An HDM interpreter for on-line tutorials

Conference Paper · November 1998

DOI: 10.1109/MULMM.1998.723000 · Source: IEEE Xplore

2 authors:

Mario A. Bochicchio
Università del Salento
121 PUBLICATIONS **315** CITATIONS
SEE PROFILE

Paolo Paolini
Politecnico di Milano
115 PUBLICATIONS **2,628** CITATIONS
SEE PROFILE

An HDM Interpreter for On-Line Tutorials

Mario Bochicchio
Telemedia Lab, University of Lecce, Italy
bomal@ingle01.unile.it

Paolo Paolini
Hypermedia Open Center, Politecnico di Milano
Telemedia Lab, University of Lecce, Italy
paolini@elet.polimi.it

Abstract

The use of WWW for training purposes is rapidly growing; in a sense it could be said that training is one of the most important applications for Internet Technology, in general, and for WWW in particular. The overall organization of the sites, is, however, a difficult task, mostly if a standard structure must be enforced across different sets of content. In addition it is of growing interest the need for a compatibility between off-line training (e.g. CD-ROM's) and on-line training (e.g. WWW sites).
In this paper, a real life experience is described: a set of complex tutorials, for an industrial consortium, had to be delivered both off-line and on-line. At design time, the content was only partially known. It was important also to be able to add new material and to modify access structure, without reprogramming.
The solution has been a flexible design schema (through the model HDM) and an advanced set of tools (based on Java), that allow the cost-effective development of advanced multimedia training material.

1. Introduction and background

Complex software tools, as well as modern applications programs, rely more and more on extensive, context sensitive tutorials in order to disclose their full potential. This caused an increasing interest about the production process of multimedia tutorials.

Nowadays the realization of a good quality, error free multimedia tutorial for a complex application is an expensive task, in term of time and money, and the effectiveness of the result is hard to evaluate.

A comprehensive methodology is required [1] to master the complexity inherent to the amount of contents, the relations among them, and to implement the flexibility of use required for a truly satisfactory tutorial.

In this paper we describe a tutorial model developed on the basis of the HDM methodology; the model is discussed referring to a specific application, the "Corinto Tutorial", we developed for a consortium[1] of software houses. The tutorial model is designed to be reusable, i.e. to implement many specific tutorials, with different set of contents and different look and feels.

The tutorial runs on the top of an HDM interpreter, also outlined here. The interpreter, named JWeb, is able to concurrently serve multiple users and multiple applications (different multimedia tutorials) in an intranet environment. A stand-alone version of Jweb, named JWeb Lite, was also developed, to distribute the same tutorials by using normal CD-ROMs

The structure of the paper is the following: in section 2 we describe the tutorial model referring to the Corinto Tutorial application; in section 3 we outline the structure of the HDM interpreter; in section 4 we draw the conclusions and sketch the future directions for work; section 5 is for bibliography and references.

2. The Application

The Corinto Tutorial has been designed to support a family of software tools [2,3] developed by the Corinto Consortium. The main goal of the Consortium is the creation of tools supporting the small software houses in the transition to the Object Technology

In particular, the "Corinto Tools" follows the guidelines of the San Francisco Project, that is an international project, leaded by IBM and others, aimed to the production of extensible and customizable frameworks of applications for commercial purposes.

The Tutorial, when fully developed (sw: december '97; contents: July '98), will contain more than 1000 images, about 200 animated examples in form of Shockwave files

[1] CORINTO (COnsorzio di RIcerca Nazionale sulla Tecnologia ad Oggetti) is a consortium among IBM Italia, Apple Italia, Selfin and other software houses.

(interactive animations) and screen-cams (digital films), 10 minutes of interviews to the tool's authors, in form of digital films, diagrams, texts (Italian and English), tables, audio comments (Italian and English) and music.

The tutorial has been designed using the Hypermedia Design Model [4-8] that provides the primitives to model the application, the directives for its implementation and the precepts for an ordered development cycle.

