Mining effective design solutions based on a model-driven approach

T. Katsimpa[2], S. Sirmakessis[1], A. Tsakalidis[1,2] & G. Tzimas[1,2]
[1]*Research Academic Computer Technology Institute, Hellas*
[2]*Computer Engineering & Informatics Department, University of Patras, Hellas*

Abstract

The World Wide Web evolution has posed new challenges for Web application developers. Today's Web applications present their content in a dynamic manner, allow users to interact with them at a high level of sophistication, and manage huge amounts of data, leading to a high degree of complexity. To cope with this emerging problem, a variety of modeling methodologies, web design patterns and techniques have been introduced.

In this work we propose a methodology for mining and evaluating recurrent design solutions in the conceptual schema of Web applications modeled using WebML, a modeling language for designing data-intensive applications. Identifying effective design solutions may lead to an efficient deployment of reuse. By means of our approach, an extension of the predefined set of patterns supported by WebML can be realized. What is more, when applying the methodology in a large number of applications of the same domain, templates for Web application frameworks can be identified.
Keywords: reuse, web modelling, web design patterns, model mining.

1 Introduction

Nowadays, the design, development and maintenance of Web applications have become some of the most challenging problems that web architects have to face. Modern Web applications are completely different in comparison to the traditional Web sites which provide only static content. They are required to manage huge amounts of data, execute various complex functions and deliver their content in a variety of forms, according to the user profile or role and according to the media with which the user accesses the application.



In order to provide means for the effective design and deployment of Web applications, several methodologies and techniques have been proposed by the research community. These methodologies, Retschitzegger and Schwinger [1], elevate the level of abstraction in the design process, provide technology independent design solutions and are based on the idea of separation of concerns. They incorporate different models for data management, navigation structure and presentation of pages to the end-users. HDM was the first methodology to introduce the model-driven approach in the area of hypermedia design and was followed by RMM. Later, based on the HDM a number of modelling methodologies were introduced mainly focusing on Web Application design, such as HDM-Lite, Strudel and OOHDM. Araneus is a proposal for web design and reverse engineering. Extensions of the UML language so as to make it suitable to model Web applications have been proposed by Conallen. Finally, a new modelling language was introduced, the Web modelling language – WebML, which is destined to specify complex Web applications at the conceptual level and along several dimensions. The methodologies noted above provide a concrete framework for Web application design, elevating the efficiency in the design process and providing a firm basis for re-engineering and maintenance of web applications.

Utilizing previous experience can also boost the final application's quality. To this end, the notion of design patterns was first introduced in the field of civil engineering, Alexander *et al.* [2] and now has expanded to a diversity of domains, including software engineering and in particular Web engineering.

Design patterns represent design knowledge culled from the solutions given by experienced designers that have encountered these same problems before, possibly with slight variations Schwabe *et al.* [3]. By using patterns, the web architect constructs applications starting from customizable components, rather than from scratch, and enforces a coherent design style over large and complicated applications, augmenting hypertext regularity and usability.

Taking the use of design patterns one step further, the notion of Web application frameworks for specific domains was introduced [3,4]. Although the use of web design patterns is valuable when building a web application, complex Web applications need to maximize reusing larger design structures (the whole navigation topology or structure plus associated functionality). These design structures may arise at the application (conceptual level) or during navigational design. Designers should find ways to express these commonalities by using generic designs such that only aspects unique to a particular site should be designed or programmed. Web application frameworks capture the design decisions that are common to a category of Web sites and provide reusable design for that category Ceri *et al.* [4].

From the above, the need and importance to identify reusable design solutions is rather obvious. The contribution of this paper is the provision of a mechanism, which mines and evaluates reusable design solutions at the conceptual schema of a Web application built using WebML. Moreover, applying the proposed mechanism in a number of Web applications of the same domain, Web application frameworks can be identified. In this work, WebML has been utilized



as design platform for the methods proposed, mainly due to its robustness and the fact that it is supported by an efficient case tool called WebRatio [5]. Most of the work presented here can be generalized and applied to applications utilizing other modeling languages with slight straightforward modifications.

