Revising Web Design to Deal with Current Development Practices

Pejman Sajjadi **Olga De Troyer**

Vrije Universiteit Brussel, Department of Computer Science,
Research Group WISE, Pleinlaan 2
1050, Brussels, Belgium
Email: {Ssajjadi,Olga.DeTroyer}@vub.ac.be

Abstract

Website design practice has evolved a lot. Websites are no longer designed from scratch but based on existing designs and patterns, and very often they are implemented using Web Content Management Systems (WCMS). Given all this support, it seems that the traditional web design methods, developed in the past, are no longer relevant. However, we argue that they still have a role to play, but they should be adapted to the new development practice and exploit this during the design process to make the development of a website easier and faster while still guaranteeing a well designed and usable website. In this paper, we show how an existing website design method, WSDM, has been adapted to take current development practice into consideration while still respecting the core principles and benefits of the method.

Keywords: Website design, website design methodology, WSDM, WSDM-Lite, design pattern, website genre, web content management system.

1 Introduction

Website design methodologies were created to provide a systematic and scientific approach to design websites. Numerous website design methodologies were developed over the years, including OOHDM (Schwabe & Rossi 1995), WebML (Ceri et al. 2000), UWE (Hennicker & Koch 2000), OOWS (Pastor et al. 2006), and WSDM (De Troyer & Leune 1998). They all consider several design phases in which various modeling techniques are used in order to model the different aspects of the website.

Over the years, Web technology has evolved a lot. While in the early years of the Web, it was easy to create a website (with a few lines of HTML), it became more and more complex due to an increase in complexity of the web technology. This has resulted into the development of Web Content Management Systems (WCMSs), which had, originally, the intention of shielding web developers from the increased complexity.

We have also seen an evolution in how websites are designed and developed. While in the beginning, websites were designed and handcrafted individually, resulting in websites with their own identity, nowadays websites targeting similar goals and audiences all look more or less similar (from a structural and functional point of view). Over the years, different so-called website genres have emerged. Each website genre targets a specific goal and audience. Examples are webshops and news websites. The websites of a particular genre all display similar features, provide the same kind of information, and are structured in a similar way. E.g., all webshops share functionalities such as a shopping cart, product comparison, and checkout. On first sight, this evolution may suggest that website design methods are of no use anymore for this type of websites. Moreover, many WCMSs offer predefined functionality and templates for different website genres. This may strengthen the suggestion that creating a website of a certain genre is just a matter of some clicks and filling in the content.

While using predefined templates could be an acceptable solution for small and simple websites, developing professional websites simply by copying or mimicking existing websites or by adapting some predefined templates is not advisable because a careful analysis of the requirements will be omitted, design decisions will not be given due consideration, and design errors or bad practices (i.e., the so-called anti-patterns[1] and dark patterns[2]) will be taken on. Such a practice may result in websites that are badly designed and are therefore difficult to maintain. In addition, some of the required functionalities may not be available in the predefined template, or a desired structure may not be possible.

For professional websites, website design methods are still necessary, even for websites of a certain genre and for websites implemented by means of a WCMS. However, website design methods should take into consideration the existence of website genres, web design patterns, and the existence of implementation tools such as WCMSs. Furthermore, the "template" concept used by WCMSs, could also be useful during modeling. Having high level specifications (i.e., models) of website genres could be useful. First, because it would be clear what features and functionalities should and could be included in a website belonging to a particular genre, and secondly, having a detailed model of the high level design for a website genre could make the design process faster as there is no need to start from scratch. In this paper, we show how we can take advantage of website genres during website design. A website genre will be considered as a high-level design pattern composed of other lower-level design patterns. To describe such high-level design patterns, i.e., to model them, a technique borrowed from software variability modeling (Kang et al. 1990), i.e., feature assembly (Zaid et al. 2010), is used. Furthermore, we explain how an existing website de-

[1] http://goo.gl/uj5dmc
[2] http://darkpatterns.org/

sign method, WSDM (De Troyer et al. 2008), can be adapted to this new approach for designing websites. The same principles can be applied to other website design methods.

We also explain the tool support developed for this new version of WSDM, called WSDM-Lite. WSDM-Lite targets web developers who are familiar with modeling and want to use a WCMS for the implementation of the website, but need a website that goes beyond the out of the box products offered by those WCMSs. We do not target casual users who want to develop a simple website, nor users who are very experienced in web development and want to have full control over the implementation.

