WOLD: A Mixed-Initiative Wizard for Producing Multi-Platform User Interfaces

Julien Stocq[1,2]
SMAP
Avenue de l'Astronomie, 19.
1210 Brussels, Belgium
+32-2-227.98.11
julien.stocq@smap-banque.be

Jean Vanderdonckt
Université catholique de Louvain
Place des Doyens, 1
1348 Louvain-la-Neuve, Belgium
+32-10-47.85.25
vanderdonckt@isys.ucl.ac.be

ABSTRACT
WOLD (Wizard fOr Leveraging the Development of multi-platform user interfaces) helps designers to produce running user interfaces to data bases of information systems simultaneously for multiple computing platforms. This software consists of a wizard application guiding designers step by step according to a mixed-initiative approach of production rules structured in a decision tree for choosing appropriate design options covering user interfaces to be produced. The main goal of this software is to speed up the development life cycle according to a transformational approach, spiral life cycle, derivation of user interfaces from data bases structure and queries, intelligent layout derived from data base structure and query. User interfaces are structured and described according to characteristics that remain independent of computing platforms.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *user interfaces.* H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – g*raphical user interfaces, input devices and strategies, interaction styles, user-centered design*. I.2.3 [**Artificial Intelligence**]: Decision tree, production rules.

General Terms
Algorithms, Design, Human Factors, Languages.

Keywords
Data base, decision tree, design option, layout algorithm, mixed-initiative, multi-platform user interface, rapid application development, transformational approach, wizard application.

1. INTRODUCTION
Researchers and practitioners seeking to devise intelligent user interfaces (UIs) begin to realize the importance of UI design as a process to be effectively supported. Consequently, there has been an increasing interest in building intelligent interactive systems that can support the UI design process in some way, for instance by asking design questions and respond intelligently to these questions in order to produce usable UIs.

The UI design process in general is known to be a hard task to support when no constraints are imposed. However, for specific families of UIs of particular kinds of interactive systems, when the domain of interest is well defined in scope, several UI design support tools have been developed that show potential results [5, 6, 7, 8]. In this paper, we are interested by UIs of information systems which are characterized by consistent UIs to database access. This family of UIs is usually equipped with simple methods such as those provided by the CRUD pattern (Create, Read, Update, Delete), and initialize, terminate, re-initialize. Equally important, specific queries relevant to the task to be carried out by the user need to be incorporated, possibly along with more complex methods such as calculation, verification of constraints, syntactical or semantic, derivation of new values for name a few. While simple methods are usually well supported in UI design tools, complex methods need to be added manually and developed further and. These types of UIs are assumed to be somewhat regular both in presentation and dialog. Consequently, they can be considered as a good candidate for UI design support by a tool. Indeed, design decisions related to these types remain well-scoped with a limited set of values. Design decisions can also be structured easily in a sequence.

UI design tools have been criticized for not being able to clearly visualize the results of the UI production before actually launching the process. It is considered that once the results are generated, it is too late to intervene and a new cycle of design-generation is initiated. Therefore, any UI design tool should be able to provide the designer with a *generation preview* that enables her to estimate the future UI before generation.

The above requirements lead to consider a design support tool shaped as a *wizard*. *Wizard*s [1,3] are typical tools useful when the process is well structured, with limited questions/answers, one question at a time, and where navigation preserves the designer's workload. At each step of a wizard, it is possible to graphically present the designer with a visualization of the current UI based on the actual values of options. Software engineers typically use domain models for creating UIs: entity-relationship model [7], class diagram, object-oriented models [8]. Although the use of domain models is not questionable, it is believed that more UI design patterns attached to configurations of this model are expected, especially when the domain model should serve for different computing platforms with some generalization [5].

IUI'04, January 13–16, 2004, Madeira, Funchal, Portugal.

This paper will present WOLD (*WIZARD* fOr Leveraging the Development of multi-platform user interfaces), a software that helps designers to produce running UIs coupled to data bases of information systems simultaneously for multiple computing platforms while addressing the above requirements identified in prior experiences.

2. RELATED WORK

Work related to this question includes efforts made on different sides: user interfaces to data base systems, automated generation of user interfaces, multiplatform UI, and *wizard* application.

In the domain of *data base systems*, Pizano et al. [7] have developed a system that automatically generate Graphical User Interfaces (GUIs) to access a data base. While this system is systematic, automated, it does not provide any control to the designer and the potential UI to be obtained are limited as the set of design decisions remains small.

In the domain of *automated generation of UIs*, a lot of work has already been done, as reported by Paternò [5]. SIERRA [16] provides a mixed-initiative approach [4] for selecting the widgets of a GUI and laying them out according to different layout algorithms. While the process is mainly guided by the system, the designer can stop the process, change parameters and re-explore step by step. Preliminary observations revealed that the mixed-initiative was appreciated but was located at a too low level.

