website pages. The main purpose of both templates is to support the efficient navigation of users from the homepage to other pages in order to easily and rapidly locate the information they are looking for, since the homepage is the starting point for navigating throughout the entire website. The key difference of the second template is that, instead of using banners for direct navigation to the pages publishing important information, it makes use of extra menus. These menus group sets of pages, which are at the same hierarchical level of the website hierarchy, allow users to access them by following the corresponding link. This template is used mainly in websites with deep content hierarchical structures. Moreover, the use of breadcrumbs in this template is very crucial (showing a link for each level of the website, from the homepage to the current page) since it helps users orient themselves and understand the website structure.

Another example of a template involving more than one page is depicted in Figure 6 which presents a commonly used pattern for supporting the user's navigation among the hierarchy of categories and subcategories of information objects. It consists of four pages: the first page presents content about an information item, from which users can navigate to a second page displaying a list of its categories. Then, by selecting a list item users can have access to a third page of the same type displaying a list with the subcategories of the selected category and, finally, by selecting a list item, users can navigate to a fourth article page displaying information about the selected subcategory. Both of the pages in Figure 6 that present a list with the categories and subcategories include a most read articles module in order to allow users navigate to the most frequently visited pages, shortening the navigation path to them and acting as a mechanism for navigating backward and forward through the website pages, regardless of the hierarchical level to which these pages belong.

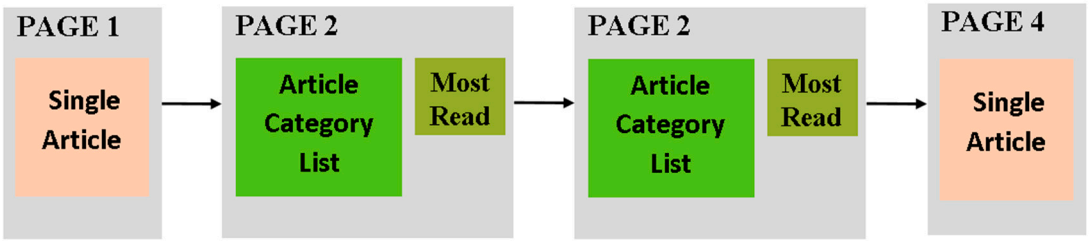


Figure 6. A typical pattern for browsing the hierarchy of categories and subcategories of information objects.

Generally speaking, the patterns of this category promote the consistent use of standard page structures used as templates for assisting designers in the predictability of user navigation. After all, consistency across website 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 intuitive user experience by reducing the number of unexpected variations in the page layout.

Navigation Category: this category includes patterns which correspond to reusable navigation structures used in the various website designs for providing access to various sets of domain concepts, i.e., information objects. For example, Figure 7a presents the most common way for browsing

the various types of end-users (i.e., students, alumni, and parents) which is realized with the use of a menu having links to the pages of the corresponding user category. Except for the commonly used types of end-users, we have also identified a pattern variant, including a link to a page publishing information about the visitor user type, occurring in a very small number of websites in the collection. Similarly, Figure 7b presents a commonly used pattern on the home page of educational websites for supporting the navigation to the pages publishing information about the undergraduate and postgraduate studies and campus life. This is realized by using four banners on the home page which navigate users to the corresponding pages. We have also identified a pattern variant including a link to a page publishing content about the summer school which is available on some of the collected websites.