According to HDM [4,5,8], a hypermedia application can be completely described in terms of an "Hyperbase in-the-large" schema, an "Hyperbase in-the-small" schema, an "Access Structures" plan and a "Layout Definition". The Hyperbase in-the-large outlines the main features of the application topics and the relations among them. The Hyperbase in-the-small describes the detailed structure of each application topic, and the involved multimedia objects. The Access Structures allows the user to locate the multimedia contents of his/her interest. The Layout Definition describes the look of the application, and the interaction rules.

2.1. Hyperbase

The Hyperbase in-the-large of the Corinto Tutorial, depicted in Fig. 1, consists of five main entity types:

- **Operation**: represents an action we can bring about when using the Corinto Tools (e.g. extend a given class, find a given hotspot, but also print the current form, edit the pointed object, …). The description of the action must emphasize the means of what we do, as well as the "performing mechanic" and the various available options; so, the operation must be explained both conceptually and operationally.
- **Procedure**: groups a set of operations having a global meaning (e.g. add a method to a given class, prepare a project for the test phase, …). The structure of the procedure can be linear (direct sequence), linear with mutually exclusive variants (direct tree) or linear with variants (direct lattice).
- **Result**: describes a goal we can attain using the Corinto Tools (e.g. guided project analysis, automatic generation of class diagrams, …). The goal

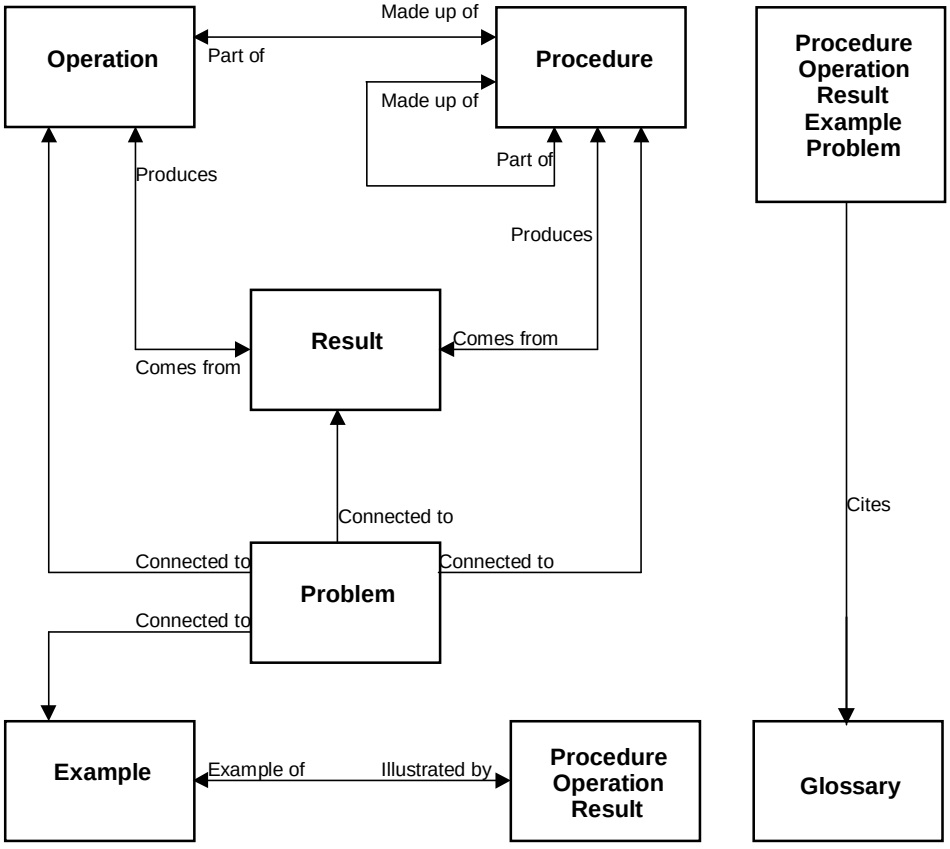

Fig. 1. Corinto tutorial: hyperbase in-the-large

description must emphasize, from the user point of view, what the Tools can do, and not how the Tools do it. This is also a good place to underline the strength points of the Corinto Tools, (comparing it with the competitors), and to anticipate the features of the future releases.