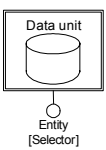
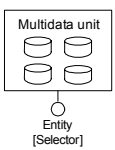
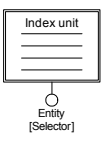
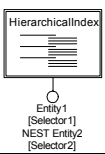
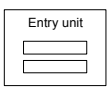
The remainder of this paper is organized as follows: Section 2 presents some background concepts referring to the WebML method. Section 3 provides a motivating example, while section 4 describes our methodology to mine reusable design solutions at the conceptual schema of Web applications. Section 5 illustrates a quality evaluation mechanism for the solutions retrieved by means of consistency. Finally, section 6 draws our conclusions.

2 WebML at a glance

Before proceeding to the rest of the paper, it is important to provide a quick overview of the WebML core concepts. WebML is a notation for specifying complex Web sites and applications at the conceptual level Ceri *et al.* [6]. WebML consists of the *Data Model* and the *Hypertext Model*. The data model utilizes the well known E/R model, in order to organize the content of the application. The hypertext model describes how data objects are assembled into information units and elements and the way these elements are connected by links to form hypertexts. Different hypertexts, called *site views*, that support different user profiles, can be created among the same data model. Site views are graphs of *pages*, which are the actual containers of information and are built of *content units*. Units represent atomic pieces of information to be published; a *unit selector* specifies a predicate identifying the data to be extracted from the underlying database and displayed by the unit. Pages and units can be linked in a variety of configurations by means of *links* to form a hypertext. Besides representing user navigation, links between units also specify the transfer of certain information (called *context*) that the destination unit uses for selecting the data instances to be displayed.

Finally, the execution of arbitrary business actions is modeled by means of *operation units*. An operation unit can be linked to other operations or content units. WebML incorporates some predefined operations for creating, modifying and deleting the instances of entities and relationships, and allows developers to extend this set with their own operations.

Table 1: Some basic WebML units.

				
It displays a set of attributes for a single entity instance.	It displays a set of instances for a given entity.	It displays a list of properties, of a given set of entity instances.	It displays index entries organized in a multi-level tree.	It represents forms for collecting input data into fields.

WebML acknowledging the importance of design patterns has defined a primitive set of design patterns, offering proved solutions to problems occurring in real-life applications. According to the role information objects play in the application, four classes of objects have been identified (core and interconnection concepts, access facilitators, personalization objects), that can be organized in a variety of ways to construct sub-schemas Ceri *et al.* [7]. The organization of content in web pages can also be simplified by making use of design patterns. Recurrent navigation chains of unit compositions for content publishing (cascaded index, indexed guided tour, object viewpoint etc.), as well as patterns for content management operations (object creation-deletion-modification, cascaded delete etc.) have been identified Ceri *et al.*[6].

A pattern in WebML, typically consists of a *core specification*, representing the invariant WebML unit composition that characterizes the pattern, and a number of *pattern variants*, which extend the core specification with all the valid modalities in which the pattern can start (*starting variants*) or terminate (*termination variants*). Starting and termination variants respectively describe which units can be used for passing the context to the core pattern and how the context generated by the core pattern is passed to successive compositions in the hypertext Fraternali *et al.* [8]. WebML, furthermore, separating the various applications according to their "information-delivery mission", has identified skeletons and WebML application frameworks Ceri *et al.* [4] that can be applied and used to Web applications, depending on their functionalities.

3 Motivating example

In this section we provide an instance of a WebML scheme modelling an e-learning scenario that will serve as a running example throughout this paper. We assume that every lesson comprises of a number of lectures, each of which contains one more exercises. Each lesson is supported by a forum, which comprises of topics containing messages. The teachers and students are the basic groups of users that can access this application, each of which has different permissions.

