A Methodology for Evaluating the Personalization Conceptual Schema of a Web Application

Evangelos Sakkopoulos[1,2], Spiros Sirmakessis[1], Athanasios Tsakalidis[1,2], Giannis Tzimas[1,2]

Research Academic Computer Technology Institute
61 Riga Feraiou str., GR-26221 Patras, Hellas

University of Patras, Computer Engineering & Informatics Dept.
GR-26504, Rio Patras, Hellas

{sakkopul, syrma, tsak, tzimas}@cti.gr

Abstract

While the market needs evolve rapidly, personalization has assumed an enormous industrial impact, which has caused a "Cambrian explosion" of technologies, claiming support to the personalization process. Deploying a methodology for the design and development of a Web application enhances effectiveness, but does not guarantee optimization in the design process, mainly due to the fact that a small number of extreme designers/programmers exist. The goal of this paper is to argue the need to approach personalization aspects from the very beginning in the Web application's development cycle. Since personalization is a key issue in the application's success, it is important that it should be dealt through a design view, rather than only an implementation view. In this paper, we will provide a methodology in order to evaluate the conceptual schema of an application, by means of the personalization features incorporated in the application model. The aim is to capture cases in which different ways are used to obtain the same personalization effect. We introduce the notion of model clones as partial conceptual schemas that are repeated within a broader application model. We define metrics to automate the detection and categorization of candidate model clones in order to facilitate potential model refactoring. Finally, we discuss possible model refactoring procedures with respect to personalization enhancements and detection of personalization defects.

1 Introduction

With the advent of the Internet and the World Wide Web (WWW), a renewed scenario has emerged in software development practice. At the beginning Web applications were used as plain information sharing platforms, but now they have expanded to a variety of domains. Modern Web applications are utilized in a diversity of productive contexts and range from simple information portals to complex applications providing a variety of e-services, including e-commerce, e-government, CSCW and e-learning. Most of them comprise a high degree of sophistication and complexity, supporting a variety of functionalities, incorporating advanced business logic and supporting one-to-one personalization features and content delivery using a diversity of devices.

Moreover, while the market needs evolve rapidly, personalization has assumed an enormous industrial impact, which has caused a proliferation of technologies, claiming support to the personalization process. Personalization has become hype in areas such as electronic commerce, and there exist hundreds of applications that claim to be fully customizable to different user profiles or individuals. The number of possible personalization variants seems countless. The involvement of a "myriad" of different technologies, trying to provide solutions for sophisticated web application development, in conjunction with the high market pressure for efficient and qualitative applications providing personalization mechanisms with zero number of defects has emerged the need for new software practices and methodologies. The solution must involve changing methods and practices in a way that allows for making the development of large-scale web applications much more productive.

1.1 Background & Related Work

In order to provide the basis for the application of improved technology independent software practices, the software community has proposed a number of modeling methods and techniques providing a higher level of abstraction in the design and development process of Web applications. Some proposals derive from the area of hypermedia applications like the Relationship Management Methodology (RMM) (Isakowitz et al., 1995) and the HDM

(Garzotto et al., 1993) which pioneered the model-driven design of hypermedia applications and influenced several subsequent proposals for Web modeling like HDM-lite (Fraternali & Paolini, 1998), a Web-specific version of HDM, Strudel (Fernandez et al., 1998) and OOHDM (Schwabe & Rossi, 1998). Araneus (Atzeni et al., 1998) is a proposal for Web design and reverse-engineering. The WSDM method (De Troyer & Leune, 1997) is an audience-driven approach that defines information objects based on the information requirements of the users to develop a web application. Some extensions to the UML notation (Booch et. al., 1998) have been proposed by Conallen (Conallen, 1999a; Conallen,1999b) to make the UML language suitable for modeling Web applications. Finally, Web Modeling Language - WebML (Ceri et al., 2002) provides a methodology and a notation language for specifying complex Web sites and applications at the conceptual level and along several dimensions. WebML provides a model-driven approach to Web site development, which is a key factor for defining disciplined development processes for the construction of complex sites, supporting advanced features like multi-device access, personalization, and evolution management.
Most of the above methodologies are based on the key principle of separating data management, site structure and page presentation and provide formal techniques and means for effective and consistent development process, and a firm basis for re-engineering and maintenance of web applications. A drawback is that only a few of them have integrated personalization techniques into their Web application development framework. The provision of mechanisms for the support of personalized content delivery has been extensively investigated by some of them like WebML, OOHDM, WSDM and UML. (Rossi et al., 2001; Ceri et al., 1999) Innovative conceptual models for Web applications, with personalization features, rigorous methodologies for the construction of personalized Web sites, and novel tools for the development of personalized Web applications are a very promising research direction.

