SOFTWARE REUSE IN HYPERMEDIA APPLICATIONS

Mario Bochicchio (bomal@ingle01.unile.it), Roberto Paiano (rpaiano@ingle01.unile.it)

SET-Lab, University of Lecce,

Via Per Monteroni, 73100, Lecce, Italy. Tel. +39 0832 320216, Fax +39 0832 320279

ABSTRACT

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.

1 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 [1-3], 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 [4-8], and design patterns[9-12] 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 [13].

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 paper: in section 2 we examine the main concepts concerning the HDM interpreter we developed; in section 3 we describe two real applications built around the interpreter; in section 4 we draw the conclusions and describe future work; section 5 is for references and bibliography.

2 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 description is made of page templates used to paginate the application.

The Hyperbase consists of Entities grouped in Entity-Types; Entities may have an inner structure.

Application Links, grouped in Link-Types, are used to connect different Entities. The definition of the Entity-Types and Link-Types is called, in HDM, "Hyperbase Schema in-the-large".

The "Hyperbase Schema in-the-small", instead, details the inner structure of the nodes, i.e. the smaller structuring units permitted for entities, and the inner structure of links. In brief "Slots" are the smallest pieces of information defined in-the-small, Frame are aggregate of slots and/or other frames [13,14].

The Access Layer of an application is organized into Collections. A Collection groups together entities and/or other Collections. Collection members are topologically organized (Sequence, Tree and Lattice are the most common). Each Collection has a "Center", used to "introduce" the collection, to "synthesize" its content, and to reach the collection members [13,14].

According to the previous description a database can store the application components described by the model, then a run time engine can extract those components from the database to display it to the reader. The overall structure of a run time engine performing that job is described in the following section.

2.1 Run Time Engine

The overall architecture of the Run-time Engine is shown in Fig.1. The Engine has been developed using Macromedia Director. Microsoft Access was used as relational database, and Datagrip, from Sight & Sound, was used to access data from Director.

The architecture is application independent, i.e. it can be readapted to different projects in few time and with a small effort.

Fig. 1 Run-Time Engine

The engine components are described in the following, referring to a typical user session:

1. When started, the Presentation-Engine (PE), performs the opening of the default starting collection (Home-Page) by sending the `OpenCollection("Home")` message to the Navigation-Engine (NE).
2. The NE initializes the navigation context and requests to the Database-Engine (DE) all data needed to build the requested "logic page". The calls between NE and DE are multiple, so in Fig.2 a dotted line is used.
3. PE selects the right visual layout from the available ones, then it merges the data retrieved from the DE with the layout description to build the "physical page" and to display it to the user.
4. PE waits for user generated events. When, for example, the user selects (by clicking on) an item, the PE performs the logical decoding of the event and sends an `OpenItem(x)` signal to the NE.
5. NE updates the navigation context by adding, the current navigation step to the navigation history, then it asks DE to retrieve the data for the requested page.
6. PE acquires the last retrieved data, then it shows the physical page by using the related layout.
7. If the user is in a collection and it asks, for example, for the NEXT collection item, PE sends a `GOTO(NEXT)` signal to the NE.

8. NE updates the navigation contest and, according to the current status, signals to the DE to retrieve the requested data, then it informs PE about the layout to be used.
9. PE presents the retrieved data to the user according to the layout description.

Fig. 2 Messapia: Hyperbase

Fig. 3 Messapia: ER schema derived from the HDM model

The main component of the engine is NE that interprets the user request (following the HDM schema) and decides the correct page to be shown.

A key concept in this architecture is represented by the separation between logic page and physic pages. This separation allows the structural designer and visual designer to work at same time on different jobs.

3 APPLICATION EXAMPLES

Using the described architecture two different applications have been developed in the last year at TeleMedia Lab. The first one, Messapia, is about an ancient population living in Salento (in the SE of Italy) before Roman conquest. The second is the presentation of the University of Lecce, with its departments, its research programs and its cultural environment.

The Entity Types of Messapia are shown in fig 2:

- Archeological Findings: describes all findings of Messapi;
- Site: describes archeological sites;
- Museum: describes the local Museum that store many archeological findings;
- Arguments: describes the Messapi language, its customs and its history using interviews and images.

The hyperbase, completed by application links and access structures, is mapped in the relational database represented in fig 3.

The Home-Page is shown in fig. 4.

Fig.4 Messapia Home Page

Fig.5 Cratere: Card way and Visual way

This application uses two different way to present the multimedia contents: a Card-Way, shown in Fig.5, based on text with related images and audio, and a Visual-Way, shown in Fig.5 based on slide-shows with audio.