The first part of figure 1 depicts part of a teacher's site view. Suppose that the teacher wants to alter the contents of an exercise. A global parameter is used to hold the object identifier, OID, of the teacher currently logged into the Web application. Initially, the teacher views a hierarchical index of the lessons he teaches per category. The 'Lesson' page contains the 'Lesson' data unit providing information about the lesson selected and the 'LecturesIndex' shows a list of the lesson's lectures. Finally, the teacher, by selecting one of the lectures, can access the 'Exercises' multidata unit, that displays all exercises related to this specific lecture. Choosing the desired exercise, he may modify its contents.

The second part of the same figure depicts part of a student's site view. Suppose that the student wants to view the messages of a lesson's forum. A global parameter is used to hold the OID of the student currently logged into the application. A hierarchical index contains a two-level hierarchy of lessons per category that the student attends. Selecting a lesson, he can view information related to it ('Lesson' data unit). In the same page, the 'ForumsIndex' displays

all the topics concerning the chosen lesson, whereas the 'Topic' data unit displays information about a specific topic. Finally, the 'Messages' multidata unit shows the messages that belong to the selected topic. The student can select a message in order to view it in detail.

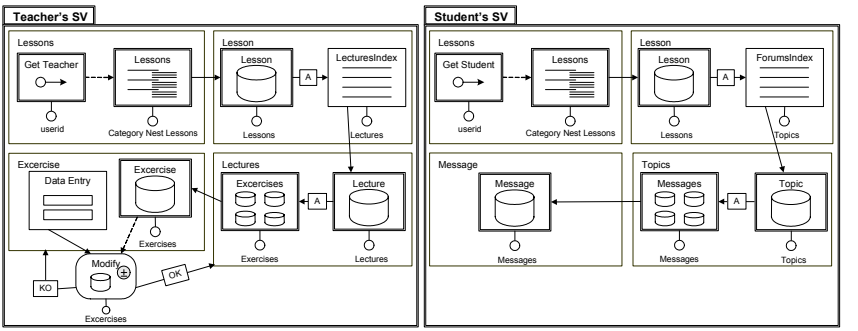


Figure 1: Teacher's and student's site views.

4 Methodological approach

In the sequel, we present in detail a methodology for mining recurrent design solutions in the WebML conceptual schemas of applications. Our objective is to capture compositions of hypertext elements (pages, units, operations, links) serving several application purposes. These configurations are captured in the process of (or after) modelling a Web application, and can serve as effective design solutions enabling reuse, consistency and quality improvement in the development and maintenance process.

4.1 Initialisation

In this phase we preprocess the application's hypertext model, in order to form the basis for the model mining mechanism needed in the phases to follow. More precisely, considering an application having a number of site views we convert every site view into a directed graph based on the notation described below, thus constructing a set of graphs representing the navigation, content presentation and manipulation mechanisms of the application.

We define a site view as a directed graph, $G(V, E, f_V, f_E)$, comprising of a set of nodes V, a set of edges E, a node-labeling function f_V: $V \rightarrow \Sigma_V$, and an edge-labeling function f_E: $E \rightarrow \Sigma_E$. f_V assigns letters drawn from an alphabet Σ_V to the site view nodes, whereas f_E has the same role for links and the edge alphabet Σ_E. Σ_V has a different letter for each different WebML element (content units, operations, pages etc). Correspondingly, Σ_E comprises of all the different kinds of links (contextual, non contextual etc.). Besides the predefined WebML links, we introduce a special kind of edge (labeled *'c'* in the graph) in order to represent the containment of units or sub-pages in pages, as well as pages, sub-

areas and units in areas (note that arbitrary containment sequences can exist). A transformation example of the two site views described in section 3 is depicted in fig. 2.

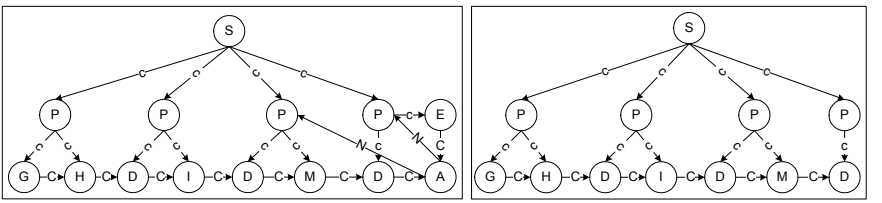


Figure 2: Graph transformation of the two site views.

