An Approach for Domain-Specific Design Pattern Identification Based on Domain Ontology

Vassiliki Gkantouna, Vaios Papaioannou, Giannis Tzimas, Zlatan Sabic

An approach for Domain-specific Design Pattern Identification based on Domain Ontology

Vassiliki Gkantouna[1[0000-0003-1612-2822]], Vaios Papaioannou[1[0000-0003-2654-6733]], Giannis Tzimas[2[0000-0002-4073-7256]], Zlatan Sabic[3]

[1] Department of Computer Engineering and Informatics, University of Patras, Patras, 26504, Greece
[2] Department of Computer and Informatics Engineering, Technological Educational Institute of Western Greece, Patras 26334, Greece
[3] Senior Operations Officer - Public Information Systems Specialist, World Bank Group

Abstract. In this work, we present an approach for supporting the identification of domain-specific design patterns based on domain's ontology, since the latter encapsulates the knowledge about the problem domain. More specifically, the proposed approach automatically analyzes the designs of a collection of domain-specific websites in terms of all the recurrent patterns occurring among them, both in the organization of their content and the front-end interface of their pages, resulting in a set of reusable design solutions which are commonly used in them by designers as building blocks for addressing typical domain problems. Then, evaluation is performed according to a number of inspection steps. At a first level, the recurrent patterns occurring at content organization, i.e., the common configurations of domain concepts occurring among website pages are evaluated by matching them against the domain's ontology and selecting the ones which are in alignment with the domain's context. At a second level, the recurrent patterns occurring at front-end organization (i.e., the common configurations of front-end design elements) are evaluated towards their consistent and effective use in designs of the collected websites. Finally, the approach categorizes the various reusable design solutions and recommends the ones with the best evaluation results as candidate domain-specific design patterns.

Keywords: Domain ontologies, Domain-specific design patterns, Content Management Systems.

1 Introduction

Domain-specific design patterns (Pree, 1997) are a powerful conceptual tool for designing web applications of high quality in a certain application domain. By using them, developers can gain a number of important advantages, such as increased design quality, reduced development and maintenance costs, and better communication since they provide a common vocabulary to discuss the various design alternatives. However, despite their numerous advantages, there is still an absence of techniques to systematically assist domain designers with their identification. To this end, we here propose an

approach which automatically supports the identification of domain-specific design patterns, by providing designers with recommendations about candidate patterns. More specifically, we propose an approach that automatically analyzes the designs of domain-specific websites in terms of the various reusable design solutions which are used in them by designers as building blocks for addressing common domain problems. Then, in order to verify and validate that the identified design solutions truly lie and are applicable in the target domain, we evaluate them against the domain ontology and also apply a number of evaluation metrics on them. Finally, the approach categorizes the various reusable design solutions and recommends the ones with the best evaluation results as candidate domain-specific design patterns.

In order to automate the process of capturing the designs of domain-specific websites, we have narrowed down the scope of the approach to the area of Content Management Systems (CMSs) and illustrate its potential on websites built on top of Joomla! CMS. To explain the concepts of the proposed methodology, we focus on the domain of academic websites and refer to various instances of real Greek academic websites. The remaining of this paper is organized as follows: Section 2 provides an overview of the related work and discusses the contribution of this work. Section 3 presents in detail the approach, while section 4 discusses conclusions and future work.

2 Related Work & Contribution

The main objective of this work is to assist domain designers with the identification of domain-specific web design patterns. Therefore, we have developed an approach to provide them with automatically identified recommendations about candidate patterns based on the domain's ontology. By reviewing the literature for studies concerning the identification of domain-specific design patterns, one can find out that the research in the field is still not mature enough. To begin with, despite the fact that there are many catalogues of web design patterns (Tidwell, 2010; Van Welie, 2019; Toxboe, 2019), we noticed that only a small percentage of the patterns included in them are domain-specific and the domains explored are very few. Another issue is that the identification of domain-specific design patterns relies on a completely manual process (Van Welie and Klaasse, 2004; Pontico et al., 2008; Cremonesi et al., 2017) and the whole burden of the process is carried by designers, requiring a great amount of time and effort. Here lies the main contribution of the proposed approach, since it aims to assist domain designers by attempting to support the pattern identification process in an automated way. More specifically, the proposed approach supports the automated design analysis of various websites in a particular application domain, as well as, the automated identification of the reusable design solutions which are commonly used in them for addressing typical domain problems. As a result, by simply providing a list of URLs that belong to domain-specific websites, domain designers can have access to a set of reusable micro-architectures which are used in the website designs as building blocks for addressing recurring domain problems.

