Improvement of a Web Engineering Method through Usability Patterns[1]

José Ignacio Panach, Francisco Valverde, Óscar Pastor

Department of Information Systems and Computation
Technical University of Valencia
Camino de Vera s/n, 46022 Valencia, Spain.
{jpanach, fvalverde, opastor }@dsic.upv.es
Phone: +34 96 387 7000, Fax: +34 96 3877359

Abstract. Usability is a feature of software quality that has traditional significance in the Human Computer Interaction (HCI) community. Recent works that have been proposed by the Software Engineering (SE) community are intended to improve the usability of software applications. This paper combines aspects that are defined in both these communities to produce usable web applications. To achieve this goal, a well-known strategy to improve usability is used: usability patterns. However, many usability patterns and guidelines could only be applied when the final system is implemented. In this work, STATUS patterns have been chosen because they solve usability issues at conceptual level. The main purpose of this paper is to improve the usability of Web Applications automatically generated by OOWS (a model-based web engineering method) applying the STATUS patterns.

Keywords: Web engineering, Web usability, MDA, model-driven engineering, automatic code generation, usability patterns, Presentation Model.

1 Introduction

Usability has become increasingly important in web engineering methods, even more important than in conventional desktop applications [6]. Some works have proposed methods for measuring usability, like Olsina's work [11]. Moreover, recent works incorporate usability as part as an MDA [9] development process [1].

Following this last emergent research line, this paper is focused on how to deal with the required usability aspects of web applications that are generated automatically in a model-driven web development process. Specifically, the objective is to improve usability in OOWS [5] (Object Oriented Web Solutions) web engineering method. OOWS has an automatic code generation process based on the MDA paradigm that produces a web application from its corresponding web conceptual schema.

OOWS is complemented by OlivaNova [3], the industrial tool that implements the methodology called OO-Method [12]. OOWS generates the code corresponding to the

[1] This work has been developed with the support of MEC under the project DESTINO TIN2004-03534 and cofinanced by FEDER

specific, web oriented user interface, preserving the business logic layer and the persistence layer generated by OlivaNova.

Currently, there are several web engineering methods that model the web interaction in an abstract way and distinguish between navigation and interface like OOWS does. Some of these methods are WebML [4], OOHDM [14] or OOH [2]. The interaction aspects related to usability are considered in all these web engineering methods by means of a specific model or a quality framework. However their proposed approaches to usability are coupled with the particular method. Therefore, applying the same concepts to another web engineering method is a difficult task.

To solve this problem the use of patterns is proposed in this work. A usability pattern suggests an abstract solution to a usability problem without taking into account platform constraints.

Several authors have written about usability patterns, like Welie [17], who makes an explicit distinction between the user's perspective and the designer's perspective. Tidwell [16] defined other patterns that are very similar to the patterns proposed by Welie. She proposes using patterns to help the design of the Conceptual Model behind the interface. Moreover, some authors, such as Kimberly Perzel [13], have been working to define usability patterns for applications oriented to the World Wide Web. However, none of the mentioned above approaches implements a true Model Compiler, meaning that the specification of the usability aspects is done at the modeling step, and is properly converted into the required software components through the corresponding transformation process.

Therefore, the main contribution of this paper is to include the usability of web applications generated with OOWS as an essential aspect to be considered. This fact is motivated by the experiences provided by users of Web applications generated by OOWS. With the purpose of solving OOWS usability problems, we have selected a set of usability patterns defined in a European project called STATUS (SofTware Arquitectures That support USability) [15]. For this purpose, the current OOWS Presentation Model (the part of the Conceptual Model that models the interaction between the user and the system) is extended. This usability improvement can be divided into two steps: 1) to select the STATUS patterns that solve the OOWS usability problems, and 2) to enrich the OOWS Presentation Model with the required expressiveness to model the functionality of the patterns that are not currently supported.

To accomplish these goals, the paper is structured as follows. Section 2 presents the usability problems in web applications generated by OOWS and which STATUS pattern provides a solution. Section 3 shows an extension of the OOWS Conceptual Model to incorporate the functionality of the usability patterns that OOWS does not currently support. Finally, section 4 presents the conclusions.

2 Analysis of OOWS Usability

STATUS patterns are defined in Juristo et al [8] as a set of generic solutions to solve common usability issues. The solution for each problem is described by means of several UML Diagrams (Class Diagram and Sequence Diagram) that can be applied in a specific methodology. The use of these patterns provides two main advantages:

- The usability mechanism is described in an abstract way from an object-oriented perspective. As a consequence, applying a STATUS pattern to OOWS is a task that can be easily performed.
- The solution proposed in the pattern is neither designed for a particular method nor a specific software platform (Web, Desktop etc.). The same principles can be applied in another Web Engineering methods

As OOWS and OO-Method are based on UML, the solution described by the STATUS pattern can be easily introduced inside the software development process. Since the set of STATUS patterns is very extensive, this paper is only focused on two STATUS patterns that, due to their functionality, proposes a solution to the usability problems that have been detected. The usability problems are related to data entry mistakes when users try to perform an operation in an OOWS Web Application. In addition, in order to make the most appropriate choice, the usability recommendations on data entry stated in [10] have been followed.