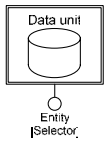
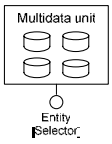
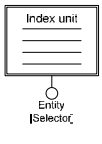
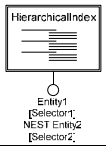
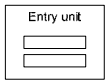
1.2 Motivation

Deploying a methodology for the design and development of a Web application enhances effectiveness, but does not guarantee optimization in the design process, mainly due to the fact that a small number of extreme designers/programmers exist (Boehm, 1981). Moreover, most applications are developed by large teams, leading to communication problems in the design/development process, often yielding products with large numbers of defects, causing serious problems of usability, reliability, performance, and other qualities of service. In most of the cases due to the lack of time, designers reuse their previous work and experience without trying to fully adapt it to the current project's requirements, resulting to a bad employment of reuse. There exists a tremendous need for restructuring/refactoring the applications, even in their conceptual level. Effective modeling must be treated as a first class citizen and be considered at early and all stages of the design process.
In the past, a number of research attempts have been conducted in the field of refactoring applications based on their design model. Most of them are focused on standalone software artifacts and deploy UML to perform refactoring in an application (Mens et al., 2002). Despite the popularity of model-driven methodologies there is an absence of assessment/analysis throughout the design and development process.
The goal of this paper is to argue the need to approach personalization aspects from the very beginning in the Web application's development cycle. Since personalization is a key issue in the application's success, it is important that it should be dealt through a design view, rather than only an implementation view.
In this paper we provide a methodology in order to evaluate the conceptual schema of an application, by means of the personalization features incorporated in the application model. We try to capture cases which have different design, but produce the same personalization effect. The evaluation is performed in several steps of inspection. A first level evaluation of the hypertext compositions used in the hypertext design, and a second level evaluation of the manipulation and presentation of data to the user. We provide metrics for the quality evaluation of the application's conceptual schema and a number of restructuring/refactoring rules to improve the final applications quality. The methodology can be deployed either in the process of designing an application or in the process of re-engineering it.
In this work, WebML has been utilized as design platform for the methods proposed, mainly due to the fact that it supports a concrete framework for the formal definition of personalization mechanisms and the fact that it is supported by a robust CASE tool called WebRatio (WebRatio, 2005). Most of the work presented here can be generalized and applied to applications utilizing other modeling languages, such as OOHDM or UML, with slight straightforward modifications.
The remainder of this paper is structured as follows: Section 2 provides a short overview of WebML and details about its personalization mechanism. Section 3 introduces the notion of model cloning, while Section 4 describes in detail the methodology for evaluating the personalization conceptual schema of an application. Finally, section 5 concludes the paper and discusses refactoring suggestions and future steps.

2 WebML: A Quick Overview

WebML is a conceptual model for Web application design (Ceri et al., 2002). WebML offers a set of visual primitives for defining conceptual schemas that represent the organization of the application contents and of the hypertext interface. These primitives are also provided with an XML-based textual representation, which allows specifying additional detailed properties, not conveniently expressible in the visual notation. For specifying the organization of data, WebML exploits the E-R model, which consists of entities, defined as containers of data elements, and relationships, defined as semantic connections between entities. WebML also allows designers to describe hypertexts, called *site views*, for publishing and managing content. A site view is a specific hypertext, which can be browsed by a particular class of users. Multiple site views can be defined for the same application.
Site views are internally organized into hypertext modules, called *areas*. Site views and areas are then composed of *pages*, and they in turn include containers of elementary pieces of content, called *content units*. Typically, the data published by a content unit are retrieved from the database, whose schema is expressed through the E-R model. The binding between the content units (and the hypertext) and the data schema is represented by the *source entity* and the *selector* defined for each unit. The source entity specifies the type of objects published by the content unit, by referencing an entity of the E-R schema. The selector is a filter condition over the instances of the source entity, which determines the actual objects published by the unit. WebML offers predefined units (such as: data, index, multidata, scroller, multichoice index, and hierarchical index – some of them are presented in Table 1) that express different ways of selecting entity instances and publishing them within the hypertext interface. WebML also provides a unit (called entry unit) representing entry forms for inputting data.
To compute its content, a unit may require the "cooperation" of other units, and the interaction of the user. Making two units interact requires connecting them with a *link*, represented as an oriented arc between a source and a destination unit. The aim of a link is twofold: permitting navigation (possibly displaying a new page, if the destination unit is placed in a different page), and enabling the parameter passing from the source to the destination unit.
Finally, WebML models the execution of arbitrary business actions, by means of operation units. An operation unit can be linked to other operation or content units. WebML incorporates some predefined operations (enabling content management) 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 elements. The whole set is described in (Ceri et al., 2002)