3 The Approach

The approach can be divided in three main steps as depicted in **Fig. 1**. First, we identify all the recurrent patterns occurring in the organization of content and front-end interfaces of a collection of domain-specific websites, resulting in a set of reusable design solutions revealing potential design practices which are commonly used by designers in the domain for supporting the realization of domain functionalities (common domain-specific tasks). Then, in order to verify which of them are effectively used, denoting the existence of domain-specific design patterns, we validate and match them against domain's ontology, apply a number of evaluation criteria on them, and finally, recommend the ones with best evaluation results as candidate patterns.

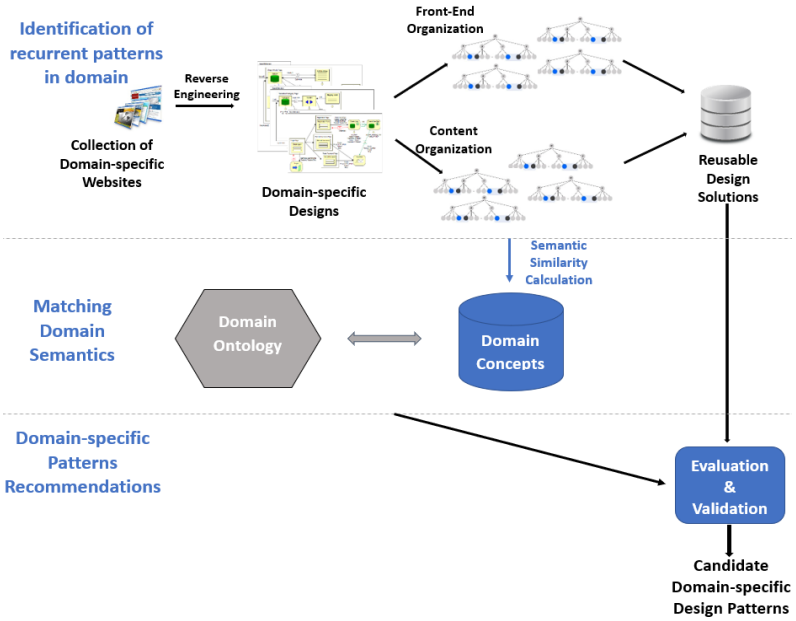

Fig. 1. The approach overview.

3.1 Identification of recurrent patterns in domain-specific designs

The identification of recurrent patterns in the designs of the collected domain-specific websites is carried out in the following four phases:

Phase 1: Domain-specific websites collection & extraction of domain concepts

In the first phase, we create the collection of websites in the target application domain by employing a web crawler which, given a list containing the URLs of domain-specific websites, traverses all of their pages and makes a local copy of them on the user's computer. Subsequently, based on the semantics that is encapsulated on the pages of the collected domain-specific websites, we automatically capture domain concepts by applying a semantic similarity measurement technique on their contents. To support this task, we have developed a tool which first collects the content of every page in the

website collection, and then applies a WordNet-based semantic similarity measurement technique (Simpson and Dao, 2005) on them in order to extract the common semantic concepts to which they refer. We consider all the computed common semantic concepts as domain concepts and store them in a database table, called domain dictionary. This way, a domain concept is assigned to every page of the collected websites.