4.2 Extraction of predefined patterns and uniformity

We traverse iteratively each graph constructed in the previous phase in order to locate predefined content publishing and management WebML patterns taking into account their variants. Every pattern found is stored in a *pattern occurrences repository*, along with its starting and termination variants. The occurrence frequency of each pattern is also stored. The retrieval of a predefined pattern can be achieved using graph matching tools such as GraphGrep [9], which given a collection of graphs and a sub-graph (in our case, graphs representing the predefined patterns), returns all the occurrences of the pattern in each graph.

Next, taking into account the various WebML predefined variants we have located, we substitute, where possible (this step requires the designer's intervention), the variants found within each graph with the default pattern variant. The default pattern variant is the one having the maximum occurrence frequency. Upon completion of this step, a more uniform definition of each site view is accomplished, and new graphs representing the hypertext schema are generated.

4.3 Design solutions extraction

We traverse each newly generated graph so as to identify identical subgraphs - *candidate design solutions*. Although the problem is NP-complete Garey and Johnson [10], this can be achieved by utilizing several heuristic algorithms (e.g. CloseGraph Yan and Han [11]). These algorithms demand as input the dataset of graphs to be searched and the desired support (the minimum percentage of occurrences in the entire database) of the subgraphs to be retrieved. The extracted subgraphs should not represent predefined WebML patterns, but may contain one or more of them. We store the mined subgraphs in the pattern occurrences repository, along with their variants.

For instance, comparing the graphs that correspond to our example, an identical subgraph is easily identified. The entire student's graph constitutes a subgraph of the teacher's graph and should be stored. The teacher's graph extends the identified candidate design solution with a subgraph that corresponds

to an object modify pattern. Thus, a variant of the candidate design solution can be extracted from the teacher's graph, and should also be stored.

Afterwards, we examine the retrieved design solutions and substitute, where possible, the variants with the default construct variant, generating new graphs representing the hypertext schema of the application. The newly generated graphs are once again searched for hypertext configurations that were not retrieved in the previous step.

4.4 Extending the sets of design solutions

This phase aims at acquiring a larger number of candidate design solutions. First, we examine every solution stored in the repository, and in case it contains predefined WebML content management patterns, but one or more of the remaining patterns is/are missing, we add the respective missing ones. For instance, if we locate a design construct, whose starting or termination variant is a create pattern, we complement it by adding the modify and/or delete pattern(s). In our example we retrieved a variant that contains a modify object pattern. Thus, more variants that contain the object deletion or creation pattern in place of the object modify pattern should be stored in the repository.

The repository can be further extended with broader design solutions, if we take into account the intermediate constructs of units that may connect two or more already mined design constructs. Traversing the graphs, we search for couples of subgraphs (already mined design configurations) that are interconnected through different nodes or subgraphs. We query for *interconnection variants* that are smaller than any other configuration involved in the larger design solution (fig. 3). Therefore, we utilize these constructs that include, at most, the minimum number of units among the configurations involved. A whole site view, in the worst case, could be thought of as a candidate design solution, but it will be discarded due to the evaluation metrics of the following section.

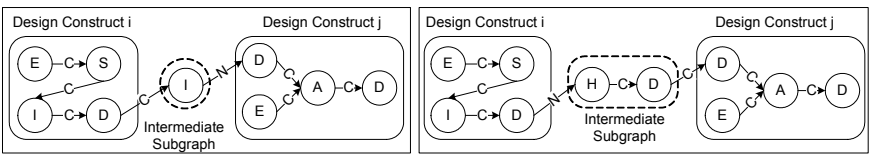


Figure 3: Extending the set of design solutions.

4.5 Evaluation metrics

Up to this point, the repository may contain a large number of candidate design solutions. This number may increase even more, if we apply the methodology to complex Web applications or to a number of applications. In order to decide whether the retrieved design constructs can be considered as effective design solutions, we have to evaluate them in terms of importance and propose a first level ranking. To this end several factors should be taken into account.