In the *domain of multiplatform UIs*, XWEB [6] produces UIs for several devices starting from a multi-modal description of the abstract UI. This system operates on specific XWEB servers and browsers tuned to the interactive capacities of particular platforms, which communicate thanks to an appropriate XTP protocol. TERESA [5] produces different UIs for multiple computing platform from a general task model which is progressively refined for the different platforms. Then, various presentation and dialogues techniques are used to map the general specifications into XHTML code for each platform.

In the *domain of wizards*, Dryer [3] conducted an empirical analysis showing how *wizard*s are used as intelligent agents to decompose a complex task into a sequence of simple sub-tasks, each sub-task being presented with a design option at a time. It was observed that a *wizard* is really helpful in both task structuring and reducing the work load of designers. Burton *et al.* observed that participants preferred a table of contents as secondary navigation in the wizard, while relying on the Back-Next push buttons [1].

3. THE WIZARD APPLICATION

3.1 Motivations and hypotheses

Two main elements have led us to develop this *wizard*. Firstly the increasing demand for small information systems. Since today computer gets into every organization, information systems are no longer reserved to big corporations. People have now growing capacity to run more powerful applications and store any kind of information on database. Secondly the emergence of embedded devices that offer now sufficient capabilities to support hand-held information systems.

Regarding that context, we have designed this tool with a twofold objective in mind. First the reduction of the work needed to develop an information system's interface. Second the achievement of more consistency amongst applications from different platforms. Since people that will use these small information's systems have usually little or no knowledge in computer science, they may experience some difficulties in expressing their real needs, until they see a potential prototype. According with these principles, the *wizard* will be used to generate a first running UI that will be presented to the end user. Thanks to this prototype, she will then be ready to express comments and suggest some modifications. Those will then be simply integrated trough incremental changes.

3.2 The UI generated

Before implementing any tool, we had first to specify the kind of UI we wanted to automatically generate. While the process followed to put up the principles of the generated UI is beyond the scope of this article, we would like to present its main characteristics and introduce the constraints the *wizard* has to deal with.

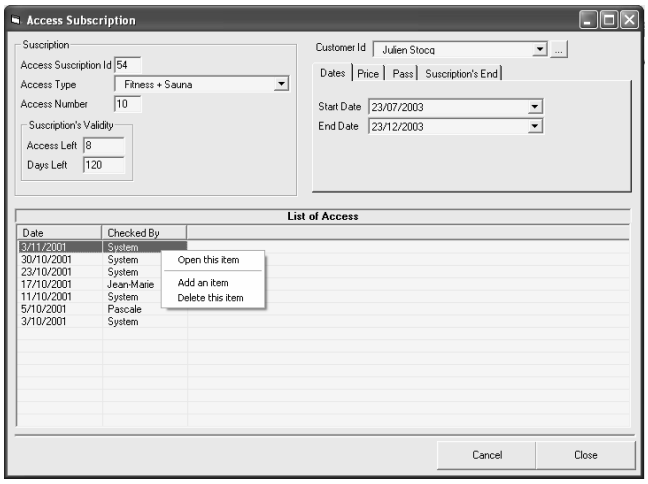

Figure 1. Main characteristics of the UI to be produced.

3.2.1 Target platforms

Since our goal is to generate a running UI, the strategy we have adopted is to define a set of specific rules for each platform we work with. So, rather than using general models, the *wizard* uses a specific one. Right now, WOLD generates UI code for the Microsoft Visual Basic (Fig. 1) and the eMbedded Visual Basic platforms. If it is true that this approach compels us to define a different set of rules for each platform, we believe that it is the best solution when it comes to generate effective running UIs. Indeed, we can see that other tools based on more general models build rather software's skeleton than effective running applications and therefore require often time consuming "post generation" customizations.

3.2.2 Architecture

As we have seen, the purpose of the generated UI is to allow the end user to easily communicate with a database. The architecture of the form has been designed in that sense. Furthermore, we known that the more simple is the code, more fast will be the customization of the UI. So, we have also paid a special attention to his simplicity and his separability. In that sense, the architecture takes into account the both points of view of the end user and the developer.

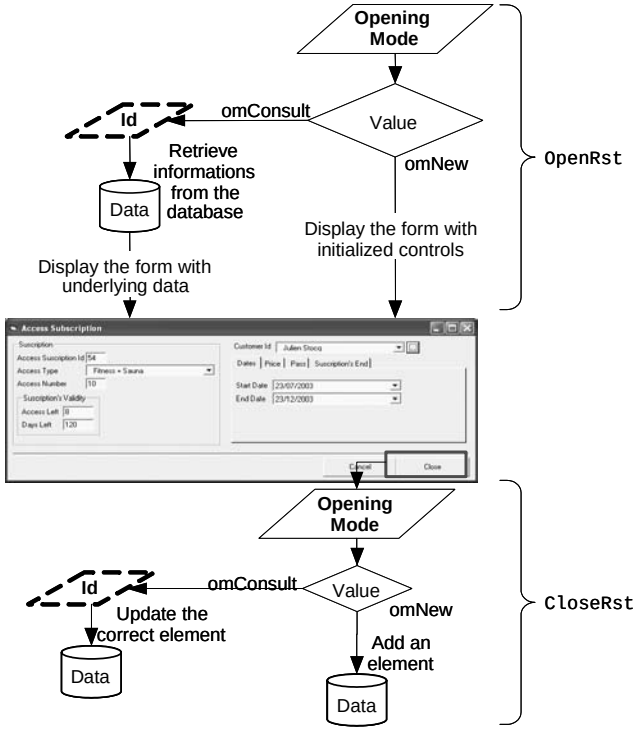

Figure 2. The production process.