Phase 2: Capturing the designs of domain-specific websites

In the second phase, we capture the designs of the collected domain-specific websites by automatically extracting their conceptual model at hypertext level, which specifies the organization of their front-end interfaces in terms of pages, made of a number of predefined structural and navigational design elements, called components and modules. When used in a page, all these types of components and modules, can be found in its HTML code as elements, having characteristic HTML class attribute values (i.e.,). As a result, by parsing the HTML code of the pages of the collected websites and locating the occurrences of these characteristic HTML class attribute values in them, we can retrieve the organization of the various front-end design elements that compose their hypertext. For example, **Fig. 3**(a) presents the hypertext organization identified for the "Faculty" page of an academic department website. This task is supported by the toolset depicted in **Fig. 2**, which retrieves in the reverse engineering way described above the conceptual model of the collected websites.

Next, in order to facilitate the subsequent identification of recurrent patterns in the designs of the collected websites, the conceptual model of every website is represented in the form of two directed graphs. The first graph representation captures their designs at the level of content organization in them, that is, the organization of the various domain concepts among their pages. The second graph representation captures their designs at the level of hypertext organization, that is, the organization of the various design elements in the front-end interfaces of their pages. This task is supported by the "Graph Translator" unit depicted in **Fig. 2**. An example can be found in **Fig. 3**(b) and (c) which present an instance of the two graph representations produced for the concep-

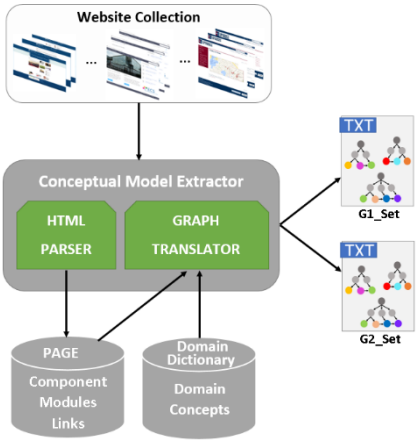

Fig. 2. The conceptual model extraction process.

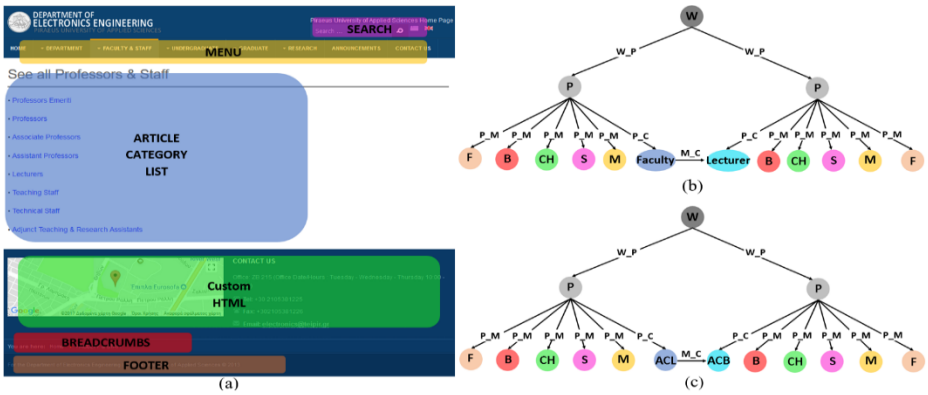

Fig. 3. (a) The organization of a page in terms of Joomla! component and modules. (b) The first graph representation of the page at the level of content organization. (c) The second graph representation of the page at the level of front-end interface organization.

tual model of an academic department website. Finally, the tool produces as output two TXT files containing the two graph representations produced for the conceptual model of every website in the collection. These two graph datasets are provided as input to the graph mining algorithm utilized in the next phase, in order to perform their pattern-based analysis and obtain the identification of the recurrent patterns in the organization of their content and hypertext interfaces.