The University CD has the Entity Types: Faculty, Department, Research Area, Event, Site and Argument.

Fig. 6 University presentation: Home-Page

Fig. 7 University presentation: sample page

These application contents are organized as slide-shows with synchronized audio comments. The Home-Page, with the access structures, is shown in Fig.6, a sample page is shown in Fig.7.

The application is bilingual (Italian and English).

The lifecycle of the two applications was the following: the first one was completed in 8 months by 4 people, including HDM modeling, engine implementation and development of multimedia contents. The visual design was done in about 1 month from a visual communication specialist; the second application was completed in about 4 months by 2 people. This time concerns essentially the development of multimedia contents. The structural and visual modeling was completed in 15 days.

The run time engine required only minor changes to be adapted to the second project. The database structure had few changes concerning the addition of some multimedia slots. The main changes was required from the Presentation Engine, concerning the integration of the new visual layout.

4 CONCLUSIONS AND FUTURE WORK

It is an obvious observation the fact that we need to improve the quality of hypermedia applications, reducing at the same time the costs and the time required for their development. The adoption of structured approaches and conceptual models, such as HDM, is an important step in this direction.

The construction of a reusable engine for hypermedia application, based on the HDM model, is the logical extension to the model-based approach. The reusable engine, in fact, is able to implement the needed hypermedia software in an application-independent way. The effectiveness of this approach is based on the ability of HDM to describe a wide range of complex hypermedia applications.

In the near future the engine will be extended to support more HDM primitives. The main goal is not the “fully automatic” implementation of hypermedia presentations, but an effective and sophisticated navigation support.

5 REFERENCES

- [1] F.Garzotto, M.Matera, “A Systematic Method for Hypermedia Usability Inspection”, In *The New Review of Hypermedia and Multimedia*, 3, 1997
- [2] J.Nielsen, R.L.Mack, “Usability Inspection Methods”, Wiley & Sons, New York 1994.
- [3] F.Garzotto, M.Matera, P.Paolini, “Abstract Tasks for Hypermedia Usability Evaluation”, Dept. of Electronics and Information, Polytechnic of Milan, Technical Report No.03-99, 1999.
- [4] F.Garzotto, L.Mainetti, P.Paolini, “Hypermedia Application Design: A Structured Approach”, In J.W.Schuler, N.Hannemann and N.Streitz Eds., “Designing User Interfaces for Hypermedia”, Springer Verlag, 1995.
- [5] D.Schwabe, G.Rossi, “The Object-Oriented Hypermedia Design Model”, Communications of the ACM, Vol. 38, N. 8, pp.45-46, Aug. 1995
- [6] T.Isakowitz, E.A.Stohr, P.Balasubramanian: "RMM: a methodology for structured hypermedia design", Communications of the ACM, 38, 8 (Aug. 1995), 33-44.
- [7] G.Mecca, P.Atzeni, A.Masci, P.Merialdo, G.Sindoni: "The Araneus Web-Based Management System". SIGMOD Conference 1998. Pp. 544-546.
- [8] P.Fraternali, P.Paolini, “The Autoweb System”; submitted for publication
- [9] E.Gamma, R.Helm, R.Johnson and J.Vlissides. Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley, 1996
- [10] F.Garzotto, P.Paolini, D.Bolchini, S.Valenti "Modeling-by-patterns of web applications ", WWWCM'99 international workshop, Paris, France, Springer-Verlag, to appear.
- [11] M.Nanard, J.Nanard, P.Kahn. "Pushing Reuse in Hypermedia Design: Golden Rules, Design Patterns and Constructive Templates". Proc. of the ACM Int. Conference on Hypertext '98, pp. 11-20.
- [12] G.Rossi, D.Schwabe, F.Lyartet, "Improving Web Information Systems with Design Patterns". Proc. of the 8th International WWW Conference, Toronto (CA), May 1999, Elsevier Science, 1999
- [13] F.Garzotto, P.Paolini, D.Schwabe, “HDM: A Model Based Approach to Hypermedia Application Design”, ACM Transactions on Information Systems, 11, 1 (Jan. 1993), 1-26
- [14] F.Garzotto, L.Mainetti, P.Paolini, “Adding Multimedia Collections to the Dexter Model”. ECHT 1994: 70-80
- [15] M.A.Bochicchio, R.Paiano, P.Paolini "JWeb: an HDM Environment for fast development of Web Applications", Proc. of IEEE ICMCS '99, Vol.2 pp.809-813
- [16] M.A.Bochicchio, P.Paolini, "An HDM Interpreter for On-Line Tutorials", Proc. of MMM '98, pp.184-190, Lausanne, Switzerland, 1998.