				
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 displays forms for collecting input data into fields

2.1 Personalization & WebML

WebML enables the actual construction of a hypertext embodying content and services personalized with respect to the user's features. Personalization is realized during various levels of the design process. In order to provide one-to-one content delivery the hypertext architect has to define the *personalization objects* in the process of data design (data personalization sub-schema), *personalization hypertexts* in the process of hypertext design (hypertext personalization sub-schema) and *presentation templates* fulfilling the presentation guidelines derived by customization policies related to personalization.
Personalization objects are used to incorporate into the data model the relevant properties of the user needed for personalization purposes. For example, entities may be used to model user profile data and the groups in which users are clustered. Relationships may be exploited to connect the user and group entities to the applicative entities, to represent aspects like object ownership or personal preferences. Personalization hypertexts are used for personalization purposes, like user identification, access rights management, personalized content delivery, and personal objects management.

In general, WebML provides a personalization model where users and groups can be explicitly specified in the structure schema of the information. More precisely, hypertext and data personalization sub-schemas may be used to represent the default rules for assigning users to the available site views. For instance, each user group may be connected to an instance of a site view entity, representing the available site views, to denote the semantic association between each group and the site view designed to fulfill the requirements of the group members.

3 The Notion of Model Cloning

Restructuring, *refactoring* and *code cloning* are well known notions in the software community. According to the reverse engineering taxonomy of (Chikofsky & Cross, 1990), *restructuring* is defined as: "… the transformation from one representation form to another at the same relative abstraction level, while preserving the subject system's external behavior (functionality and semantics). A restructuring transformation is often one of appearance, such as altering code to improve its structure in the traditional sense of structured design. While restructuring creates new versions that implement or propose change to the subject system, it does not normally involve modifications because of new requirements. However, it may lead to better observations of the subject system that suggest changes that would improve aspects of the system."

In the case of object-oriented software development the definition of *refactoring* is basically the same: "… the process of changing a software system in such a way that it does not alter the external behavior of the code, yet improves its internal structure" (Fowler et al., 1999). The key idea here is to redistribute classes, variables and methods in order to facilitate future adaptations and extensions. Finally, *code cloning* or the act of copying code fragments and making minor, non–functional alterations, is a well known problem for evolving software systems leading to duplicated code fragments or *code clones*. Code cloning can be traced by code smells that is, certain structure in code that suggests the possibility of refactoring (Fowler et al., 1999).

Roundtrip engineering has reached a level of maturity that software models and program code can be perceived as two different representations of the same artifact. With such an environment in mind, the concept of refactoring can be generalized to something improving the structure of software instead of just its code representation.

In this work we extend the notion of code cloning to the modeling level of a Web application. Analogously, to code cloning we introduce the notion of *model cloning* as the process of duplicating, and eventually modifying, a block of the existing application's model that implements a required functionality. This ad-hoc form of reuse occurs frequently during the design process of a Web application. We will try to capture *model smells* that is, certain blocks in the Web application's model that imply the possibility of refactoring.

4 Evaluating Personalization in the Conceptual level – The Methodology

The methodology presented in the sequel comprises of three distinct phases. In the first phase, we transform the Web application's hypertext model into a number of directed graphs, representing the navigation structure and the distribution of content within the areas and pages of the application. This forms the basis for an information extraction mechanism needed in the next phase. In the second phase, we extract, utilizing graph mining techniques, potential model clones and information related to the navigation and semantics of the application, while in the third phase we provide evaluation metrics on the need for model refactoring.

