IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4860

From Design to Development: A W2000-Based Framework; Issues and Guidelines

Roberto Paiano, Andrea Pandurino
Dipartimento Ingegneria dell'Innovazione
Università di Lecce
Via per Arnesano, 73100 Lecce, Italy
Tel: +39 0832 320229
Fax: : +39 0832 320279
{roberto.paiano, andrea.pandurino}@unile.it

ABSTRACT

The increasing complexity of Web applications underlines the importance of achieving suitable organic design before starting development. The IT community has produced powerful methodologies to help the application designer to analyze and manage the complexity of the task. On the other hand, developer tools from many software houses are flooding the market, promising fast and easy development. But the IT community's approach and that of the software houses, while both are oriented to the question of web application development, are too far apart and have no point of contact; we shall try to fill this gap. In this paper we introduce a new framework idea,, based on W2000 methodology [1][2], which allows fast prototyping and the development of web applications using standard technologies (Microsoft .Net and Sun JavaOne) [3].

This work is divided into two steps: the first describes the development of a pilot application and the second builds on this, setting out the guidelines of the framework.

1. INTRODUCTION

The production of hypermedia applications in general, and of web applications in particular, is a very complex process, but often receives insufficient attention. The Market, led by the necessity to produce applications in a short time, reduces and sometimes eliminates the design phase, discarding proven Software Engineering design methodologies, which it considers technically unsuitable for the new media and too expensive in resources and time. Seen from this viewpoint, there are many commercial products, technology-oriented and often designed for a specific kind of application such as hypermedia catalogues, able to help the developer produce what the Market demands; the particular features of these products have led to the design phase being neglected, or even discarded altogether.

As long as applications remained small in size, serious problems rarely arose. However, as they grew in dimensions and complexity, the applications started to have features similar to the those of traditional software, combining data-intensive and transactional issues with purely navigational ones.

The consequences of such a situation are becoming more evident, and are forcing the whole IT community to reflect. It is plain therefore that new methodologies are needed; these may arise from traditional ones, but the new requirements have to be taken into account.

The HOC (Politecnico di Milano) and the SET-Lab (University of Lecce) have been involved in the development of W2000, which tries to reach the above-stated goal by allowing accurate conceptual modeling of the applications before the implementation phase. On the basis of W2000, we are now seeking to build a complete development framework which is able to join more closely the stages of methodology and to implementation, so that precious design effort can quickly lead to real applications, as automatically as possible.

1.1 W2000 in a nutshell

To better understand the topics referred to in this study, it is helpful to know something of W2000 methodology fundamentals. It assumes that a clear distinction between the different aspects of the application which need to be observed during its design is essential, in order to make the design itself a structured and easily controllable process and to obtain clear modeling, suitable for different users and delivery devices.

After the indispensable Requirement Analysis phase, carried out according to a goal-oriented approach, the methodology suggests a sequence of steps which may be summarized as follows:

- Information Design: the goal here is to describe the information that the application is going to deal with, giving it a structured organization. An important feature of this phase is that during the construction of the information structure the user's point of view [4] is held as fundamental.
- Navigation Design: this clarifies the most important aspect of hypermedia applications, reconsidering the information and its organization specifically from the viewpoint of its fruition and defining the navigational paths the user can follow.
- Publishing Design: the results of the previous steps must be completed with presentational considerations and organized into "pages" and "publishing units".
- Operations Design: this is the step in which all functional and transactional features (such as "register", "submit", etc.) not strictly connected to the hypermedia paradigm are modeled. Here the model allows the user to invoke the "functionalities" of the application.

Moreover, operations are the "building blocks" to be used to support complex transactions: in combining them the designer must pay attention to the consistency of the application state and also to the behavior of the users. The "customization methodology" [5] [6] comes together with hypermedia design to manage the different user profiles and the various devices in many contexts.

The current state of our methodology, as described above, allows the designer to obtain a very advanced conceptual model of the applications before their development. The next phase of the research will try to make the abstractions concrete and to bring the methodology stage closer to those of development and implementation.

2. THE FIRST STEP

There is currently a gap between the analysis (with the described steps) on the one hand, and the software architecture and implementation on the other.

However, before we could tackle the construction of a complete development framework around the methodology, we had to properly understand the size and nature of the gap. Thus we decided to produce a

pilot application, whose information, navigation and functional aspects were as balanced and general as possible.