Phase 3: identification of recurrent patterns in content and hypertext organization

In the third phase, we analyze the designs of the collected websites in terms of the recurrent patterns occurring in them, both at the level of content and hypertext organization. To achieve this, we reduce the problem of pattern identification in the two graph datasets produced in the previous phase to the subgraph isomorphism problem, synopsized in its general form into finding whether the isomorphic image of a subgraph exists in a larger graph. Intuitively, the isomorphic subgraphs occurring within a graph dataset is an alternative way to obtain the identification of the recurrent patterns among its graphs. Quite a few heuristics have been proposed to solve it, among which we have selected the most prominent one, that is, the gSpan algorithm (Yan and Han, 2002). Thus, by applying gSpan (Philippsen, 2011) on the two graph datasets, we identify all the recurrent patterns occurring in the content and hypertext organization of the collected domain-specific websites. We must note that the recurrent patterns in the content organization of the websites specify information flows which determine domain functionalities, i.e., common domain-specific tasks, whereas the patterns in hypertext organization specify design fragments which are used to support the realization of these domain functionalities. For example, **Fig. 4** presents two recurrent patterns identified in the designs of academic websites. **Fig. 4**(a) presents a recurrent pattern occurring in the content organization of various academic websites that we have used in our case study, consisting of the following configuration of domain concepts among three website pages, i.e., {Staff → Faculty → Professor}. **Fig. 4**(b) presents a recurrent pattern

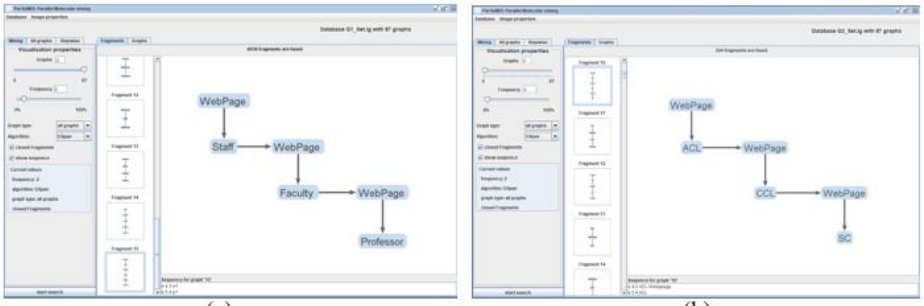

Fig. 4. Recurrent patterns in the designs of academic websites. (a) A pattern occurring in the content organization of academic websites. (b) A pattern occurring in the front-end interface organization of academic websites.

occurring in the front-end interface organization of various academic websites, consisting of the following configuration of front-end design elements among the three involved website pages, i.e., {ACL → CCL → SC}. If the pattern in Fig. 4(b) coexists in the designs of the academic websites with the pattern of Fig. 4(a), we can assume that the front-end design elements composing the design fragment specified in Fig. 4(b) are used to support the domain functionality depicted in Fig. 4(a), that is, to allow users to find the contact details of the various professors included in the faculty staff category of an academic department. As a result, in order to identify the commonly used design practices in the collection of domain-specific websites, we examine the occurrences of the identified recurrent patterns in the two graph datasets and identify those patterns that coexist, i.e., they occur simultaneously both at content and hypertext organization level.

At the at the end of this phase, for every domain functionality (recurrent configuration of domain concepts), we identify a list of all the various recurrent design fragments which are commonly used in the designs of the collected websites in order to support its realization. Every possible combination of a design fragment in this list at the hypertext organization of the websites with the particular information flow of this domain functionality at their content organization can possibly result in a reusable design solution that is commonly used in the domain for addressing a certain domain problem. Have in mind that behind such a reusable design solution, there may be hidden a good design practice used by designers in the domain under study for addressing typical domain problems, signifying the existence of a possible domain-specific design pattern. To verify this, in the next phase, we proceed with the evaluation of the identified reusable design solutions.

Phase 4: Evaluation

In this step, we apply a number of evaluation criteria on the previously identified recurrent patterns in order to determine which of them can be considered as effective design solutions for the target application domain. Evaluation is performed in two steps. First, we evaluate the various configurations of domain concepts, captured as recurrent

patterns in the content organization of the websites, by matching them against the domain's ontology. Then, we evaluate the various recurrent design fragments, captured as recurrent patterns in hypertext organization, towards (i) their consistent use in the designs of the website and (ii) the degree of their design similarity, since the underlying design reuse among them possibly reveals designers attempt to apply a certain design practice, indicating a possible domain-specific design pattern.