A first crucial factor is the frequency of appearance of a specific design construct in the conceptual schema. High frequency implies high probability of effective use of a specific navigational chain. We define as *appearance f* the frequency of a design solution, which comprises of: a) the overall number of appearing instances of a design solution, f_0, b) the appearance in f_a areas, c) the appearance in f_σ site views, and d) the appearance in f_π pages.

The size of an extracted configuration can be also considered as a very important metric. It is obvious that repetition of a large design configuration in the hypertext model of an application implies a high probability for having identified the use of an important solution. Thus, we introduce *population p*, as the number of WebML elements involved in the configuration. Population depends on: a) p_δ, the number of content units, b) p_l, the number of links and c) p_{op}, the number of operation units.

Combining the frequency and the population of a design solution, we compute the *importance* V_j of a design solution j utilizing the following formula:

$$V_j = f_j \, x \, p_j \tag{1}$$

$$\text{where } f_j = \frac{f_{o,j}}{f_{o,\max}} + \frac{f_{a,j}}{N_\alpha} + \frac{f_{\sigma,j}}{N_\sigma} + \frac{f_{\pi,j}}{N_\pi}, f_j \in [0,4] \tag{2}$$

f_{max} represents the maximum value of the corresponding metric, while N is the corresponding sum of areas, site views and pages overall in a specific WebML conceptual schema, and

$$p_j = \frac{p_{\delta,j}}{p_{\delta,\max}} + \frac{p_{l,j}}{p_{l,\max}} + \frac{p_{op,j}}{p_{op,\max}}, p_j \in [0,3] \tag{3}$$

Another factor that should be taken into account is the complexity d_j of a design construct, depending on the number of pages that host the construct (*fragmentation* φ). That is:

$$d_j = 1/\varphi_j \tag{4}$$

Finally, we introduce the *semantic value* e_j of a design solution j that represents the number of data model entities participating in the design solution considered per conceptual factor as: a) e_c, entities belonging to the core sub-schema, b) e_{per}, entities belonging to the personalization sub-schema and c) e_{acc}, entities belonging to the access sub-schema. The value of e_j is calculated by:

$$e_j = \frac{e_{c,j}}{e_{c,\max}} + \frac{e_{per,j}}{e_{per,\max}} + \frac{e_{acc,j}}{e_{acc,\max}}, e_j \in [0,3] \tag{5}$$

Combining the above metrics, the total impact I_j of a design solution j in a WebML conceptual schema, is given by:

$$I_j = \frac{e_j}{\phi_j} V_j \tag{6}$$

After computing the factor I_j for every design solution j in a specific conceptual schema, results are presented in a descending order to depict the most important and effective design solutions on the top. The same procedure can be applied repeatedly to site views from different applications in order to detect similar "pattern" implementation behaviours and group the results.



Having evaluated and ranked the various design constructs; the hypertext architect has access to a library of design solutions along with their variants. Applying the proposed mechanism in a number of applications of the same domain, patterns and templates for Web application frameworks, supporting the applications' commonalities and accommodating validations, can be identified. What is more, the designer's task can be further facilitated; utilizing languages such as *transformers-by example* Lechner and Schrefl [12] (a language which allows defining transformers for WebML schemes by example). The identified frameworks, in combination with transformers by example, assist the designer in performing effective and consistent modelling activities in an automatic manner and constructing wizards for future web application development.

5 Quality evaluation

The quality of the final application can benefit from the efficient adoption of successful design solutions. Even when the web architect has effective design solutions at his disposal, the problem that arises is that of verifying if they are applied consistently throughout the application. If the design solutions are consistently used, end-users find it easier to understand how to navigate through an application and how to activate operations. Moreover, web architects can predict in advance how to organize an unfamiliar section of the application.