The rest of the paper is structured as follows. Section 2 discusses website genres and design patterns and how they can be modeled for their use in website design methods. Section 3 shows how they can be taken into consideration when using a website method such as WSDM. Section 4 provides a demonstration and section 5 discusses tool support. Section 6 reviews related work and section 7 concludes the paper.

2 Website Genres and Design Patterns

Various sources have categorized websites into genres, e.g., Wikipedia[3], (van Welie & Van der Veer 2003), (Van Duyne et al. 2007), (Lindemann & Littig 2011), and (Dawson 2010). The strategies behind the categorization vary from source to source, but certain genres appear in all resources, e.g., e-commerce websites and social networks. Furthermore, a number of resources provides long lists of design patterns, e.g., (Yahoo 2014) and (Scott & Neil 2009). Most of these patterns focus on the user interaction.

To describe a website genre we use the notion of design pattern. In software engineering, a design pattern is used to define a generic solution for a reoccurring problem. We use the notion of design pattern at two levels: (1) to describe a website genre and (2) to describe the different features usually available in websites of a certain genre. We explain this further.

First, a website genre is defined by means of a pattern, called *a website genre pattern*. Next to the usual textual description given for a design pattern, we also provide a formal model for the pattern by providing a model for the features usually available in websites of the genre. The individual features are also described as patterns, i.e., by *feature patterns*. For example, the e-commerce website genre pattern includes the Login, Shopping cart, and Products feature patterns.

To provide an overview of the features usable for a certain website genre, and to describe which features are considered mandatory and which are optional, as well as the relationship between the features, we use a modeling technique borrowed from software variability modeling, i.e., feature modeling used to model software product lines. A software product line represents a family of software products, rather than a single software product. Feature modeling is used to model the common and variable features in the software products, as well as the dependencies between those features (Kang et al. 1990).

For our purpose, a website genre can be considered as a family of websites and thus as a software product line. Furthermore, the different feature patterns appearing in such a website genre can be considered as the features of the software product line. By modeling a website genre as a software product line, we can specify which features can appear in a website genre, which are mandatory and which are optional, and which dependencies (e.g., "Requires" and "Excludes") exist between the features. We use the Feature Assembly technique (Zaid et al. 2010) to do so. A website genre design pattern is defined as follows:

Definition 1
A *website genre design pattern* is a triple $\mathcal{WG} = (\mathcal{D}, \mathcal{E}, \mathcal{FM})$, where:
$\mathcal{D}$ is a textual description of the website genre;
$\mathcal{E}$ is a set of website examples of the website genre;
$\mathcal{FM}$ is a feature model describing the features usually available in a website of the website genre, their variability and their dependencies.

We omit the formal definition for the feature model; it can be found in (Zaid 2013). In practice, a graphical representation is used for a feature model (see Figure 1 for an example). In Feature Assembly, we distinguish between Concrete Features, i.e., concrete and well-defined characteristics of the software products (represented by a solid line rectangle) and Abstract Features, i.e., virtual features that represent a generalization of certain characteristics or capabilities of the software products (represented by a dotted line rectangle). In Figure 1, Login and Shopping cart are examples of concrete features and Payment method is an abstract feature. An abstract feature is associated with more specific feature(s) (i.e., variants) of which it is a generalization (represented by an arrow pointing from the variant to the abstract feature). For example, Bank transfer, Credit/Debit card, and PayPal are three variants (or options) of the abstract feature Payment method. A concrete feature may be further decomposed into sub-features, representing whole-part relations. The composition relation is graphically represented in the same way as a UML aggregation. A solid line expresses a mandatory sub-feature, while a dotted line represents an optional sub-feature. In Figure 1, the Like & Share feature is an optional feature while the Identity information feature is mandatory. Feature dependencies are represented by means of a graphical symbol on a line connecting the features, e.g., Shopping experience "requires" Product.

As an example, the e-commerce website genre pattern can be defined as follows:
Description ($\mathcal{D}$): A website that provides a virtual store where visitors can browse for goods, inspect them, and buy them. The website can support different online payment methods (e.g., bank transfer, credit/debit cards). Visitors can add items to a shopping basket and decide to purchase them at checkout.
Examples ($\mathcal{E}$): Amazon[4], ebay[5], Nike[6].
Feature Model ($\mathcal{FM}$): Given in Figure 1.