3.2.1 Evaluation based on domain ontology

Domain ontology encapsulates the knowledge about the problem domain. It is an intentional semantic structure that encodes the set of objects and terms that are presumed to exist in the semantic domain, the relationships that hold among them, and the implicit rules constraining the structure of this (piece of) reality (Giaretta and Guarino, 1995). Therefore, in order to validate that the identified recurrent configurations of domain concepts are actually within the context of the target domain, we cross-check the semantic concepts composing them against the concepts of domain ontology.

To define the domain ontology, we either use an already developed domain ontology as basis for domain reference or we define the domain ontology in OWL using Protégé (Noy et al., 2001). Then, based on the domain ontology definition, we either programmatically query the ontology repository for matched terms between the ontology's concepts and the identified configurations of domain concepts with SPARQL queries (**Fig. 5**), or, given the output of the previous phase in OWL format, we perform an ontology matching algorithm (**Fig. 6**) to check the validity of the terms composing the identified configurations of domain concepts. **Fig. 5** and **Fig. 6** provide some relevant examples. In **Fig. 5**, the user validates an identified configuration of domain concepts by programmatically querying this configuration against the domain ontology of the academic domain. We can validate all terms included in an identified configuration of domain concepts one by one, such as in **Fig. 5**. Assuming that the concept "Professor" is one of these concepts, the toolkit validates this concept via a SPARQL query. It also suggests the concept "Lecturer", since this is also a subclass of the class Academic staff.

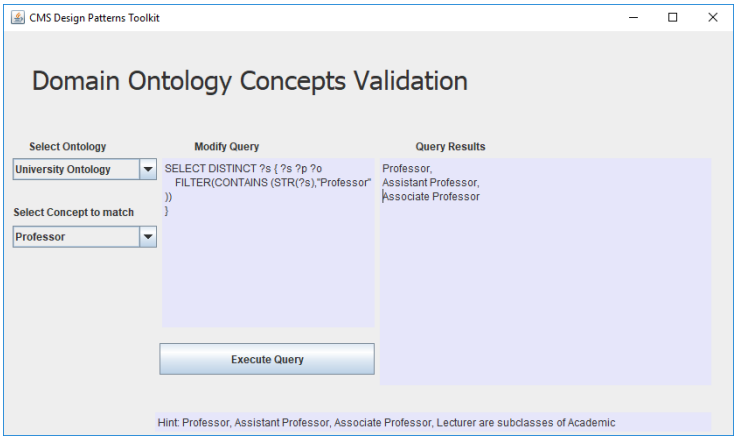

Fig. 5: Querying the domain ontology.