- **Problem**: illustrates a typical problem we can experience using the Tools (e.g. limits to the project size, incomplete implementation of a standard, …).
- **Example**: explain a typical case of interest showing recorded instances (animations or digital films with a suitable audio comment) of interaction with the Corinto Tools (e.g. open a new project, specialize an abstract class, …).
- A hypertextual **Glossary**, also reported in Fig. 1, provides the definitions of the terms specific to the Object Technology or to the Corinto Tools, with additional information, where it is needed.

The Application Link types, providing the semantic relationships, are:

- **Produces** / **Comes from**
  {Result}⇔{Operation; Procedure}
  Describes how to proceed (in term of procedures or operations) to achieve a given result.
- **Made up of** / **Part of**
  {Procedure}⇔{Operation}
  Describes the structure of a procedure.
- **Made up of** / **Part of**
  {Procedure}⇔{Procedure}
  Describes the recursive structure of some procedure.
- **Example of** / **Illustrated by**
  {Example}⇔{Procedure; Operation; Result}
  Exemplifies a procedure, an operation or a result.
- **Connected to**
  {Problem}⇒{Procedure;Operation;Result;Example}
  Links the problem with its cause. It is a mono directional link because, in a general purpose tutorial, the omitted direction (from right to left) is to much populated (e.g. too much potential known problems related to a given procedure) and rarely useful.
- **Cite**
  {Procedure; Operation; Result; Example; Problem} ⇒ {Glossary}
  Links a technical term, everywhere in the Tutorial, to its concise explanation or definition. We must consider that the glossary is conceived to help the understandability of the adopted language; it is inadequate to drive the navigation as a sort of “analytical index”. Consequently, this is a mono directional link.

The design in-the-small [9], not reported here for sake of briefness, takes into account the internal structure of the entity types in terms of multimedia slots (images, slide shows, films, audio, ...) and relations among them (synchronization between audio and animation, text and pictures, ...). For the Corinto Tutorial we adopted relatively simple (internally unstructured, flat) entity types, migrating each structuring need at the in-the-large level (in particular for the entity types "procedure" and "operation"). This results in a simplified production and maintenance for the multimedia contents. From the implementation point of view, in fact, each instance of the previously defined entity types can be packed as a separate file, and each single "instance file" can be created, edited or updated at any time. This allows the Corinto staff to easily modify and upgrade the tutorial contents without external intervention, and without the need to rebuild the whole tutorial at each time [9].

A template-based version of the "instance editor" has been developed by using Macromedia Director. The Shockwave technlogy (also from Macromedia) was selected for the "instance files", because of its suitability for on-line and off-line applications.

An interesting benefit of HDM is the reusability of the structure in the large of the hyperbase. In fact we can observe that the previously defined description of the hyperbase structure is not specific for the Corinto Tutorial. The same schema in the large can be reused to implement many different tutorials, even with different contents and dissimilar interfaces. In other words, the design in the large tries to capture the intrinsic meaning of the tutorial concept, making it available for different specific implementations, but also for successive refinements.

2.2. Access structures

Various fruition strategies needs to be supported in order to satisfy the various readers, from the potential customer to the neophyte of the Corinto Tools, up to the experienced user asking for advanced information on the Corinto Tools.

Therefore, we defined the following access structures:

- A guided tour is planned to explain the main concepts and the terminology of the Object Technology, as implemented in the Corinto Tools.
- A free-running collection of examples will be used for demo purposes.
- A number of heterogeneous collections, ordered by increasing skill level (lessons), will be used for an interactive training to the features of the tools and to the related operating procedures.
- A separated subset of the Hyperbase objects, organized as an automatic guided tour, will be used to

illustrate the concepts underlying the San Francisco Project [10].