Next, each feature used in the feature model must be described by a feature pattern. The format for describing a feature pattern is very similar to the regular format used for software design patterns.

Definition 2
A *feature pattern* is a 5-tuple $\mathcal{F} = (\mathcal{D}, \mathcal{E}, \mathcal{W}, \mathcal{PS})$, where:
$\mathcal{D}$ is a textual description of the intention of the feature pattern;
$\mathcal{E}$ is a set of example websites where the feature

[3] http://goo.gl/jXIorO

[4] http://www.amazon.com/
[5] http://www.ebay.com/
[6] http://www.nike.com/

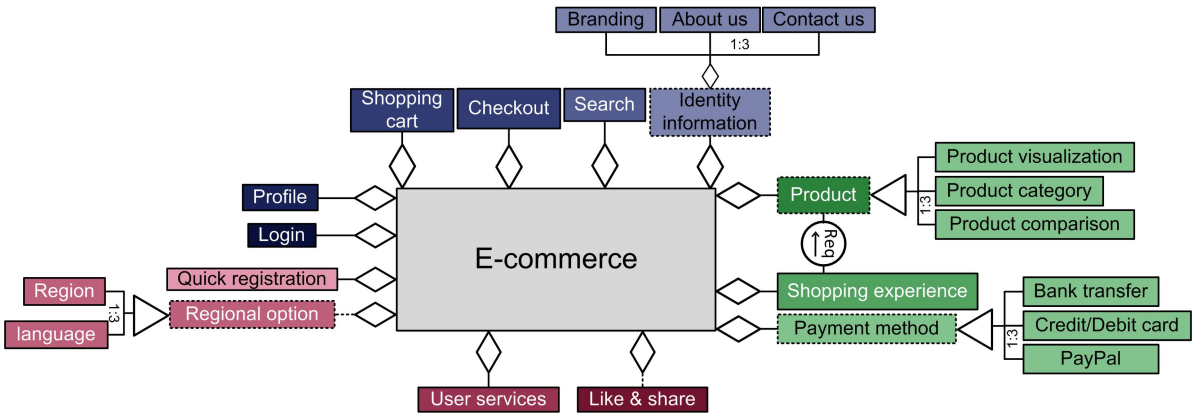

Figure 1: Feature Model for the e-commerce website genre pattern

pattern has been used;
$\mathcal{W}$ is a textual description of when it is appropriate to apply the feature pattern;
$\mathcal{PS}$ is a (possibly empty) set of tuples $(\mathcal{P}, \mathcal{S})$. Each tuple describes a possible problem $\mathcal{P}$, which may arise when using the feature pattern, and the associated solution $\mathcal{S}$. Both $\mathcal{P}$ and $\mathcal{S}$ are textual descriptions.

No formal model is given for a feature pattern. The reason for this is that many WCMSs may provide functionalities that implement a feature pattern (usually called components or modules). Although different components intend to implement the same feature pattern, they may be (slightly) different, as different websites may have (slightly) different requirements. In general, it will not be possible to provide a model for a feature that conforms to all possible components or that is general enough to serve all situations.

An example feature pattern is the Shopping cart:
Description ($\mathcal{D}$): This is a tool provided in e-commerce websites to assist the user with the purchasing process. It imitates the shopping cart of a real shop: it shows the items added to the cart, and items can be easily added and removed. The total expenses are calculated in real time.
Example($\mathcal{E}$): See http://goo.gl/QKoqxn.
When to use ($\mathcal{W}$): Shopping carts are a key feature of e-commerce websites.
Possible problem($\mathcal{P}$): The state of the page and user's navigation should not be changed after each item deletion. The "add item to the cart" function should not overwhelm the overall interface of the website since this might be confusing. It should be possible to inspect the shopping cart content and total expenses while shopping.
Solution ($\mathcal{S}$): Use technologies that allow partial updates of the page. Use "hover over item" functions to show/hide the "add item to cart" feature.

3 Considering Website Genres, Design Patterns, and WCMSs in WSDM

WSDM has been tailored to deal with website genres, design patterns, and WCMSs. First, a brief description of the original method is given. Next, we describe how the method has been adapted. Figure 2 shows the phases of the revised method. The newly added phases are indicated in black, the original ones in white, and modified phases are gray. The output of each phase is indicated (on the right side). Although this revised method has two additional phases, the phases are lighter in terms of work and time required, making the complete method easier and faster. For simplicity, the different phases are shown sequentially, but in practice the design process may be iterative.