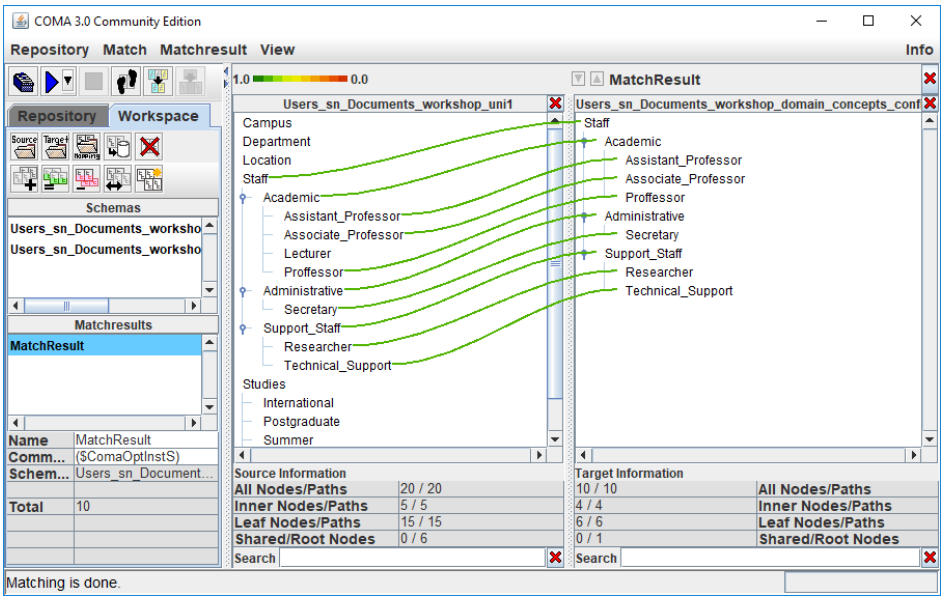

Fig. 6: An example of ontology mapping.

Fig. 6 presents an example of ontology matching approach based on the open-source API implementation of COMA (Massmann et al., 2011). In **Fig. 6**, the entire configuration of domain concepts belongs to the University ontology with estimated matching value of 1.0 (ranging from 0.0 to 1.0, green color). Thus, the configuration of domain concepts is totally matched against the domain ontology. The matching results shows that this configuration lies in the semantic context of the domain and it can also be optionally enhanced. For example, the concept of "Lecturer" is missing from the original discovered domain concept configuration (no connection line) and now can be utilized.

In both cases (either in programmatically querying ontology or in ontology mapping), the matching results can fall into three cases:

1. The identified domain concepts composing a recurrent pattern are a subset of the domain ontology's concepts, verifying that the websites composing the collection truly fall under the umbrella of the target application domain under consideration. Web designers can either use this configuration as it has been identified or even extend it with additional concepts specified in the domain ontology. For example, if the following configurations of domain concepts have been identified in a collection of academic websites, {Professor → Associate} and {Professor → Assistant}, then by matching them against the University Ontology, a further examination of the Academic class hierarchy can reveal the concept of "Lecturer" which can be used to extend the identified configuration of domain concepts.
2. The identified domain concepts in a configuration do not belong in the set of domain ontology's concepts, denoting either that the website collection is invalid for this

domain or that the semantic analysis of the contents of website pages has produced invalid results.

3. The identified domain concepts composing the recovered patterns partially belong to the set of domain ontology's concepts. In this case, web designers can decide on how to make use of the identified domain concepts and even enhance them with concepts of the domain ontology. For example, the semantic analysis of domain concepts concluded to the term Professor among other academic terms, but no other types of Academic staff were discovered. In this case, web designers might examine the academic staff class hierarchy, and finally include Associate Professor and Assistant professor in their domain concepts configuration.

This way, we validate and verify which of the identified configuration of domain concepts truly lie in the specific context of the target application domain, and select those which are in alignment with its semantics.

3.2.2 Evaluation based on consistent use and effectiveness

After selecting the design fragments which are used to deliver to end-users the configurations of domain concepts which are valid according to the domain ontology evaluation of the previous step, the next step is to evaluate them towards their consistent use in the designs of the collected website, as well as, towards the degree of their design similarity.

3.2.2.1 Consistent use of recurrent design fragments

To determine which of the design fragments are effectively used for the realization of the identified domain functionalities (i.e., configurations of domain concepts), we evaluate them towards their consistent use in the website designs with respect to their starting and ending variants. More specifically, in phase 3 of Section 3.1, when obtaining the identification of the recurrent patterns in the graph datasets, except from the pattern's core specification (i.e., invariant composition of design elements), we also identify its starting and ending variants, extending the core specification of the pattern with all the valid modalities in which it can start or terminate. Intuitively, these variants correspond to the various design structures with which it is connected (on its left or right side) in the collected websites in order to execute a domain functionality. Thus, to verify the consistent use of a recurrent design fragment in the various website designs, we need to consider it with respect to its starting and ending variants.