4.1 Initialization Phase

In this phase we preprocess the hypertext model, in order to extract the potential model clones and the evaluation parameters. More precisely, consider an application having a number of site views that are utilized to provide personalization features to the end users. First, we convert every site view into a directed graph based on the notation described below, thus constructing a first set of graphs representing the navigation and 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, areas, etc). Correspondingly, Σ_E comprises of all the different kinds of links (contextual, non contextual, transport & automatic). Besides the predefined WebML links, we introduce a special kind of edge (labeled *'c'* in the graph) in order to represent the *containment* of content units or sub-pages in pages, as well as pages, sub-areas and operation units in areas. Note that arbitrary containment

sequences can exist. Thus, we demand that units in WebML do not exactly correspond to nodes in V and the same is true for links between units and edges in E. A transformation example is depicted in Figure 1 (Transformation A), where we transform a page containing a data, an index and a multidata unit, connected with contextual links.

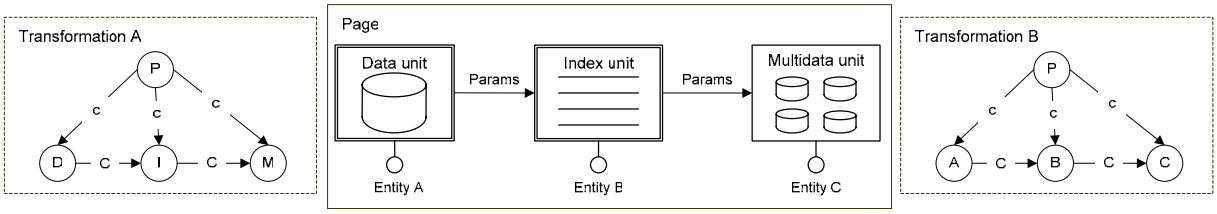


Figure 1: Transformation of a WebML hypertext composition to its graph equivalents

Following a similar procedure, for every site view of the hypertext schema we create a second graph representing the data distribution within each area, sub-area and page, thus constructing a second set of graphs. More precisely, in this case we define a site view as a directed graph, $Q(N, L, f_N, f_L)$, comprising of a set of nodes N, a set of edges L, a node-labeling function $f_N: N \rightarrow \Sigma_N$, and an edge-labeling function $f_L: L \rightarrow \Sigma_L$. f_N assigns letters drawn from an alphabet Σ_N to the site view nodes, whereas f_L has the same role for links and the edge alphabet Σ_L. Σ_N has a different letter for each different source entity used by the WebML elements comprising the hypertext schema, as well as for the pages, sub-pages, areas and sub-areas of the site view. Σ_L comprises of all the different kinds of WebML links in order to model the context navigation within the hypertext schema. As in the previous transformation, we also introduce edges denoting containment. A transformation example is depicted in Figure 1 (Transformation B).

4.2 Selection and Categorization of Potential Model Clones

Having modeled, utilizing graphs, the navigation, content presentation and manipulation mechanisms of the application, as well as the data distribution within each site view, in this phase we try to capture model smells in order to facilitate the quality evaluation of the whole applications personalization schema and provide metrics for possible refactoring.

4.2.1 Mining at the Hypertext and Data Level

In the sequel, we traverse the first set of graphs constructed in the previous phase, in order to locate identical configurations of hypertext elements (subgraphs) along with their variants, either within a graph representing a single site view or among graphs representing different site views. The variants include all the modalities in which the configuration retrieved starts and terminates. That is all the nodes or sets of nodes in the graph passing the context and receiving context from the hypertext configuration. For every instance of the configuration we also retrieve its variants. The recovery of the various configurations can be achieved using graph mining algorithms. Intuitively, after modeling the site views as directed graphs the task is to detect frequently occurring induced subgraphs. The problem in its general form is synopsized to finding whether the isomorphic image of a subgraph in a larger graph exists. The latter problem is proved to be NP-complete (Garey & Johnson, 1979). However, quite a few heuristics have been proposed to face this problem. The most prominent approaches include *gSpan* (Yan & Han, 2002), *CloseGraph* (Yan & Han, 2003) and *ADI* (Wang et al., 2004). Any of the above approaches can be utilized to perform the extraction of the hypertext configurations.

