Software Reuse in Hypermedia Applications

Roberto Paiano
University of Lecce, Italy

INTRODUCTION

Hypermedia applications were, at the beginning, hand-coded pages with "ad-hoc" links. This production method was acceptable until a few pages had to be produced, but it became rapidly unmanageable when several hundreds of pages with complex interactive objects had to be considered. In particular, two interwoven problems rapidly became relevant: how to ensure the "usability" of modern large hypermedia-applications (Garzotto, Matera & Paolini, 1999), and how to improve the efficiency of its production/maintenance process.

In good hypermedia applications, in fact, the reader should be able to effectively exploit the information contained in the application: that is, he or she should be able to quickly locate the objects of interest, to understand the inner structure of the objects and to easily navigate from one object to another. Several factors concur to the achievement of usability: one of the most important is to have a good structuring of the information objects and a good structuring of the navigation patterns.

BACKGROUND

Several authors have recently proposed the adoption of design models (Garzotto, Mainetti & Paolini, 1995; Isakowitz, Stohr & Balasubramanian, 1995; Schwabe & Rossi, 1995)and design patterns (Rossi, Schwabe & Lyardet, 1999), in order to improve the quality of hypermedia applications, at least for those aspects concerning structure and navigation. Other authors (Conallen, 1999; Schwabe & Rossi, 2000) have proposed the use of object oriented paradigm to model this kind of application, but the navigation structures are more simple. Design models provide, in fact, the primitives that allow structuring the information objects and the corresponding navigation patterns along regular and systematic features, improving consistency, predictability (for the user), robustness of the design, and therefore improving usability. The ancestor of these models can be traced to HDM (Garzotto, Paolini & Schwabe, 1993) and its evolution: W2000 Model (Baresi, Garzotto & Paolini, 2000).

The adoption of W2000 to design the internal structure and the navigational features of hypermedia applications is desirable for three reasons:

- resulting applications are usable;
- the production process can be decomposed into sub-problems easy to manage;
- the application model can be "executed" by a suitable "interpreter" to create the application pages in a way that is independent from the specific application.

Furthermore, in several real-life projects we encountered the problem of dealing with application families. An application family is a set of applications sharing (part of) the content and also (part of the) conceptual design. The problem of application families is the typical situation where the application owner, after a successful first application, needs a second one very similar to the first one. At first it seems a simple problem of "reuse" of content: use the same pictures, use the same texts, use the same data, and so forth (Garzotto, Mainetti & Paolini, 1996). After a while it becomes apparent that not only content, but also (pieces of the) conceptual structure must be "reused". Then comes a third "similar" application, and so on. So, the truth emerges: the designer should have started from the beginning having in mind a family of applications, knowing that several specific applications could have resulted from it. In other words the designer should have optimized the activity of "carving out," from a family, a specific application, for a specific need.

Therefore it became clear that the design process, the design model and the design support system should adopt the notion of family of applications. Such kind of activity is made easily using a structured model.

BRIEF DESCRIPTION OF W2000 METHODOLOGY

The methodology was developed by the UWA Consortium (UWA), and specifically by Polytechnic of Milan.

W2000 methodology assumes that it is essential to make a clear distinction between the different aspects of the application that need to be observed during the design phase, 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 Requirements Analysis phase, guided by a goal-oriented approach, the methodology suggests a sequence of steps that may be briefly summarized as follows:

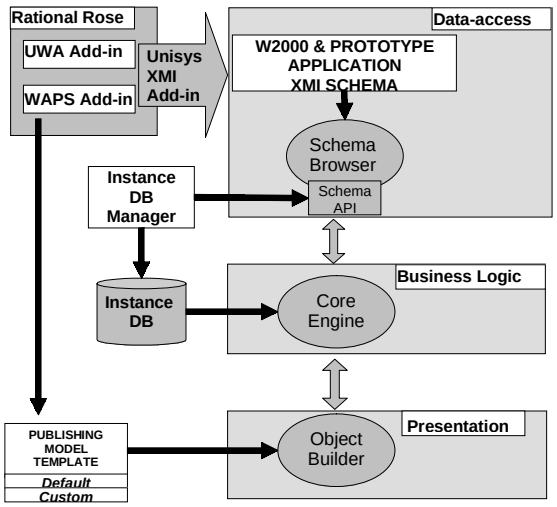
- Information Design: the goal is to describe the information that the application is going to deal with, giving it a structured organization from the user's point of view.
- Navigation Design: this reconsiders the information and its organization from the viewpoint of its fruition, defining the navigational paths the user can follow.
- Publishing Design: the results of the previous steps must be complemented with considerations on presentation and organized into "pages" and "fruition units".

According to the previous description, a database can store the application components described by the model, and then a run-time engine can extract those components from the database to display it to the reader. This kind of engine, named WAPS, is application independent, so it is really reusable and it may be defined as a W2000 methodology Interpreter. It is the last evolution of a family of navigation engines according to the evolution both of methodology (Bochicchio & Paiano, 2000; Paiano & Pandurino, 2003) and available technology (Bochicchio & Paolini, 1998; Bochicchio, Paiano & Paolini, 1999).

A REUSABLE INTERPRETER

The run-time environment, the WAPS core, has the main task of creating a mock-up application starting from the W2000 model in XMI format.

Figure 1. WAPS architecture



In accordance with the modular structure of the W2000 methodology and the various aspects of WA, as shown in Figure 1, it is possible to identify a clear n-tier architecture for the WAPS run-time environment. This choice was highly suitable for the W2000 methodology: in order to manage the complexity, each architecture layer manages a single aspect and provides services to the other levels. All the data managed in the modules are in XML format; furthermore, all interaction between modules is in the same format according to the market trend and standard.

It allows the use of transformation parsing techniques like XSL in the visualization and processing phases, also allowing the following of the evolution of methodology.

- Rational Rose Add-in: Rose helps to design the WA in graphic format using standard UML notation, in accordance with W2000 methodology, in order to obtain a "machine readable" description. Another rational rose add-in: "Unisys Rose XML Tools," produced by Unisys, exports the UML diagram into a standard XMI output.
- Schema Browser: This module allows a unique entry point to the WA schema, hiding the complexity in order to manage the XMI in raw mode. The module provides a set of schema APIs (S-API) to navigate through the WA model via W2000 primitives, bypassing the UML MOF used by XMI.
- Core Engine: This module corresponds to the business level for a three-tier application. This module has the task of understanding the requests from the Object Builder, using the S-API of the schema browser to compose the reply schema that will contain the application data taken from the Instance DB. Since this module creates the reply schema, all design customizations take effect at this stage.
- Object Builder: This module is the door to WAPS systems: the user request comes in, the prototyped page goes out. The module moves the request to the Core Engine and receives the response in XML-like format. Its main task is to apply a template to make the page visible. WAPS uses the XSLT transformation to obtain HTML or WML pages.
- Instance DB: It is an E-R database that contains the data that will be shown to the user. The E-R schema is derived from the W2000 model; thus the schema is fixed and does not change with the domain of the WA being prototyped.
- Publishing Model Template: It is an E-R database that contains the references to the visualization template to create the page.