In Fraternali *et al.* [8] a methodology is introduced for the evaluation of the consistent application of predefined WebML design patterns within the conceptual schema of an application. We utilize that methodology and extend it, in order to introduce metrics depicting the consistent application of the design solutions retrieved. In order to achieve that, we introduce two metrics that compute the statistical variance of the occurrence of the *N* termination or starting variants of the configurations retrieved, normalized with respect to the best case variance. The *Start Point Metric (SPM*) and the *End Point Metric (EPM)*. The definition of SPM follows (EPM is defined analogously):

$$SPM = \frac{\sigma^2}{\sigma_{BC}^2} \tag{7}$$

where σ^2 is the statistical variance of the starting variants occurrences, which is calculated through the following formula:

$$\sigma^2 = (\frac{1}{N})\sum_{i=0}^{N}(p_i - \frac{1}{N})^2 \tag{8}$$

N is the number of variants addressed by the metric, while p_i is the percentage of occurrences for the *i-th* configuration variant. σ_{BC}^2 is instead the best case variance, and is therefore calculated through eqn. (8), assuming that only one variant has been coherently used throughout the application.

Table 2: SPM metric measurement scale.

Metric's Range	*Measurement Scale Value*
0 <= SPM<0.2	Insufficient
0.2<= SPM<0.4	Weak
0.4<= SPM<0.6	Discrete
0.6<= SPM<0.8	Good
0.8<= SPM<=1	Optimum

The last step in the metrics definition is the creation of a measurement scale, which defines a mapping between the numerical results obtained through the calculus method with meaningful discrete values. The SPM metric adopts an ordinal nominal scale; each nominal value in the scale expresses a consistency level, corresponding to a range of numerical values of the metrics, as defined in table 2. The same scale covers EPM metric too.

6 Conclusions

In this paper, we proposed a methodology and provided metrics for mining effective design solutions at the WebML conceptual schema of an application. Moreover, we illustrated a quality evaluation technique for the solutions retrieved by means of consistency. This methodology when applied to a large number of WebML application schemas provides a mechanism for the identification of templates in specific domain Web application frameworks and can form a valuable tool for hypertext architects for the identification of Web design patterns. We are currently working on locating more precise metrics for the secure extraction of design solutions. Furthermore, applying the methodology to a large number of applications, in order to fine-tune it, is one of our goals.

References

[1] Retschitzegger, W. & Schwinger, W., Towards modeling of dataweb applications - a requirements' perspective. *In Proc. of AMCIS*, vol. 1, 2000.

[2] Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I. & Angel, S., *A Pattern Language*, Oxford University Press, 1997.

[3] Schwabe, D, Esmeraldo, L., Rossi, G. & Lyardet, F., Engineering Web Applications for Reuse. *IEEE Multimedia*, **8(1)**, pp.20-31, 2001.

[4] Ceri S., Fraternali, P. & Matera, M., WebML Application Frameworks: a conceptual Tool for Enhancing Design Reuse. *In Proc. of WWW10*, 2001.

[5] WebRatio, http://www.webratio.com.

[6] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S. & Matera, M., *Designing Data-Intensive Web Applications*, Morgan Kauffmann, 2002.



[7] Ceri, S., Fraternali, P. & Matera, M., Conceptual Modelling of Data-Intensive Web Applications. *IEEE Internet Computing*, **6(4)**, pp.20-30, 2004.
[8] Fraternali, P., Matera, M. & Maurino, A., WQA: an XSL Framework for Analyzing the Quality of Web Applications. *Proc. of IWWOST'02*, 2002.
[9] GraphGrep, http://alpha.dmi.unict.it/~graphgrep/.
[10] Garey, M.R. & Johnson, D. S., *Computers and intractability: A guide to NP-completeness*, Freeman, 1979.
[11] Yan, X. & Han, J., CloseGraph: Mining Closed Frequent Graph Patterns. *Proceedings of the 9th ACM SIGKDD03*, pp. 286–295, August 2003.
[12] Lechner, S. & Schrefl, M., Defining web schema transformers by example. *In Proceedings of DEXA'03,* Springer, 2003.