There were many applications designed with the W2000 methodology, but we chose a bank web application for managing debit cards, for two reasons:

1. we had direct contact with the bank; so the application and its design would be based on true requirements in the real world.
2. the features of the application were suitable to fully verify the completeness of the methodological support, but also to give a good idea of the problems specific to implementation. The application is designed for different delivery devices and addressed to various kinds of users.

Our attention, therefore, focused on the following elements:

- to verify which logical steps had to be taken to achieve practical implementation from the conceptual model.
- to verify the support that the model gave, first to the designer of the software architecture and then to the developer.

In accordance with the goals of validating the methodology, we chose to go directly to the development of the specific application, using existing market standards: n-tier architecture, with immediate recognition of presentation, business and data levels, each with its own features and functionalities.

W2000, focusing on the design at a conceptual level, gives complete freedom to choose the particular implementation technology. We decided to use the Microsoft .NET development framework. This choice does not affect the generalisability of the results we obtained and is perfectly in accordance (as we will explain later) with the object-oriented nature of the W2000 model.

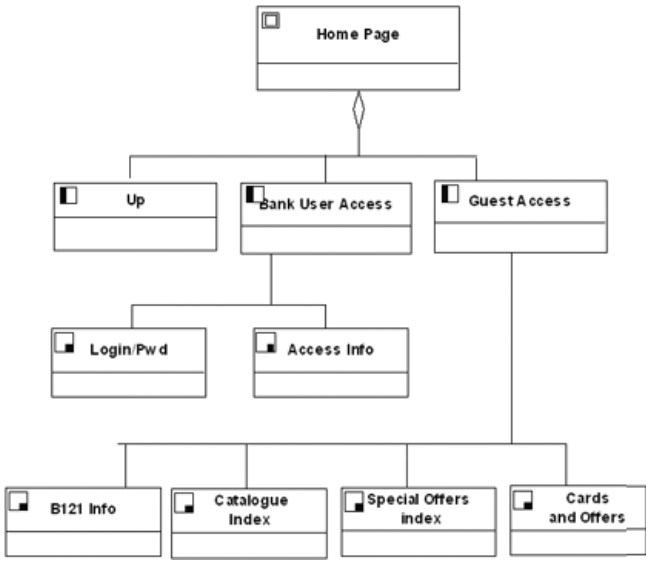
2.1 The chosen architecture

On the basis of the choices we made, the "Debit Card" application was developed using a *three-tier* architecture:

- **Data access**: Access to the application data. We decided to describe the data of the pilot application in a classic Entity-Relationship diagram. According to the model, the user-experience approach of W2000 describes the application content in several Information Design views, one for each user of the application. So to obtain a single database we unified the different users' views and removed all the redundant links; after this operation the resulting diagrams are more easily translatable into an E-R.
- **Business**: holding all the business logic behind the application; it is concerned with the support of multi-user access, making clearer user roles and the relative behaviors of the application. Thanks to the object-oriented approach of the chosen development framework (Microsoft .Net) and to the property of polymorphism [7] of the objects, it is possible to implement and obtain, with minimal effort, different behaviors, depending on the role of the user requesting a service.
- **Presentation**: on this level, using the data supplied by the data access level via the business logic level, the presentation pages for the Web, PDA, WAP clients, etc. are composed. In accordance with the model resulting from the Publishing Design and with the kind of implementation chosen, it is possible to make an accurate mapping of the model's structures onto those of the .NET framework. The Publishing Model allows the development of pages starting from page templates. The pages are then modeled as groups of Sections, which are modeled in their turn as aggregations of Publishing Units, in a hierarchical structure, which describes the page suitably. In a similar way, .NET, using ASP.NET, allows the creation of UserControl classes with rendering interfaces (HTML, WML, etc.), with which a presentation page may be composed.. Pages are simply containers for classes, which correspond to the Publishing Units of the model. In order to make the concepts clearer, we can take as an example the structure of the home page for the pilot application, as shown in Figure 1.

In figure 2, it should be noted that the various UserControls inside the page are instances of the classes below; this is made plain using

Figure 1. Publishing Model of the Homepage for the pilot application



appropriate names: CatalogueIndex1.ascx, for example, is an instance of the class (UserControl) CatalogueIndex. These features of the .NET framework accord perfectly with the goals of the W2000 model, giving us good feedback on the choices made.