2.1 User Input Errors Prevention: Structured Text Entry

The main objective of this pattern is to anticipate possible mistakes caused by invalid user actions. To do this, the pattern proposes the use of different input mechanisms and default values. Several widgets can perform the same task, even though their visual representations are different. The goal of the usability analyst is to choose the correct widget for a concrete input data. Currently, OOWS does not allow the user or the analyst to choose the widget type, for example “list boxes” for a concrete set of values or “edit masks” to insert data in a specific format.

Another way to avoid user errors is to provide default values to the user that can be changed when they are not appropriate. OOWS supports this functionality by means of the OO-Method Structural Model. However, frequently, the list of possible values in a widget may depend on the values inserted in other widgets, thereby creating dependency relationships between widgets. OOWS does not have any primitives to model this behaviour.

2.2 Wizard: Step by Step

This pattern helps users to execute a complex action that requires several steps. Using this pattern**,** the analyst can define a wizard that will help users with complex actions that require them to introduce information in several steps. The fact of splitting the operation into different steps improves the user guidance. This functionality cannot be modelled in OOWS yet.

3 Improving the OOWS Presentation Model: A Case of Study

This section explains how the OOWS Presentation Model can be improved with usability patterns whose functionality is not currently supported. The solution proposed is to use UML stereotyped elements in order to abstract STATUS patterns functional-

ity. These new UML elements are introduced inside the OOWS Presentation Model extending the current conceptual primitives. Since many web engineering methods are based on UML, this approach can be used to define new usability concepts into their models. A prototype of model compiler that includes the functionality of *Step by Step* and *Structured Text Entry* patterns is used to generate the code of the case study.

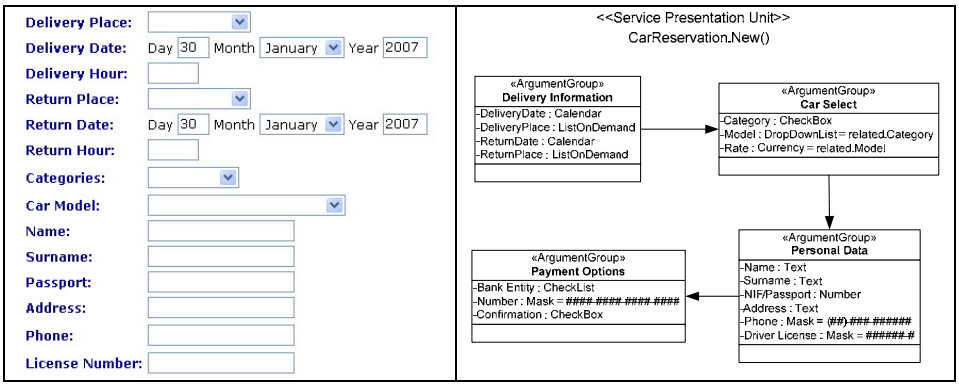

Fig. 1. a)Service Reservation Interface before applying usability patterns; b) SPU for car reservation service

To test the advantages of the proposed usability patterns, we have developed a case study based on an on-line car rental service. This paper focuses on the **reservation** service to simplify. A part of the current automatically generated web interface is shown in Figure 1a.

To support both patterns, the *Service Presentation Unit* (SPU) primitive has been introduced into the OOWS Presentation Model. The purpose of the SPU is to model how the interface that executes a service (usually a web form) will be shown to the user. Each SPU is related to a service defined in the Structural Model and is composed of the set of correspondent arguments. Figure 1b shows the SPU for the case study presented here. In the following subsections, we detail the conceptual primitives that the SPU has to model our service interface.

3.1 Supporting the "Step by Step" pattern

This paper defines an *Argument Group* represented as a stereotyped UML Class. Thanks to this primitive, it is possible to model how the service arguments are grouped in a SPU. The sequence, in which each group of arguments should be introduced, is represented by means of arrows. Moreover, each *Argument Group* can include a text description to inform the user. Briefly, an *Argument Group* represents a Wizard step (Figure 1b).

At the implementation level, the result is a set of web pages (a Wizard) to collect the argument values. In the proposed case study four *Argument Groups* are defined: Delivery information; Car Selection; Personal Data; Payment Options.

3.2 Supporting the "Structured Text Entry" Pattern

With the current OOWS Presentation Model, it is not possible to delimit the correct set of values for an argument or its input mechanism. The compiler takes into account the data type of an argument to render an appropriate widget. To solve this problem, three mechanisms have been added to the *Service Presentation Unit*:

- **Argument Widget:** It specifies the widget type that will receive the value. This primitive is defined in the attribute type for a particular argument. If no widget type has been defined, the default widget (Text) is used to input any kind of alpha-numerical string. In our case study (Figure 1b), the delivery and return dates are typed as Calendar, the car categories are rendered as checkboxes and phone and credit cards are masks.