In the first phase of the original method, the mission statement for the website is formulated. The goal is to identify the purpose of the website, as well as the subject and the target users, i.e., the users that the website wants to address. The subject is related to the purpose and the target users. It must allow fulfilling the purpose of the website and it must be adapted to the target users. The output of this phase is the mission statement, formulated in natural language. It enables designers (in the following phases) to decide which information or functionality to include or exclude, how to structure it, and how to present it.

The next phase is the Audience Modeling phase. The target users identified in the mission statement are refined and classified into so-called audience classes. The classification is based on the requirements of the different users. For each audience class relevant characteristics (e.g., age) are also given.

The next phase, the Conceptual Design, is used to specify the information, functionality, and structure of the website at a conceptual level. The information and functionalities are specified during the Task & Information Modeling sub phase. The overall conceptual structure including the navigational possibilities for each audience class, is specified during the Navigational Design sub phase.

During the Implementation Design phase, the conceptual design models are complemented with information required for an actual implementation. The phase consists of three sub phases: Site Structure Design, Presentation Design, and Logical Data Design. During Site Structure Design, the conceptual structure of the web system is mapped onto pages. For the same Conceptual Design, different site structures can be defined, targeting different devices, contexts, or platforms. The Presentation Design is used to define the look and feel of the website as well as the layout of the pages. The Logical Data Design is only needed for data-intensive web systems. If the data will be maintained in a database, a database schema is constructed (or an existing one can be used) and the mapping between the conceptual data model and the actual data source is created. Evidently, other types of data source are possible (e.g., XML, RDF).

The last phase is the Implementation. The actual implementation can be generated from the information collected during the previous phases.

We now describe how WSDM has been adapted to deal with website genres and feature patterns. Note that the method is still usable if no website genre is considered or applicable. In this paper, we focus on

the version that targets an implementation by means of a WCMS. This version is called *WSDM-Lite*. Note that it is also possible to have a version where the implementation is done by means of a code framework. However, this is outside the scope of the paper.

The first phase is still the Mission Statement Specification. The second phase, the Site Genre Identification, is new. In this phase, the designer identifies the website genre for his website (if applicable). For this, the designer can inspect a list of available website genres (see e.g., http://wise.vub.ac.be/dpl/ for such a list). It is also possible to select two or more website genres and combine them. The feature model associated with the website genre gives the designer an already well-defined model of the necessary and optional features for the website. However, this does not mean that the website is not allowed to have any other features. The output of this phase is the website genre pattern(s) to be used as the starting point.

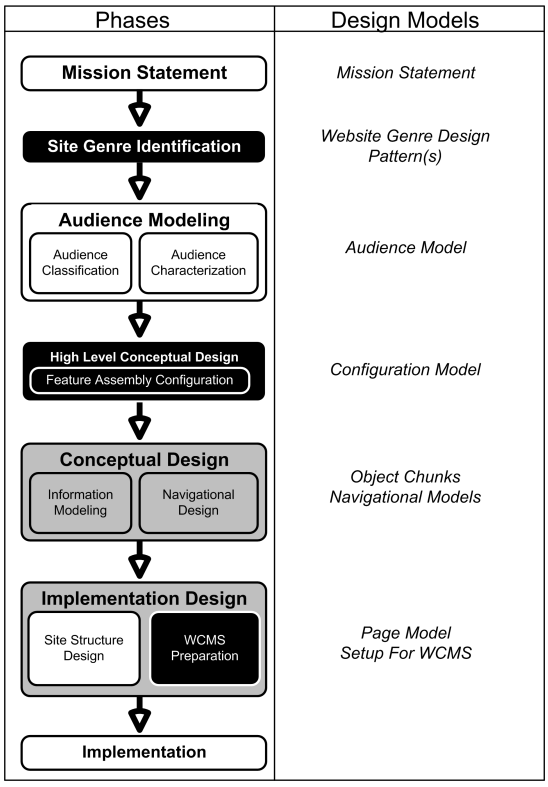

Figure 2: WSDM-Lite

The Audience Modeling is not changed. The next phase, the High Level Conceptual Design, is new. In this phase, the designer has to determine which features from the website genre's feature model(s) should be included in the design. It is also possible that functionality is required that was not foreseen in the website genre's feature model(s) but was formulated in the requirements of one, some, or all of the audience classes (i.e., the website extends beyond the website genre). These functionalities should be described as features. Furthermore, it is necessary to specify which features are required for which audience classes (because of the audience-driven nature of WSDM). Therefore, the features are mapped onto the requirements of the different audience classes (which are formulated during the Audience Modeling). A feature may cover several requirements. The output of this phase is the configuration model.

