In the hypermedia communication process, authors are connected to readers through hypermedia applications. The underlying software is an important part of the “communication channel” between authors and readers: it is effective if it is “transparent.” This means that hypermedia software must be flexible and powerful, to support naïve users as well as “experienced navigators;” it must be developed and tested before contents, because contents, in real projects, are released very late; it must be robust and error-free; and finally it must be inexpensive with respect to contents, because contents are the focus of the production effort. For all these reasons, the run-time software must be contents independent and application independent; in fact, it can be powerful enough only if its development costs can be shared over a large number of projects. Two applications, built around a reusable run-time engine based on the previous considerations, are presented in the paper.

INTRODUCTION AND BACKGROUND

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 and Matera, 1997; Garzotto, Matera and Paolini, 1999; Nielsen and Mack, 1994), 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: i.e. he should be able to quickly locate the objects of his 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. Several authors have recently proposed the adoption of design models (Fraternali and Paolini; Garzotto, Mainetti and Paolini, 1995; Isakowitz, Stohr and Balasubramanian, 1995; Mecea, Atzeni, Masi, Merialdo and Sindoni, 1998; Schwabe and Rossi, 1995), and design patterns (Gamma, Helm, Johnson and Vlissides, 1996; Garzotto, Paolini, Bolchini and Valenti, 1999; Nanard, Nanard and Kahn, 1998; Rossi, Schwabe and Lyardet, 1999) in order to improve the quality of hypermedia applications, at least for those aspects concerning structure and navigation. Design models provide, in fact, the primitives that allow to structure 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 and Schwabe, 1993).

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

- resulting applications are usable;
- the production process can be decomposed in 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.

The third point is the subject of the remaining of this chapter: in the next section, we examine the main concepts concerning the HDM interpreter we developed; in the section following, we describe two real applications built around the interpreter; in the section after that, we draw the conclusions and describe future work; the final section is for references and bibliography.

A REUSABLE HDM INTERPRETER

A short introduction to HDM is essential for the comprehension of the following of the paper.

In brief, for HDM a multimedia application consists of an Hyperbase, an Access Layer and a Visual Layout description. The Hyperbase contains the actual information objects and the basic connections among them, the Access Layer provides access paths to the objects of the Hyperbase, the Visual Layout descrip-
Venture to the Interior – Virtual Object Lessons
www.igi-global.com/chapter/venture-interior-virtual-object-lessons/41082?camid=4v1a