- A number of heterogeneous collections, most of all containing operations and procedures, will be used for context sensitive tutoring, invoked directly from the Tools when detecting incorrect user actions.
  The detection logic needs to be embedded in the Corinto Tools.
  The invoking procedure spawns an HTML Browser instance with the entry point of the suitable HDM collection.
- The implicit collections, i.e. the entire indexes of the entities in the Hyperbase, complete the Access Structures of the Tutorials, allowing the access to whatever multimedia object in the Hyperbase. The indexes are searchable; the standard ordering is the alphabetic one, but an author-defined ordering is also implemented and available.

A substantial benefit for the advanced user consists of the possibility to create its own collections or indexes at run time (i.e. also after that the tutorial has been released to the users). Adopting the HDM terminology, in the following of the paper this activity will be referred as "second level authoring" or "co-authoring".

The co-authoring makes possible to reuse the multimedia objects in the hyperbase for its own purposes. The type of co-authoring we adopted (other types are also possible) is based on the creation of a set of multimedia objects structured in a linear sequence. An audio comment recorded at run-time or imported later, and an optional text, explains the global meaning of the collection as well as the logical steps outlined by the sequence.

From the user point of view, this is obtained by bookmarking the desired multimedia objects during a normal session, finally, the bookmark sequence can be saved as a new user-defined collection. The sequence can be completed by adding the audio comment and the text explanation [11].

In this way, for example, we can create customized copy of the existing access structures, able to better match the cultural and professional background of a specific user (or set of users). Moreover, completely new access structures can be created for specific purposes (e.g. presentations, computer supported lessons, personal bookmarks, ...).

2.3. Layout and Interactions

The main interface of the Tutorial was subdivided in five areas maintaining a constant functional meaning and a similar visual appearance across the application functioning [12,13]. The adopted layout structure is presented in Fig. 2 while in Fig. 3 is reported the corresponding (raw) prototype of the main interface.

-The area labeled "Access Structures" contains the controls to activate the author collections, the indexes and the user collections defined in the previous paragraph.

-The area "Interaction & Navigation In-The-Small" is aimed to receive the multimedia object to be played.

-The area labeled "Applications Links Navigation" encompasses the controls needed to follow the applications links. For example, referring to Fig 3, when the user is reading the procedure "Extending a CBOF Class" (i.e. an instance of the entity type "procedure"), then the button labeled "Operations" is highlighted (enabled) because they exist operations "part-of" the given procedure. This operation's index (alphabetically ordered and searchable) will pop-up by clicking on the highlighted button.

Grayed (disabled) buttons denote the absence of linked instances.

-In the lower-left area are located the "Session Controls", useful to define some general parameters such as the tutorial language, the volume level for the audio and so on.

-The area "Collection Navigation" contains the well-known controls: "first", "last", "next", "previous", plus a "go to the collection index" and a "status indicator". The last one reports the collection name, the total number of items in the collection and the number of the current item. The status indicator gives to the user the feedback needed to "estimate its position in the Tutorial" during the reading process. This information is very useful to minimize the "be lost in the Tutorial" feeling, when a complex navigation is performed [12,13].