Likewise, employing the same graph mining techniques, we traverse the second set of graphs in order to locate identical configurations of data elements (source entities) along with their variants.

Finally, we try to locate compositions of identical hypertext elements referring to exactly the same content and being interconnected with different link topologies. Ignoring the edges in the first set of graphs, except from those representing containment, we mine identical hypertext configurations within a graph or among graphs. Then, we filter the sets of subgraphs acquired; utilizing the information represented in the second set of graphs (source entities), and keep the compositions referring to exactly the same data.

4.2.2 Categorization of Potential Model Clones

For every instance of the hypertext configurations mined in the first case of the previous phase, we make a first level categorization according to the source entities and attributes that the WebML elements of the configurations refer to. To accomplish that, we utilize the information provided by the XML definition of each site view, where the source entities and the selectors of each element included in the site view are described in detail. More precisely, for a specific configuration retrieved, we categorize its instances in the various site views of the application according to the following criteria:
Configurations constituted by WebML elements referring to:

- exactly the same source entities and attributes.
- exactly the same entities and differ to some of the attributes.
- some of the source entities are the same (i.e. Figure 2, instances A and B).
- different source entities.

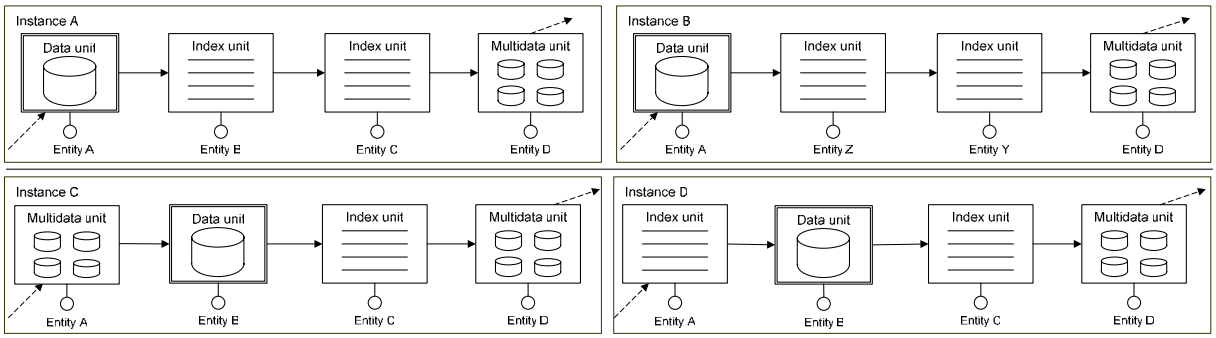


Figure 2: Categorizing potential model clones

We also categorize (exploiting the site views XML definition) every instance of the data element configurations acquired by the graphs representing the data distribution as:
Configurations constituted by source entities utilized by:

- different WebML elements.
- similar WebML elements (of the same type: composition or content management) e.g. in the instances C and D of Figure 2, entity A is utilized by two composition units, a multidata and an index.
- exactly the same WebML elements.

The last category captures exactly the same hypertext configurations as the first case of the previous categorization. Finally, potential model clones are the sets of hypertext configurations retrieved in the third step of the previous phase, where compositions of identical WebML elements referring to common data sources, utilizing different link topologies, have been retrieved.

4.3 Metrics and Quality Evaluation

Having identified the potential model clones based on the hypertext and data model of the application, we provide metrics for the categorization through quality evaluation of the application's personalization conceptual schema. For the various model clones identified, we introduce factors, denoting possible need for refactoring. The evaluation is threefold: First, we evaluate the hypertext compositions used in the hypertext design by means of consistency, based on their variants throughout the conceptual schema. Second, we quantify the categorization presented in 4.2.2 based on the similarity and the independence among potential model clones and third we make an evaluation based on the link topology.

4.3.1 Consistency of the Hypertext Schema

(Fraternali et al., 2002) introduce a methodology for the evaluation of predefined WebML design patterns consistent application, within the conceptual schema of an application. We utilize this methodology and extend it, in order to

introduce metrics depicting the consistent application of the design solutions retrieved during the first step of the procedure described in section 4.2.1 throughout the application's conceptual schema. 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. We call them respectively *Start-Point Metric (SPM*) and *End-Point Metric (EPM)*. The definition of SPM follows (EPM is defined analogously):

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

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

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

where 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 the best case variance, and it is calculated using again equation 2, assuming that only one variant has been coherently used throughout the application.