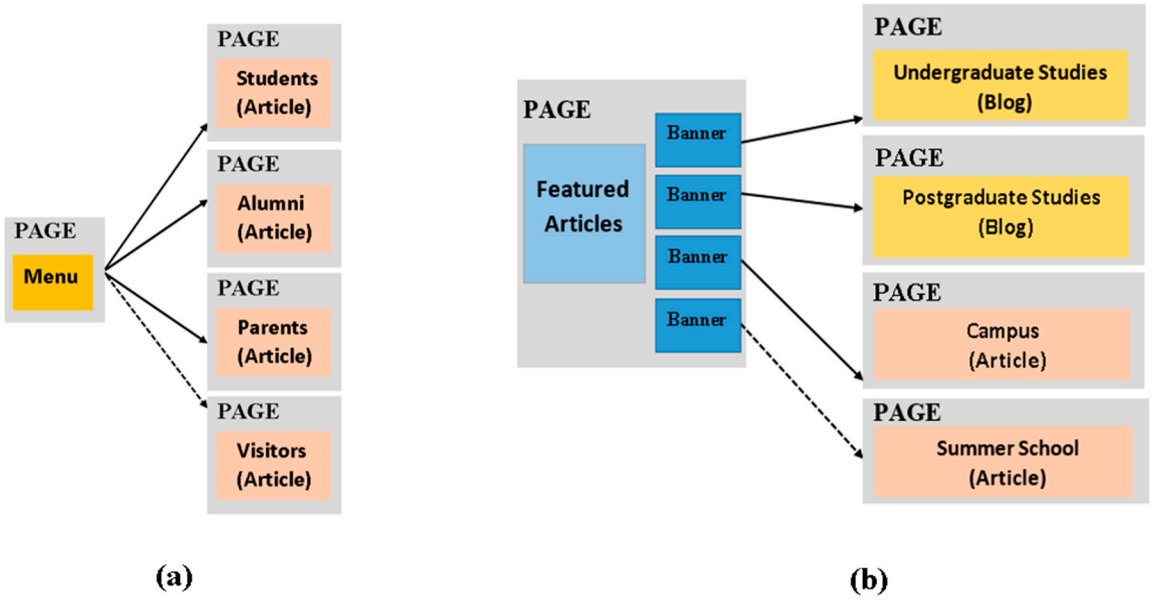


Figure 7. Reusable navigation structures for educational websites.

Another example is depicted in Figure 8 presenting the use of a "fat" footer for providing users with a mechanism for quick access to specific sections of the website, bypassing the navigational structure. An occurrence of this pattern can be found on the website of the Graduate School of Arts and Sciences [23] in which the footer contains links to pages which are frequently used by users.

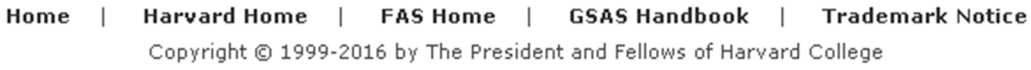


Figure 8. An example of a fat footer.

To understand the importance of using domain concepts, I should mention that in our previous work in [7] we could not automatically identify the concept to which the pages in the example of Figures 7 and 9 refer. We could only identify the recurrence occurring at the hypertext level, i.e., a menu providing links to four other website pages. In this work, we can capture these concepts by locating the domain concepts computed in Section 3.2.2 among the pattern's occurrences. It must be clear that, except for the organization of the hypertext elements, we are now able to provide end-users with design guidelines on how to organize the content (expressed by the appropriate domain concepts) on the website pages.

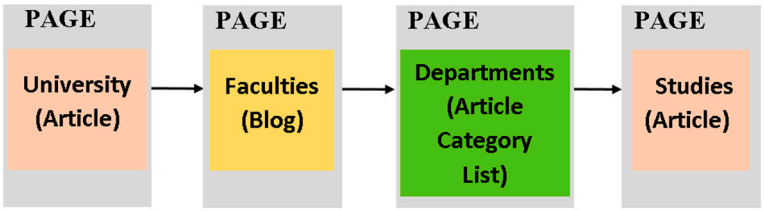


Figure 9. A pattern for supporting access to the available studies of a specific department of a certain faculty in a university website.

Domain functionality: this category includes patterns that we have identified to be used for supporting common user browsing activities within the context of educational websites, i.e., to provide end-users access to common sets of information objects types. For example, Figure 9 presents a pattern consisting of four pages specifying their organization for allowing users to access the following set of information objects: {University, Faculties, Departments, Studies}. In other words, the navigation purpose of this pattern is to allow users browsing from the university page to another page publishing a list of all the available faculties in the university, from which users can navigate to another page publishing a list of all the departments in a specific faculty, from which users can finally navigate to a page publishing information about the available studies in a specific department. In the patterns of this category, except for their structural and navigational aspect, we focus mainly on their semantic aspect.

The main benefit of using the identified domain-specific design patterns is the increased design reuse in future designs of educational websites. Developers can rely on commonly used design practices for the educational websites domain and have access to design guidelines guiding them on how to organize the content and navigation of their educational websites.

5. Conclusions and Future Work

In this paper, we have presented a model-driven approach for mining domain-specific design patterns from a set of concrete website designs in a target application domain. The key concept is to analyze the designs of various websites in a particular domain in order to detect the possible design commonalities among them which represent partitions of the domain implying the existence of domain-specific design patterns. Based on this, the proposed methodology analyzes the conceptual models of the collected websites in terms of the reusable design fragments used repeatedly in them for supporting common functionalities within the domain context. These fragments can be used as building blocks for producing future designs. We focus on the domain of educational websites and present the domain-specific design patterns that we have identified for this domain. Most of the work presented here can be also applied to websites built by using other CMSs, such as the Drupal [10] and the WordPress [24] platforms, with slight, straightforward modifications. These modifications concern mainly the adaptation of the web crawler that we have used in Phase I so that it can be able to recognize the structural and navigational design elements into the HTML code of websites built on other CMSs and, thus, extract their conceptual model.