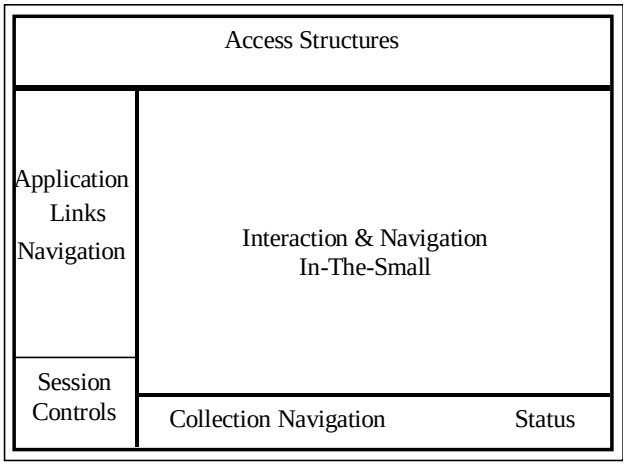

Fig. 2. Main interface: layout structure

3. The HDM interpreter

In fig. 4 is reported the simplified logical structure of the architecture we implemented to deliver the Corinto Tutorial in an intranet environment.

An intranet server (hardware) running a standard Web Server and a DBMS, is also equipped with the HDM interpreter. The interpreter, named JWeb, is a Java application communicating with the Web Server (via CGI) and with the DBMS (via JDBC [14]); the job performed by JWeb is the on-the-fly creation of the tutorial pages when requested.

A typical session of use, for the so configured intranet server, is the following:

1) a user, on the intranet, by means of a Web Browser, issues a request for a tutorial page (e.g. the home page, a bookmarked page or a page invoked by the context sensitive engine of the Corinto Tools).
2) The request, accepted by the Web Server, is redirected to JWeb (thanks to an appropriate URL format).
3) JWeb decodes the parameters contained in the URL; this gives to the interpreter the information about the current status in the form: <tutorial name, collection name, object name, others session parameters (language, skill level, …), navigation action to be performed>. The last parameter refers to the action requested by the user, in terms of navigation in the large (e.g. "go to the next object in the collection", "go to the index", "open the glossary", "open a new access structure", …).
4) The interpreter, based on the deciphered parameters, submits a query to the DBMS to obtain the needed new objects (this is a simplified explanation, more details on this step and on the next one are reported in the following of the paper).
5) On the receptions of the required multimedia object, JWeb performs the "on-the-fly composition" of the page required by the user; the result is sent to the Web Server in form of an HTML file trough the file system.
6) Finally, the Web Server send the page back to the user.

In the previous description we assumed that, at the step 4, the HDM interpreter submits the query to the DBMS on the base of the URL parameters, but this need an intermediate step. In fact the parameters contained in the URL needs to be interpreted as a function of the