The last step in the metrics definition is the creation of a measurement scale, which defines a mapping between the numerical results obtained through the calculus method with meaningful discrete values. According to the scale types defined in the measurement theory (Roberts, 1979), 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 the following table (Table 2). The same scale covers EPM metric too.

Table 2: The measurement scale defined for the SPM metric

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

4.3.2 Similarity and Independence of Hypertext Elements and Data Entities Combinations

To evaluate the similarity between two hypertext configurations instances (potential model clones either with the same entities or the same WebML elements), we adopt the vector model and compute the degree of similarity. To convert these configurations into vectors in the space model we define the vector $\vec{d}_i = (x_{i1}, x_{i2}, x_{i3} \ldots x_{in})$. The compounds include the distinct WebML elements or entities of all configurations involved (for instance in figure 2, instances A and B, there are 6 distinct entities involved and in the same figure, instances C and D, there are four distinct WebML elements involved in each couple of configurations compared). These compounds are considered as uni-grams and are weighted by the frequency φ of this uni-gram (correspondingly). In the case of entities that have different attributes involved in a page (usually personalized instances of a page), the uni-gram of the corresponding entity is transformed to a fraction of frequency φ proportionate to the number of the participating attributes as opposed to the number of all designed attributes. The vector model proposes to evaluate the degree of similarity of a configuration d_m and a configuration d_μ (with the same entities or WebML elements) as the correlation between the vectors $\vec{d}_m$ *and* $\vec{d}_\mu$. This correlation can be quantified by the cosine angle between these two vectors, that is:

$$similarity(d_m, d_\mu) = \cos(\vec{d}_m, \vec{d}_\mu) = \frac{\vec{d}_m \cdot \vec{d}_\mu}{|\vec{d}_m| \times |\vec{d}_\mu|} = \frac{\sum_{i=1}^{t} w_{i,m} \times w_{i,\mu}}{\sqrt{\sum_{i=1}^{t} w_{i,m}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,\mu}^2}}, \in [0,1]. \quad (3)$$

where $|\vec{d}_m|$ and $|\vec{d}_\mu|$ are the norms of two configurations.

Potential clones can be categorized as relevant or not using the vector space model that ranks them according to the degree of similarity to each other.

One additional uni-gram compound is included (see equation 5) called clone *independency*. Based on (Chien, 1997), a potential clone is independent when the entropy of its context is high (i.e. the left and right contexts are random enough). Intuitively, clone independency extends clone consistency and distinguishes between potential model clones and parts of model clones that belong to a broader configuration. We use *IND* to measure the independence of the potential clones. IND_l, which is the left context independence value can be computed by equation 4, where *0log0 = 0* is defined. In our case, we only consider one adjacent element as context.

$$IND_l = -\sum_{t=l(d)} \frac{\phi(t)}{TF} \log \frac{\phi(t)}{TF} \quad (4)$$, where *TF* is the *Total Frequency*.

The IND_r value for right context could be calculated similarly. The final *IND* value is the average of those two.

$$IND = \frac{IND_l + IND_r}{2} \quad (5)$$

In the following, we propose the different thresholds and corresponding categories on the *similarity*(d_m, d_μ), always having in mind the potential clones *independency*. This second stage of our evaluation approach uses as input the instances of the hypertext configurations retrieved. The goal of this stage is to check whether instances can be actually considered clones, and to identify the opportunities of refactoring. Results of this step are depicted in ordinal scaled categories presented below.

As a first step, configuration instances retrieved by the first set of graphs are classified according to the clone classification scheme shown in Table 3.

Table 3: Classification of potential model clones based on the hypertext model

Level	*Configurations of WebML elements referring to:*
1	the same source entities and attributes.
2	source entities that up to 75% of them are the same.
3	source entities that up to 50% of them are the same.
4	source entities that up to 25% of them are the same.
5	different source entities.

Classification follows an ordinal scale based on the degree of similarity between the potential clones. When the similarity degree increases the level of potential model clones successful detection decreases. Therefore, the higher the level's value is, the smallest the probability to have identified a potential model clone gets.

Following, configuration instances retrieved by the second set of graphs are classified according to the clone classification scheme shown in Table 4.