The next phase is the Conceptual Design. This phase is similar in nature to the original one, but works at a different level. The purpose of conceptual modeling is to create models independent of the actual implementation. In the original WSDM version, the models to be created were very detailed. Elementary tasks and the information for these tasks were modeled in full detail using the modeling techniques ORM (Halpin 2006) and CTT (Paternò et al. 1997). For a large website, this resulted in many models, which was a lot of work. In addition, when implementing the website using a WCMS, most, if not all, features can be implemented with functionality provided by the WCMS. Very often, there was a mismatch between the level of detail in the models created and the level of control over the components, modules and plugins of the WCMS used to implement them. Therefore, it is useless to create these fine grained models when ready-made components of a WCMS will be used for the implementation. Therefore, we limit the modeling to a minimum. For each required feature, only a limited amount of information needs to be modeled, i.e., the information to display, and the information that can be entered or/and updated. For this purpose, an information modeling technique, such as ORM, can be used. As an example, consider the shopping cart feature. It is in general possible to find a predefined module (or a combination of modules) for a WCMS that implements this feature. Therefore, we only need to model the information about the products that should be displayed in the shopping cart (name, price, amount, ...).

As a WCMS may impose its own navigational structure on the predefined components, it is also useless to make a fine-grained navigational model as required in the original WSDM. Therefore, only the top-level navigational structure for the complete website and the navigational structure for each individual audience track (i.e., how the user is allowed to navigate between the different features) is modeled during the Navigational Design sub phase. The navigational modeling technique provided in the original WSDM can be used for this.

The next phase is the Implementation Design, which included, in the original method, Site Structure Design, Presentation Design, and Logical Data Design. When a WCMS is used, the Logical Data Design can be omitted as the WCMS usually takes care of the data storage. WCMSs also provide considerable support for the presentation, so in general the presentation design will be reduced to finding a suitable presentation theme for the WCMS. This means that the Implementation design in WSDM-Lite retains only the Site Structure Design, which is used to map the conceptual structure of the website (i.e., the different features and the navigational structure) onto pages. Also the selection of a suitable WCMS, suitable components, and a suitable presentation theme is part of the Implementation Design. In general, the WCMS selection will be based on the available support for implementing the required features and navigational structure.

The Implementation phase deals with the actual implementation of the web system, this means implementing the models using the selected WCMS, components, and presentation template.

4 Example

The method is demonstrated for an example bookshop website. The mission statement was formulated as follows: "To provide an interactive online book store to attract more customers by allowing customers

to search, select, compare, and purchase books.". The genre was determined as "e-commerce".

Next, the audience classes, their requirements, and their characteristics were identified. The following audience classes were identified (WSDM provides a method to do this): Visitors, Customers, and Managers. Because of space limitations, we focus on the audience class Customer. For Customer, the following requirements were identified (full description is omitted): Login, Manage Profile, Search book, Add Book to Shopping Cart, Check Out, View Book details, Compare Books, and Manage Order information. Next (in the High Level Conceptual Design phase), these requirements are mapped onto the features in the available e-commerce's feature model, e.g., Login is mapped onto the feature "Login", and Manage Profile onto the feature "Profile". We also need to determine whether other features are needed. No additional features were identified.

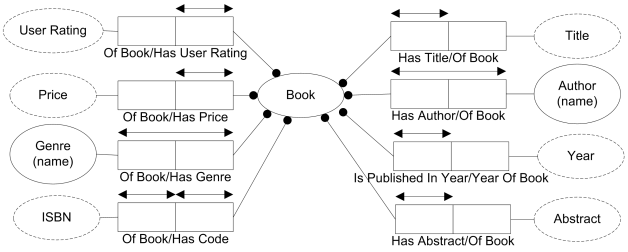

Figure 3: "View Book Details" model (ORM format)

In the Conceptual Design we model the details of the selected features. For example, we have to model the details for "View Book Details". The information model for this feature is given in Figure 3 and specifies that a book should have a title, at least one author, a year of publication, an abstract, a price, be of at least one genre, have an average user rating, and is identified by an ISBN. It is necessary to model this information as otherwise it is not known at implementation time which attributes are required for a book. Figure 4 shows the navigational model created for the audience class Customer. The rectangles represents components offering information and functionality and the arrows indicate the navigation possibilities.