To this end, we have defined two metrics called Start-Point Metric (SPM) and End-Point Metric (EPM), respectively. Assuming that a recurrent design fragment can have N starting and M ending variants, these metrics compute the statistical variance of the occurrences of these variants in the website designs, normalized with respect to the best-case variance. They are calculated according to the following formulas:

$$SPM = \sigma_S^2 / \sigma_{S,BC}^2 \qquad EPM = \sigma_E^2 / \sigma_{E,BC}^2 \quad (1)$$

σ_S^2 and σ_E^2 is the statistical variance of the N starting variants occurrences and the M ending variants occurrences respectively, calculated according to the formula (2):

$$\sigma_S^2 = \frac{1}{N}\sum_{i=1}^{N}\left(p_i - \frac{1}{N}\right)^2 \qquad \sigma_E^2 = \frac{1}{M}\sum_{i=1}^{M}\left(p_i - \frac{1}{M}\right)^2 \quad (2)$$

where p_i is the percentage of occurrences for the i-th pattern variant. $\sigma_{S,BC}^2$ and $\sigma_{E,BC}^2$ are instead the best-case variances for the starting and ending variants, calculated by the formula (2) assuming that only one variant has been coherently used throughout the

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

Table 1. The measurement scale for the SPM and EPM metrics.

website. We have also defined a measurement scale specifying a mapping between the numerical results obtained through the calculus method and a set of (predefined) meaningful and discrete values, as defined in **Table 1**.

We compute the SPM and EPM metrics for all the starting and ending variants of the recurrent design fragments supporting the various domain functionalities and select only the ones having SPM and EPM values less than 0.6, and store the results in the "Results Repository". This way, we obtain a first level categorization of them based on the consistent use of their variants.

3.2.2.2 Design similarity of recurrent design fragments

Another important factor that must be taken into account in order to determine whether the identified recurrent patterns can be considered as candidate domain-specific patterns is their design similarity. This is due to the fact that high design similarity among them implies underlying design reuse which possibly occurs due to developers attempts to maintain a common design practice in the domain for the realization of a domain functionality, thus, signifying the existence of domain-specific design patterns. To this end, we perform a second level categorization of the previously selected recurrent design fragments based on the degree of their design similarity. To compute the degree of design similarity among a list of recurrent design fragments (i.e., configurations of design elements) that support a domain functionality, we adopt the vector space model. More specifically, every design fragment is represented as a vector $d_i = (x_{1,i}, x_{2,i}, \ldots, x_{n,i})$, where the compounds comprise all the distinct Joomla! design elements occurring among the various design fragments included in the list. These compounds are considered as unigrams and are weighted by the frequency of each respective unigram, that is, the number of times in which the design element (to which they correspond) occurs in the recurrent design fragment under consideration. Then, we compute the degree of design similarity between every possible pair of recurrent design fragments in the list, by calculating the cosine of the angle between their corresponding vector representations d_i and d_j. The latter is calculated according to the following formula:

$$Similarity(d_i, d_j) = \cos(d_i, d_j) = \frac{d_i \cdot d_j}{|d_i| \cdot |d_j|} = \frac{\sum_{t=1}^{n} w_{t,i} \cdot w_{t,j}}{\sqrt{\sum_{t=1}^{n} w_{t,i}^2} \cdot \sqrt{\sum_{t=1}^{n} w_{t,j}^2}}, \in [0,1]$$

where $|d_i|$ and $|d_j|$ are the norms of the two vectors. To quantify the degree of this similarity, we have defined five similarity levels, presented in **Table 2**. The higher the level of design similarity among the design fragments, the more possible the design reuse occurring in them to capture a common design practice in the domain for the realization of the domain functionality.

Design Similarity Level	Design fragments composed by:
1	▪ totally different configurations of design elements
2	▪ configurations of design elements identical up to 25%
3	▪ configurations of design elements identical up to 50%
4	▪ configurations of design elements identical up to 75%
5	▪ identical configurations of design elements

Table 2. Design fragments categorization based on design similarity.