Table 4: Classification of potential model clones based on the data distribution

Level	*Configurations constituted by source entities utilized by:*
1	WebML elements of the same type, which are the same.
2	WebML elements of the same type, which up to 75% are the same.
3	WebML elements of the same type, which up to 50% are the same.
4	WebML elements of the same type, which up to 25% are the same.
5	different WebML elements.

Likewise, classification follows an ordinal scale based on the degree of similarity between the candidate model clones. Checking proceeds, from level 1 to 5 and stops when a level can be recognized. Again, the probability to have identified a model clone increases as the level's value decreases.

4.3.3 Links Topology Evaluation

To further refine the evaluation procedure, the hypertext configurations of candidate model clones, which exist within a page, are examined in this phase based on their link connectivity-topology. Intuitively, similarity of link topology in a local (within a page) or broader (within an area or site view) modeled hypertext implies similarity in the business logic. When the differences are broader, then either the business logic is changed or potential

misconfigurations are detected. We define as *link topology similarity* the cosine angle between vectors as the following: $\vec{v}_i = (x_{i1}, x_{i2}, x_{i3} \ldots x_{in})$, where n are the different link connections in the hypertext model in a group of candidate clones with the same (instances and number) of WebML elements. The uni-gram compounds receive values in {1,0,-1}, respectively when the link is of the same direction, if it does not exist at all or if the link creates an opposite direction element connection. As a result, we classify the hypertext configurations retrieved in the third step of the mining phase, in accordance to their link topology similarity as depicted in Table 5.

Table 5: Classification of potential model clones based on the links topology

Level	*Description*
1	Very high link topology similarity. (equivalent)
2	75% link topology similarity. (high)
3	50% link topology similarity. (semi)
4	25% link topology similarity. (low)
5	Different link topology. (disjoint)

In this case too, we follow an ordinal scale for the classification scheme.

5 A Concluding Discussion on Refactoring – Future Work

After having classified the potential model clones according to the various similarity levels, the hypertext architect has a first set of metrics pointing out hot spots for potential improvements in the personalization conceptual schema of the application. The potential model clones are ranked according to their similarity level and their size (number of elements constituting the configuration), thus forming a valuable tool providing quality improvement guidelines for the overall model. The methodology can be extended to support automatic model refinement, but there are a number of reasons that impose the need for the designer's intervention and visual inspection of the model. If an automated model refactoring process would be deployed checks for the syntactic and semantic correctness (e.g. conflicts, racing conditions, deadlocks) should be performed. Moreover, since personalization is realized through various levels of the design process (e.g. links or content personalization) automatic model refactoring could lead to inconsistencies with respect to the application's initial functional requirements. Nevertheless, a number of refactoring suggestions are straightforward.

Based on the first classification, the existence of two or more model clones belonging to Level 1 implies a very high refactoring opportunity. The designer should keep only one of the model clones, utilizing the "change siteview" unit if the clones are in different site views, or reconsider the link topology if the clones are in the same site view. In case that the WebML elements of two or more clones refer to exactly the same entities and differ to some of the attributes, merging all the attributes in every element should be considered (if the content personalization is not affected). If the clones are categorized from levels 2 to 4, the designer should consider refactoring the hypertext schema by merging pages or areas. Refactoring can also be performed when SPM and/or EPM metrics values are under 0.6, by applying where possible the variant having the larger occurrence frequency.

The various similarity levels computed during the second classification denote cases where the same data are presented to users (or manipulated by them) by means of different presentation (or content management) mechanisms. High similarity between clones implies hot spots where the designer should examine redesigning the hypertext in order to accomplish consistency in the presentation level and enhance usability of the final application.

Analogously, the last classification identifies cases in which, the link topology of segments of the application model should be reconsidered in order to accomplish consistency in the navigation and presentation level.

Overall, in this paper we have illustrated a methodology that aims at capturing potential problems, by means of personalization within the conceptual schema of a Web application. We have introduced the notion of model cloning and provided a technique for identifying model clones within an application's hypertext model. Moreover, we have supplied evaluation metrics quantifying the "inappropriate" reuse of clones and proposed refactoring recommendations. Applying the methodology can result to a higher degree of consistency in the personalization schema of a Web application, as well as enhancement during the design process and an elevation of usability.