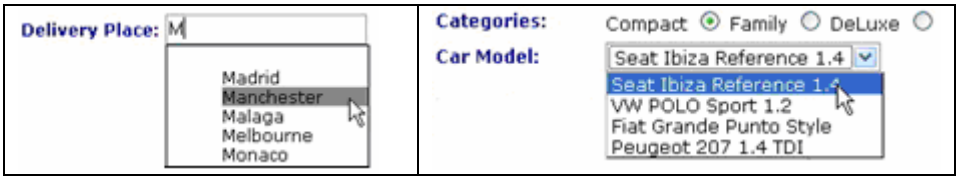

Fig. 2. a) List On Demand input Widget; b) Argument Car Model related to Category

- **List on Demand:** A *List On Demand* is a type of dropdown list with an input text whose values are retrieved dynamically. When the user writes a string, a list of instances which matches the text written will be shown. This primitive is used in the arguments "Delivery Place" and "Return Place" show in Figure 1b. An example of the interface generated from this primitive is shown in Figure 2a.
- **Related Argument**: It is useful to restrict the values that the user can insert in a widget depending on the values that the user has previously inserted in other widgets. For example, if the user has selected the desired car category, only the car models that are related to that category must be shown. Figure 1b shows this primitive in the arguments "model" and "rate". Figure 2b shows the final interface.

4 Conclusions

This research work presents a usability improvement of web applications built with an automatic code generation process. The proposed solution has the following steps:

1. **Choice of STATUS usability patterns**. STATUS usability patterns have been selected because these patterns can be incorporated in the architecture of the system throughout the entire software development process. As a consequence, this is a suitable approach to improve usability in another web engineering methods.
2. **Selection of a subset of STATUS patterns**. Of all the STATUS patterns, this paper is centered only on the patterns whose functionality has been considered to be more appropriate for web usability according to [10] and users' experiences.
3. **OOWS Conceptual Model Extension.** Two STATUS Patterns, *Step by Step* and *Structured Text Entry*, are introduced into OOWS as UML elements.

A prototype version of the new Model Compiler has been developed in order to introduce the changes in the OOWS Presentation Model. As an example of an application of this approach, this paper presents a little case study that includes the usability patterns presented. Users feedback verifies that the new web interface is more usable than previous one because it has been generated using the new conceptual primitives.

As future research, the rest of the STATUS patterns that are not currently supported by OOWS must also be considered. In addition, the Conceptual Model should include a set of metrics to measure the usability before generating the system. Finally, an empirical evaluation of usability, with industrial web applications generated by OOWS, will be carried out to validate the usability improvement of the generated code.

References

[1] Abrahao, S., Insfrán, E. (2006). Early Usability Evaluation in Model-Driven Architecture Environments. 6th IEEE International Conference on Quality Software (QSIC 2006), Beijing, China.

[2] Cachero, C., Genero, M., Calero, C., Meliá, S. (2006). Quality-driven Automatic Transformation of Object-Oriented Navigational Models. Second International Workshop on Quality of Information Systems (QoIS'06). ER2006 Workshops., Tucston, Arizona.

[3] Care Technologies: http://www.care-t.com Last visited: Feb-2007.

[4] Ceri, S., Fraternali, P., Bongio, A. (2000). Web Modeling Language (WebML): a modeling language for designing Web sites. WWW9 Conference, Amsterdam,.pp. 137 - 157.

[5] Fons J., P. V., Albert M., and Pastor O. (2003). Development of Web Applications from Web Enhanced Conceptual Schemas. ER 2003, LNCS. Springer.pp. 232-245.

[6] Hitz, M., Leitner, G., Melcher, R. (2006). Usability of Web Applications. Web Engineering, Wiley.

[7] ISO/IEC 9126-1 (2001), Software engineering - Product quality - 1: Quality model.

[8] Juristo, N., López, M., Moreno, A., Sánchez, I. (2003). Improving software usability through architectural patterns. International Conference on Software Engineering. Workshop "Bridging the Gaps Between Software Engineering and Human-Computer Interaction". Portland, USA.

[9] MDA: http://www.omg.org/mda Last visited: Feb-2007.

[10] Nielsen, J. (2006). Prioritizing Web Usability, New Riders Press; 1 edition.

[11] Olsina, L., Rossi, G. (2002). A Quantitative Method for Quality Evaluation of Web Sites and Applications. IEEE Multimedia Magazine.pp. 20-29

[12] Pastor, O., Gómez, J., Insfrán, E. Pelechano, V. (2001) The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming. Information Systems, 26(7) 507–534.

[13] Perzel, K., Kane, D. (1999). Usability Patterns for Applications on the World Wide Web. PloP'99 Conference.

[14] Schwabe D.,R.G., and Barbosa. S. (1996). Systematic Hypermedia Design with OOHDM. In ACM Conference on Hypertext. In ACM Conference on Hypertext, Washington, USA.

[15] STATUS Project: http://is.ls.fi.upm.es/status. Las visit: Feb-2007.

[16] Tidwell, J. (2005). Designing Interfaces, O'Reilly Media.

[17] Wellie, M., Traetteberg, H. (2000). Interaction Patterns in User Interfaces. PLoP 2000, Allerton Park Monticello, Illinois, USA.