Once the conceptual model of a website is extracted, the two other phases of the methodology remain the same. By applying the methodology on a particular application domain, developers can gain important information regarding its design specificities and have access to a set of consistently used methods of organizing and displaying information to end-users. In the future, we plan to explore more application domains, and, thus, to come up with useful design guidelines for building successful Joomla!-based domain-specific websites.

Author Contributions: Vassiliki Gkantouna and Giannis Tzimas participated in the concept, the design and the implementation of the methodology. They also performed the experiments and prepared the manuscript in order to present the identified domain-specific design patterns.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison-Wesley: Reading, MA, USA, 1995.
2. Yahoo Design Pattern Library. Available online: https://developer.yahoo.com/ypatterns (accessed on 15 November 2016).
3. Patterns Catalog. Available online: http://hillside.net/patterns/patterns-catalog (accessed on 15 November 2016).
4. Hypermedia Design Patterns Repository. Available online: http://wiki.c2.com/?HypermediaDesignPatternsRepository (accessed on 15 November 2016).
5. Patterns in Interaction Design, the Carrousel Pattern. Available online: http://welie.com/patterns/showPattern.php?patternID=carrousel (accessed on 23 January 2017).
6. Arango, R.G. Prieto-Diaz, Introduction and overview: Domain analysis concepts and research directions. In *Domain Analysis and Software Systems Modeling*; Prieto-Diaz, R., Arango, G., Eds.; IEEE Press: New York, NY, USA, 1991; pp. 9–32.
7. Gkantouna, V.; Tzimas, V.; Tampakas, B.; Tsaknakis, J. Mining Domain-Specific Design Patterns. In Proceedings of the AIAI, Thessaloniki, Greece, 16–18 September 2016.
8. Domain-Specific Patterns for CMS. Available online: http://alkistis.ceid.upatras.gr/research/modeling/domainspecificpatterns (accessed on 15 November 2016).
9. Joomla! CMS Website. Available online: http://community.joomla.org (accessed on 15 November 2016).
10. Drupal CMS Website. Available online: https://www.drupal.org/ (accessed on 15 November 2016).
11. A Pattern Library for Interaction Design. Available online: http://www.welie.com/patterns (accessed on 15 November 2016).
12. UI Patterns. Available online: http://ui-patterns.com/patterns/miscellaneous/list (accessed on 15 November 2016).
13. Fraternali, P.; Matera, M.; Maurino, A. WQA: An XSL Framework for Analyzing the Quality of Web Applications. In Proceedings of the 2nd International Workshop on Web-Oriented Software Technologies—IWWOST'02, Malaga, Spain, 10–14 June 2002.
14. Lucca, G.A.; Fasolino, A.R.; Tramontana, P. Recovering interaction design patterns in web applications. In Proceedings of the Ninth European Conference on Software Maintenance and Reengineering, Manchester, UK, 21–23 March 2005.
15. Rekhisa, S.; Bouassidaa, N.; Bouaziza, R.; Duvalletb, C.; Sadegb, B. A new method for constructing and reusing domain specific design patterns: Application to RT domain. *J. King Saud Univ. Comput. Inf. Sci.* **2016**. [CrossRef]
16. Rekhis, S.; Bouassida, N.; Duvallet, C.; Bouaziz, R.; Sadeg, B. A Process to derive Domain-Specific Patterns: Application to the Real-Time Domain. In Proceedings of the Advances in Databases and Information Systems, Novi Sad, Serbia, 20–24 September 2010.
17. Montero, S.; Dıaz, P.; Aedo, I. A semantic representation for domain-specific patterns. In Proceedings of the International Symposium on Metainformatics, Salzburg, Austria, 15–18 September 2004.
18. Kim, D.; France, R.B.; Ghosh, S. A UML-based Language for Specifying Domain-Specific Patterns. *J. Visual Lang. Comput.* **2004**, *15*, 265–289. [CrossRef]
19. MMLAB Educational Website. Available online: http://mmlab.ceid.upatras.gr/en/ (accessed on 15 November 2016).
20. Yan, X.; Han, J. gSpan: Graph-based substructure pattern mining. In Proceedings of the ICDM'02, Washington, DC, USA, 9–12 December 2002.
21. Philippsen, M. ParSeMiS—The Parallel and Sequential Mining Suite. Available online: https://www2.cs.fau.de/EN/research/zold/ParSeMiS/index.html (accessed on 15 November 2016).

22. Simpson, T.; Dao, T. WordNet-Based Semantic Similarity Measurement. Available online: http://www.codeproject.com/Articles/11835/WordNet-based-semantic-similarity-measurement (accessed on 15 November 2016).
23. The Graduate School of Arts and Sciences (GSAS) Website. Available online: https://gsas.harvard.edu/ (accessed on 15 November 2016).
24. Wordpress CMS Website. Available online: https://wordpress.org/ (accessed on 23 January 2017).