In the future we plan to apply the methodology to a large number of Web applications personalization schemas, in order to refine it and fine-tune the model clone evaluation metrics. Moreover, we intend to extend the methodology in order to support automatic model refinement at some level (syntactic and semantic correctness) based on the evaluation results.

References

Atzeni, P., Mecca, G., & Merialdo, P. (1998). Design and Maintenance of Data-Intensive Web Sites. *In the Proceedings of EDBT*, 436-450.

Boehm, B. (1981). Software Engineering Economics. Prentice Hall PTR.

Booch, G., Jacobson, I., & Rumbaugh, J. (1998). The Unified Modeling Language User Guide. The Addison-Wesley Object Technology Series.

Ceri, S., Fraternali, P., & Paraboschi, S. (1999). Data-Driven One-To-One Web Site Generation for Data-Intensive Applications. *In Proc. of VLDB'99*. Morgan Kaufmann.

Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., & Matera, M. (2002). Designing Data-Intensive Web Applications. Morgan Kauffmann.

Chien L. F. (1997). PAT-Tree-Based Adaptive Keyphrase Extraction for Intelligent Chinese Information Retrieval. *In the Proceedings of the 20th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pp. 50-58.

Chikofsky, E., & Cross, J. (1990). Reverse engineering and design recovery: A taxonomy. *IEEE Software*, 7 (1), 3-17.

Conallen, J. (1999). Building Web Applications with UML. Addison-Wesley, Reading MA.

Conallen, J. (1999). Modeling Web application architectures with UML. *Communications of the Association for Computing Machinery*, 42 (10), 63–70.

De Troyer, O., & Leune, C. (1997). WSDM: A user-centered design method for Web sites. *In the Proceedings of the 7th International World Wide Web Conference (WWW'7)*.

Fernandez, M. F., Florescu, D., Kang, J., Levy, A. Y., & Suciu, D. (1998). Catching the Boat with Strudel: Experiences with a Web-Site Management System. *In the Proceedings of ACM-SIGMOD Conference*, 414-425.

Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (1999). Refactoring: Improving the Design of Existing Code. The Addison-Wesley Object Technology Series.

Fraternali, P., & Paolini, P. (1998). A Conceptual Model and a Tool Environment for Developing More Scalable, Dynamic, and Customizable Web Applications. *In the Proceedings of EDBT 1998*, 421-435.

Fraternali, P., Matera, M., & Maurino, A. (2002). WQA: an XSL Framework for Analyzing the Quality of Web Applications. *In the Proceedings of IWWOST'02*, Malaga, Spain.

Garey, M., R., & Johnson, D., S. (1979). Computers and Intractability: A guide to NP-Completeness. New York: Freeman.

Garzotto, F., Paolini, P., & Schwabe, D. (1993). HDM - A Model-Based Approach to Hypertext Application Design. *TOIS*, 11 (1), 1-26.

Isakowitz, T., Stohr, E., & Balasubramanian, P. (1995). RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38 (8), 34-44.

Mens, T., Demeyer, S., Du Bois, B., Stenten, H., & Van Gorp, P. (2002). Refactoring: Current research and future trends. *Language Descriptions, Tools and Applications (LDTA)*.

Roberts., F.S. (1979). Measurement Theory with Applications to Decision Making, Utility and the Social Sciences. Addison Wesley.

Rossi, G., Schwabe, D., & Guimaraes, M. (2001). Designing Personalized Web Applications. *In the Proceedings of the 10th International World Wide Web Conference (WWW'10)*.

Schwabe, D., & Rossi, G.(1998). An Object-Oriented Approach to Web-Based Application Design. *Theory and Practice of Object Systems (TAPOS)*, 4 (4), 207-225.

Wang, C., Wang, W., Pei, J., Zhu, Y., and Shi, B. (2004). Scalable Mining of Large Disk-based Graph Databases. *In the Proceedings of ACM KDD04*, 316-325.

WebRatio. http://www.webratio.com

Yan, X., & Han, J. (2002). gSpan: Graph-based substructure pattern mining. *In the Proceedings of International. Conference on Data Mining (ICDM'02)*, Maebashi, 721-724.

Yan, X., & Han, J. (2003). CloseGraph: mining closed frequent graph patterns. *In the Proceedings of ACM KDD03*, 286-295.