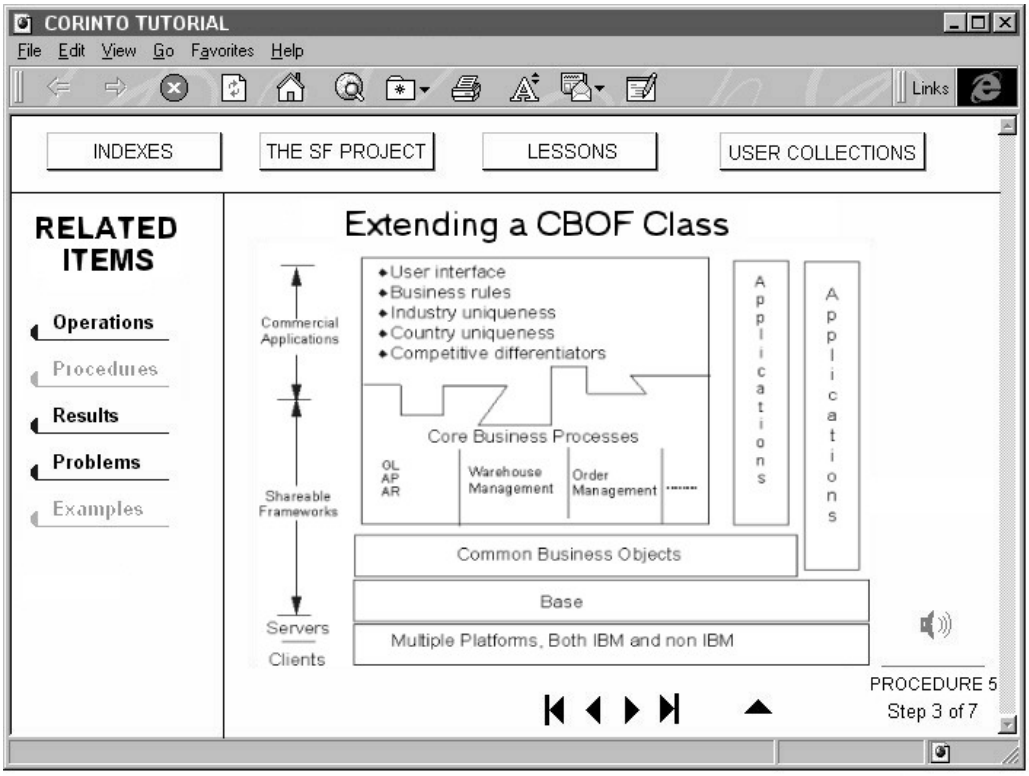

Fig. 3. Layout: prototype

Tutorial Model in order to identify the new object required by the user.

So the step 4 can be subdivided in:

4a) JWeb submits to the DBMS a first query, to obtain the components of the Tutorial Model needed to decipher the parameters in the URL;

4b) JWeb submits to the DBMS a second query, based on the deciphered parameters, to obtain the multimedia objects (in term of "instance files" and multimedia slots) needed to create the new tutorial page.

The step 4a) is essential if JWeb is used to concurrently support different Tutorial Models. However, in the case of the Corinto Tutorial, a single model is required, so that it can be loaded once for all in a suitable data structure in the main memory. This result in a reduced number of queries submitted to the DBMS and in an increased throughput of tutorial pages that JWeb can serve.

Similar considerations can be applied to the step 5, even in this case, in fact, the layout description is usually stored into the database, and a suitable query is needed to obtain the structural description of the page to create.

Moreover, it should be noted that this structural description is done in term of pathnames and filenames of the multimedia objects forming the page, rather than as binary fields stored in the database. For example, as previously stated, the contents section of each tutorial page is stored in the file system as a Shockwave file. This is a general rule: each multimedia object (e.g. the Java Applets we used to program the visual interface objects, the digital films, the audio files, the still images, …) is stored on the file system, and this is a key point of our implementation. This choice ensures, in fact, the flexibility to create, to edit and to manage the multimedia objects by using the third-part tools preferred by the authors (Macromedia Director, Adobe PhotoShop, ...)