3. THE SECOND STEP

During the development of the pilot application we verified the W2000 approach and its design: we are now confident that the W2000 design allows for the development of the whole application.

The experience gained showed us how we can create a complete general-purpose framework able to automatically prototype and publish

Figure 2. Publishing Model of the Homepage for the pilot application

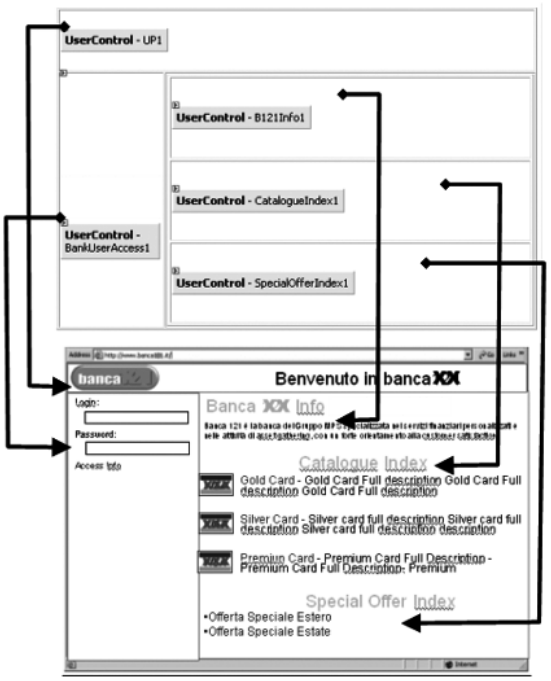
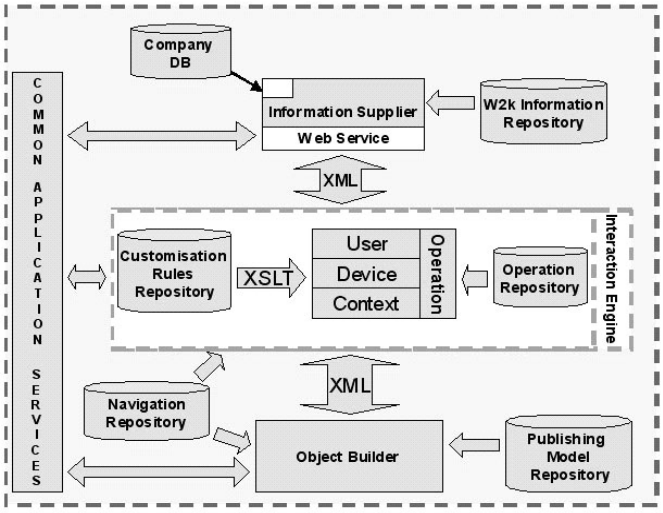


Figure 3: Design in-the-large of the W2000 framework



web applications based on the W2000 methodology: figure 3 shows the in-the-large design of our framework idea, born as a generalization of the three-layer software architecture used to develop the pilot application.

To understand the choices we made in the design of the framework, it is necessary to describe the details and the functions of each module:

- **Information Supplier**: all the information content is described in the Information Model; in the pilot application we built a *design-based* database; however, this assumes an ideal context: usually the information is stored in several ways but rarely exactly as we would wish. This module works as a middleware: it adjusts the information and provides a standard way to access it (using a set of API's) . It is plain that if necessary, this module will manage a specific database where the designed application's information was stored.
- **Interaction Engine**: a normal web application has a high customization level, depending on its multi-user and multi-device nature. So every information type has to be filtered for specific users, devices and contexts. This module corresponds to the business level (for a three-tier application) we adopted for the pilot application; in this module our idea is to encapsulate the logic into an XML structure [8]; a good technique may be to use an XSLT [9] pattern (stored in the "Customization Rules Repository"), which filters and manipulates the information (supplied by the Information Supplier module in XML format). Using this approach enables us to increase the dynamic aspect of the framework, providing the faculty of piloting the customization from a single point and in a standard way. Users not only access the information but interact through the operations. This module implements the operations, using a description of the operations (stored in a repository) and the customization rules. The importance of this piece of the framework is plain, if we consider its function, and it is so big that we may call it the "Core Engine".
- **Object Builder**: this module is less defined than the others, because the framework leaves as much freedom as possible to the layout developer. For this reason the module implements only the basic methods such as an XSLT transformation method (to obtain HTML or WML pages) or a pass-through method to manipulate the raw data directly. In accordance with the pilot experience and the W2000 model, this module has to operate with publishing units not only in its device visualization (HTML, WML) but also in its W2000 schema; so this module needs to know the publishing units' structure and how they must be joined to create the XML description of the page. This publishing information, described in XML format, is stored in a repository that will be managed by a sub-module called the "Publishing Module".
- **Common Application Services**: The framework software architecture, described in-the-large above, does not cover all the aspects of a framework; thus this module implements standard services such as logging, users' accounts, users' status, context variables. Many of these services are included in the specific implementation technology, but in order to provide a generic, free-from-technologies framework, they must be supplied.