Figure 2 graphically depicts the production process. Each Visual Basic form has two main procedures: the `OpenRst` that retrieves data from the database and the `CloseRst` that updates it. When writing the code to call a form, the developer must also specify the `OpeningMode` and if necessary the `Id` of the element on which the database has to be positioned. Except for these two parameters, all other mechanisms involved during the form's interaction with the database have local scopes that guarantee the separability of the UI with respect to the rest of the application. In other words, each form deals with its own processes and variables. This architecture provides an easy way for the developer to display a form simply by calling the following procedure:

```
Frm.Display(Id,OpeningMode).
```

3.2.3 Two kinds of Constraints

When generating forms the *wizard* is subject to two kinds of constraints. First it has to produce a high quality and efficient UI. In order to achieve this it has the capability to manipulate complex widget such as `MultiPage` and `Frame`. In addition the developer can work with the full range of controls available in Visual Basic (`ComboBox`, `CheckBox`, `OptionButton`,…). We have also provided the *wizard* with a number of rules so it has the opportunity to built superior UI.

For example, widgets will be automatically resized according to the size of data they will have to display. An example of moderately complex UI generated is reproduced in Figure 1. Second, since one of the purposes of our tool is to speed up the development cycle, the *wizard* codes everything in hard. That means that it does not incorporate components (or properties such as `ControlSource`) that force the developer to loose control over some processes. This enables the developer to customize generated forms in a very quick and simple way. The wizard consists of seven steps where step *n* takes into account inputs that were given at step *n*-1: database selection, data source selection, building the `OpenRst` procedure, data source selection for combination and list boxes, building the `CloseRst` procedure, setting the size of the widgets, and laying the widgets out in their containers. These steps are supported by a mixed-initiative approach [4] where some values of parameters are suggested by WOLD when possible. These suggested values can then be superseded by values provided by the designer when she want to keep the control over the generation process.

4. ACKNOWLEDGMENTS

The authors would like thank Jean-François Hemmerlin and Lionel Neyts for the collaboration in the development of the WOLD project. This work was supported by Salamandre research project (http://www.isys.ucl.ac.be/research/salamandre.html) under convention n°001/4511 of "Initiatives II" research program, Walloon Region (Belgium). We gratefully acknowledge the support of the CAMELEON research project (http://giove.cnuce.cnr.it/cameleon.html), the CAMELEON partners for fruitful exchanges, for useful discussions, and the SIMILAR network of excellence (http://www.similar.cc), the European re-search task force creating human-machine interfaces similar to human-human communication of the European Sixth Framework Programme.

5. REFERENCES

[1] Burton, M., Pupons-Wickham, D., Phelps, L., Spain, K., Crews, J., Rich, N. Secondary Navigation in Software Wizards, in Proc. of CHI'99 (Pittsburgh, May 15-20, 1999), ACM Press, 294-295.

[2] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, 3 (June 2003), 289-308.

[3] Dryer, D.C. Wizards, Guides, and Beyond: Rational and Empirical Methods for Selecting Optimal Intelligent User Interface Agents, in Proc. of IUI'97 (Orlanda, January 6-9, 1997), ACM Press, 265-268.

[4] Horvitz, E., Principles of mixed-initiative user interfaces, in Proc. of CHI'99 (Pittsburgh, May 15-20, 1999), ACM Press, 159-166.

[5] Mori, G., Paternò, F., Santoro, C. Tool Support for Designing Nomadic Applications, in Proc. of IUI'2003 (Miami, January 12-15, 2003), ACM Press, 141-148.

[6] Olsen, D.R., Jefferies, S., Nielsen, T., Moyes, W., Fredrickson, P. Cross Modal Interaction using XWEB, in Proc. of UIST'2000 (San Diego, November 5-8, 2000), ACM Press, 191-200.

[7] Pizano, A., Shirota, Y., Iizawa, A. Automatic Generation of Graphical User Interfaces for Interactive Databases Applications, in Proc. of CIKM'93 (Washington, November 1993), ACM Press, 344-355.

[8] Vanderdonckt, J., Berquin, P. Towards a very large model-based approach for user interface Development, in Proc. of UIDIS'99 (Edinburgh, September 5-6, 1999), IEEE Computer Society Press, 76-85.