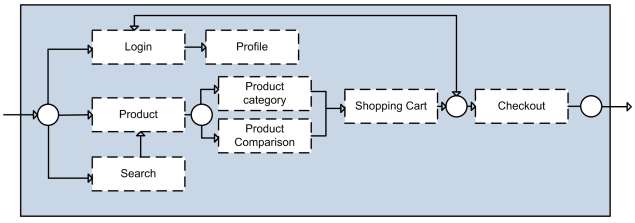

Figure 4: Customer's Navigational Track

During the implementation design, and after the analysis of available WCMSs, we decided to use WordPress. Next, we select the appropriate plugins and a proper theme for the presentation design. Finally, the website can be implemented using WordPress.

Note that when the original version of the WSDM would be used to model this website, for each single task (derived from a requirement), such as "View Book Details", two extra models would have been required: a task model using the CTT modeling technique and a task navigational model. To illustrate this extra effort, Figure 5 shows the task model and Figure 6 gives the task navigational model for the task "View Book Details". This means that when a website involves n tasks, 2n extra models are needed compared to WSDM-Lite.

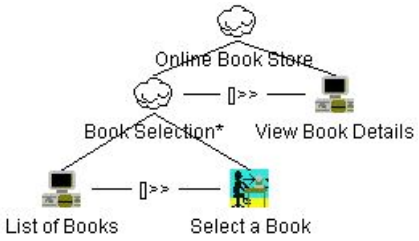

Figure 5: WSDM Task Model for "View Book Details"

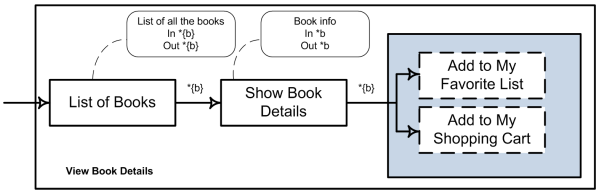

Figure 6: WSDM Task Navigational Model for "View Book Details"

5 Tool Support

A proof of concept support tool for WSDM-Lite was implemented. The tool is a web application and users can manage several projects. The designer is taken through all steps of WSDM-Lite in the proper order while he or she can see the overall progress, the current locus with respect to the entire methodology and has the possibility of navigating to any step instantly, which supports iteration. The tool supports the designer by generating automatically parts of models and by providing default models. As an example, based on the different audience classes defined by the designer and their associations with the required features, a default overall site structure of the website can be automatically generated and the designer only needs to adjust it, if needed.

The WSDM-Lite tool also provides two different pattern repositories, one for website genre patterns, and one for the lower level feature patterns. For these feature patterns, default information models can be provided. The designer can load such a default model and adapt it according to his needs.

6 Related Work

Some work, (van Welie 2014) and (Van Duyne et al. 2007), on website genres also identified commonalities across websites belonging to the same genre and also labeled them as "design patterns". Work most similar to the work presented in section 2, is the work in (van Welie & Van der Veer 2003). Design patterns are defined from high level to lower level in a hierarchical fashion; they explicitly mention five levels: Business goals, Posture level, Experience level, Task level, and Action level. They make a differentiation between patterns based on the category they belong to. A network of connected patterns is obtained with relationships among the patterns using object oriented modeling.

The WebML design methodology also incorporates the notion of design pattern into its website design method (Fraternali et al. 2008). The authors define a design pattern as follows: "A WebML hypertext design pattern is a WebML model that specifies a set of core hypertext elements and their organization in order to fulfill a defined task". As explained in section 2, we do not do this. Our feature patterns are of a higher level and used to specify features common to different websites. This work is similar to ours in the sense that patterns are used to capture a general picture of the constituents of a web application, but unlike WSDM, the design of the composing elements for each website genre pattern is not provided.

Also in (Rossi et al. 1997), design patterns are introduced for websites but on a different level. In their work, the definition of the concept is limited to interface and navigational design patterns.

Regarding the adaptation of website design methods to WCMSs, we can mention (Souer et al. 2007). In contrast to our approach, a new method for the implementation of web applications using a WCMS is proposed. This method is also for a specific WCMS. We extended an existing method and our method is independent of a WCMS.