During the development of the pilot application, the developer had complete responsibility for content, layout and software architecture. With the introduction of the framework and its intensive use of open standards to describe, exchange and manipulate the information (XML and XSL), it was possible to better subdivide the actors involved:

- **Web author:** The creation of a "W2K Information Repository" allows the editing of the content independently of its visualization layout; so the web authors focus only on the content-oriented aspect of filling the repository.
- **Publisher Developer**: the framework must lead to a real web application, so someone must create the layout; this actor can create the specific client pages by using the service supplied by the object builder (XSLT transformation) or by creating a specific service outside the framework, which allows total freedom in layout and technology choices using the raw content data .
- **Application Domain Developer**: as part of the in-the-large design of the framework we described the customization rules and the operations in a repository; this way, the dynamic aspects linked to the multi-user and multi-device nature of the application are separated from the layout and the content; this actor translates the logic described in the W2000 design into repository structures. With the introduction of automatic design tools, this actor will probably become unnecessary.
- **Framework Administrator**: just like the rest of the software, the framework also has an administrator, who supervises the correct functioning of the whole system.

The framework uses XML to describe the application content and the design structures, and uses XSLT to represent the customization rules and the dynamic aspects of the design; these *"text"* representations make it possible to describe the pages (and their behaviors) of the model in an abstract and standard form, so it is possible to reuse them for different devices; in addition the device behavior will always be consistent because it depends on the rules stored in xml form for all devices.

In the in-the-large design of the framework and in its description we explained how the main modules work and gave a draft of how these modules interact. We did not provide explanations of the XML structure for data exchange or for the repository format, because we are still working on these.

4. CONCLUSION AND FUTURE WORK

The time we spent developing a real application, starting from the W2000 model but using standard technologies, has been of great use, considering that the development phases were executed with the new approach in mind.

On the basis of the experience gained, the main framework concepts are now clearer and are able to generate the framework design in-the-large described above. In addition, we are now aware of which changes need to be made to the W2000 model, and what it requires to be complete. We have already made many choices and collected many ideas for present and future work in order to complete our framework vision.

Once we have determined the framework design, the XML data format and the repository structures, we will be able to start on the framework implementation, to be tested with other applications (which will provide feedback).

REFERENCES

[1] L. Baresi, F. Garzotto, Paolo Paolini, From Web Sites to Web Applications: New Issues for Conceptual Modeling, *Proceedings WWW Conceptual Modeling Conference*, Salt Lake City, October, 2000.

[2] L. Baresi, F. Garzotto, and P. Paolini, Extending UML for Modeling Web Applications, *Proceedings of 34th Annual Hawaii International Conference on System Sciences (HICSS-34)*. IEEE Computer Society, 2001.

[3] http://wwws.sun.com/software/sunone/

[4] V. Perrone: *User Centered Design di Applicazioni per il Web* (Lecce, University of Lecce, 2000).

[5] P. De Bra, P. Brusilovsky, G. J. Houben, Adaptive Hypermedia, From Systems to Framework, *ACM Computing Surveys*, 2000.

[6] K. Tochtermann, A. Kussmaul, D. L. Hicks, Support for Customization and Personalization On the Web, *Proceedings of WebNet 99 – World Conf. on the WWW and Internet*, Honolulu, 1999.

[7] G. Rossi, D. Schwabe, F. Lyardet. Web Application Models are more than Conceptual Models. In P.P. Chen et al. (Eds.). *Advances in Conceptual Modeling* : 239252. Springer LNCS 1727, 1999.

[8] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML), 1.0. *Recommendation, W3C*, February 1998. http://www.w3.org/TR/1998/REC-xml-19980210.

[9] James Clark. XSL Transformations (XSLT), 1.0. *Recommendation W3C*, 16 November 1999T http://www.w3.org/TR/1999/REC-xslt-19991116