At the end of this step, for every identified list of recurrent design fragments that supports a domain functionality, we locate all the sets of similar design fragments in the list, and identify the various design reuse schemes occurring in them. Then, we categorize them according to their design similarity level, and present the results in descending order. This way, we manage to capture the various common design practices which are used in the domain for addressing the domain functionalities. In order to reduce the large number of the identified design fragments, we select only the ones that occur in design fragments having a design similarity above the third level. By inspecting the identified design reuse schemes, domain designers can have a global picture of the various design practices that are commonly used in the domain for realizing the various domain functionalities. By combining these results with the results concerning their consistent use in the designs of the collected websites, it becomes easier for designers to determine which of the identified reusable design solutions can be considered as good design practices for the target domain, signifying the existence of candidate domain-specific design patterns.

4 Conclusions & Future Work

In this work, we have presented an approach that recommends automatically identified candidate reusable design solutions to domain designers in order to assist them with the identification process of domain-specific design patterns. To obtain these recommendations, it automatically analyzes the designs of a collection of websites in a target application domain, identifies and evaluates all the reusable design solutions occurring among them for addressing typical domain problems. For the evaluation of the recovered design solutions, we have relied on the domain ontology in order to capture

those which are in alignment with the domain's semantics, on their consistent use and on their design similarity. Finally, the approach categorizes the various reusable design solutions and recommends the ones with the best evaluation results as candidate domain-specific design patterns. By having access to the results, domain designers can have an overview of the common design practices used in the domain for addressing typical domain problems, among which they can possibly recognize best practices and capture domain-specific design patterns. In future, we plan to apply the methodology in various application and extend it, so that it can be applied in websites built on top of other popular CMS platforms, such as Drupal and WordPress.

References

1. Cremonesi, P., Elahi, M., Garzotto F., 2017. User interface patterns in recommendation-empowered content intensive multimedia applications. Springer Multimedia Tools and Applications, 76(4):5275–5309.
2. Giaretta, P., & Guarino, N. (1995). Ontologies and knowledge bases towards a terminological clarification. Towards very large knowledge bases: knowledge building & knowledge sharing, 25(32), 307-317.
3. Massmann, S., Raunich, S., Aumüller, D., Arnold, P., & Rahm, E. (2011, October). Evolution of the COMA match system. In Proceedings of the 6th International Conference on Ontology Matching-Volume 814 (pp. 49-60). CEUR-WS. org.
4. Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R. W., & Musen, M. A. (2001). Creating semantic web contents with protege-2000. IEEE intelligent systems, 16(2), 60-71.
5. Philippsen, M., 2011. ParSeMiS - the Parallel and Sequen-tial Mining Suite. Available at: https://www2.cs.fau.de/EN/research/zold/ParSeMiS/index.html (Accessed: 16th March 2019).
6. Pontico, F., Winckler, M., Limbourg, Q., 2007. Organizing User Interface Patterns for e-Government Applications. In: Gulliksen, J., Harning, M.B., van der Veer, G.C., Wesson, J. (eds.) EIS 2007. LNCS, vol. 4940, pp. 601–619. Springer, Heidelberg (2008)
7. Pree, W., 1997. Essential Framework Design Patterns. Object Magazine, vol. 7, pp. 34-37.
8. Simpson, T., Dao, T., 2005. WordNet-based semantic similarity measurement. Available at: http://www.codeproject.com/Articles/11835/WordNet-based-semantic-similarity-measurement (Accessed: 16th March 2019).
9. Tidwell, J., 2010. Designing interfaces. O'Reilly Media, Inc
10. Toxboe, A. User interface design pattern library. Available at: http://ui-patterns.com/patterns (Accessed: 16th March 2019)
11. Van Welie, M. Web Design Patterns. Available at: http://www.welie.com/patterns/ (Accessed: 16th March 2019)
12. Van Welie, M., Klaasse, B., 2004. Evaluating Museum Websites using Design Patterns. Technical report, Internal Report IR-IMSE-001.
13. Yan, X., Han, J., 2002. gSpan: Graph-based substructure pattern mining. In: Proceedings of ICDM'02, pp 721–724