7 Conclusion & Future Work

In this paper, we have exploited the fact that, more and more, websites belong to certain genres, where websites in a specific genre are providing similar features, and that many Web Content Management Systems (WCMS) exist that offer a lot of functionality to implement a website. This evolution has resulted in a perception that website design methods are no longer relevant. However, a proper design is still essential for well designed and usable websites.

Website genres, as well as individual features commonly appearing in such websites, are considered as design patterns. For a website genre, an overall model is provided to indicate the common and optional features and dependencies. Next, we described WSDM-Lite, which is an adaptation of the existing website design methodology WSDM. WSDM-Lite considers website genres during design and allows us to take advantage, during modeling, of the fact that a WCMS will be used for the implementation. In addition, tool support for WSDM-Lite was briefly discussed.

Future work is related to constructing a more complete library of website genres and related design patterns, and extending the tool support. Furthermore, user evaluations are planned.

References

Ceri, S., Fraternali, P. & Bongio, A. (2000), 'Web modeling language (WebML): a modeling language for designing web sites', *Computer Networks* **33**(1), 137–157.

Dawson, A. (2010), What kind of website should i make?, *in* 'Getting StartED Building Websites', Springer, pp. 1–63.

De Troyer, O., Casteleyn, S. & Plessers, P. (2008), WSDM: Web semantics design method, *in* 'Web engineering: Modelling and implementing web applications', Springer, pp. 303–351.

De Troyer, O. & Leune, C. J. (1998), 'WSDM: a user centered design method for web sites', *Computer Networks and ISDN systems* **30**(1), 85–94.

Fraternali, P., Tisi, M., Silva, M. & Frattini, L. (2008), 'Building community-based web applications with a model-driven approach and design pattern', *Handbook of research on Web* **2**(3.0).

Halpin, T. (2006), Object-role modeling (orm/niam), *in* 'Handbook on architectures of information systems', Springer, pp. 81–103.

Hennicker, R. & Koch, N. (2000), A uml-based methodology for hypermedia design, *in* ' UML 2000The Unified Modeling Language', Springer, pp. 410–424.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E. & Peterson, A. S. (1990), Feature-oriented domain analysis (foda) feasibility study, Technical report, DTIC Document.

Lindemann, C. & Littig, L. (2011), Classification of web sites at super-genre level, *in* 'Genres on the Web', Springer, pp. 211–235.

Pastor, O., Fons, J., Pelechano, V. & Abrahão, S. (2006), Conceptual modelling of web applications: the oows approach, *in* 'Web Engineering', Springer, pp. 277–302.

Paternò, F., Mancini, C. & Meniconi, S. (1997), Concurtasktrees: A diagrammatic notation for specifying task models, *in* 'Human-Computer Interaction INTERACT97', Springer, pp. 362–369.

Rossi, G., Schwabe, D. & Garrido, A. (1997), Design reuse in hypermedia applications development, *in* 'Proceedings of the eighth ACM conference on Hypertext', ACM, pp. 57–66.

Schwabe, D. & Rossi, G. (1995), 'The object-oriented hypermedia design model', *Communications of the ACM* **38**(8), 45–46.

Scott, B. & Neil, T. (2009), *Designing web interfaces: Principles and patterns for rich interactions*, " O'Reilly Media, Inc.".

Souer, J., Van De Weerd, I., Versendaal, J. & Brinkkemper, S. (2007), 'Situational requirements engineering for the development of content management system-based web applications', *International Journal of Web Engineering and Technology* **3**(4), 420–440.

Van Duyne, D. K., Landay, J. A. & Hong, J. I. (2007), *The design of sites: Patterns for creating winning web sites*, Prentice Hall Professional.

van Welie, M. (2014), *Patterns*, accessed 29-July-2014, http://goo.gl/Grv0Kh.

van Welie, M. & Van der Veer, G. C. (2003), Pattern languages in interaction design: Structure and organization, *in* 'Proceedings of interact', Vol. 3, pp. 1–5.

Yahoo (2014), *Yahoo! Design Pattern Library*, accessed 29-July-2014, http://goo.gl/dSpG0v.

Zaid, L. A. (2013), The feature assembly approach for modeling & knowledge management of software variability, PhD thesis, Vrije Universiteit Brussel.

Zaid, L. A., Kleinermann, F. & De Troyer, O. (2010), Feature assembly: a new feature modeling technique, *in* 'Conceptual Modeling–ER 2010', Springer, pp. 233–246.