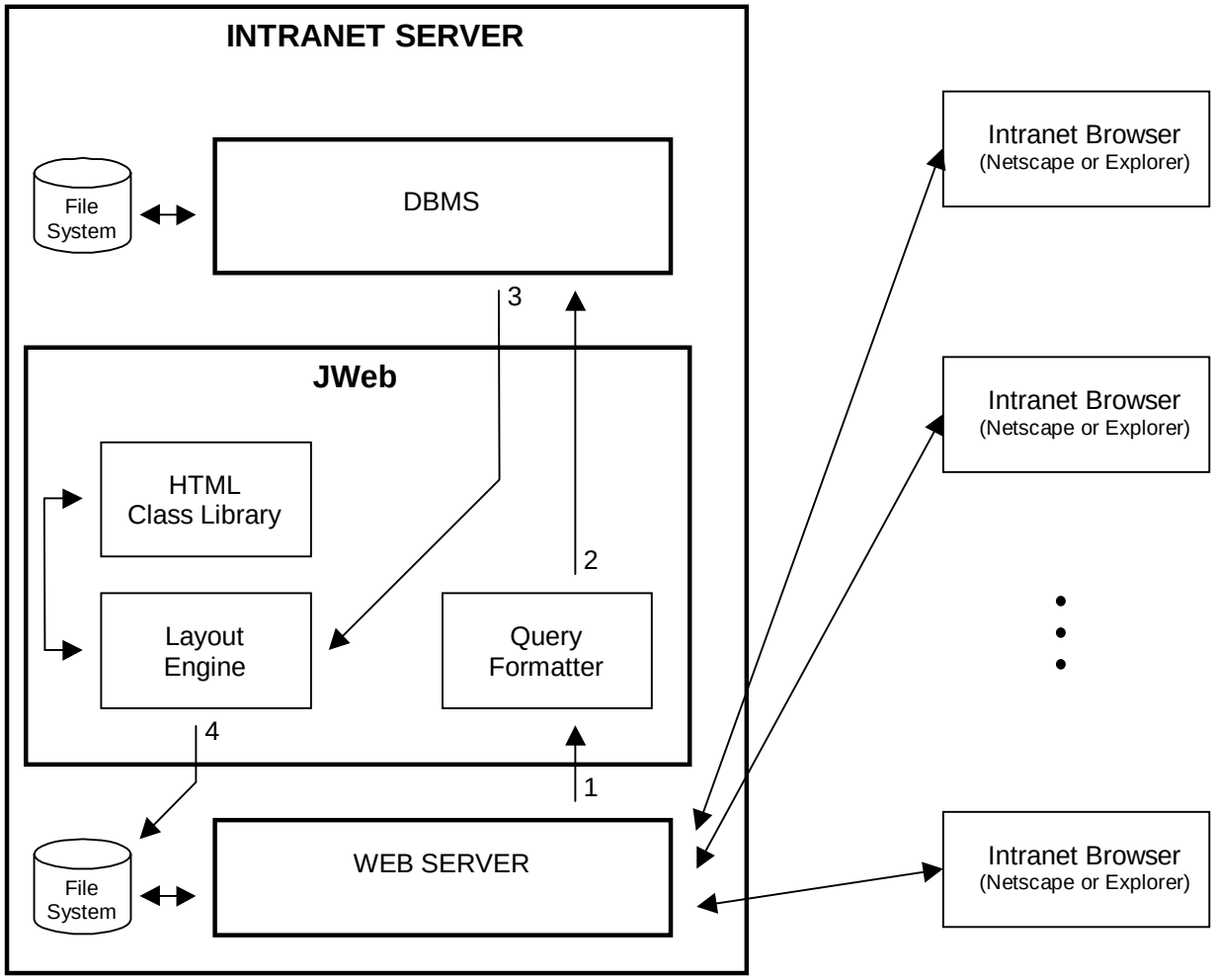

Fig. 4. JWeb: structure and interaction

instead of the rigid choice forced by many multimedia database.

An advanced prototype of JWeb has been implemented and tested on a pool of networked Win95 workstations, using JDK 1.1. Moreover, we verified the compatibility with various DBMSs (MS Access, IBM DB2 and Oracle running on various Operating Systems, via JDBC/ODBC) and various Web Servers (Apache and MS Personal Web Server).

4. Conclusions and future work

The advantages offered by the application described in the previous section could be summarized as it follows:

- a generalized tutorial schema that allows to accommodate nearly any kind of content for any type of subject;
- a data base driven implementation, with automatic generation on web pages, that allows a great flexibility in maintaining and expanding the tutorial;
- a high quality interface, ensured by the use of shockwave pieces of content;
- the possibility of defining directly specific paths, guided tours and indexes, without need for reprogramming anything; this allows the specific tailoring and tuning of the material to the needs of the training context.

As future work we are planning the following developments:

- at requirement level we need to expand the possibility of customization of the material; it would be nice, for example, to be able to cooperatively annotate parts of the content [15], to be able to insert new pieces of content, relating them to the content already stored in the tutorial, to introduce more powerful collections (i.e. indexes and guided tour, etc.);
- at design level, the schema should be slightly revised and generalized, since the current version does not fully support complex bodies of material;
- at the implementation level, we should allow a better mix between use of data base formatted data items (to be composed in a web page dynamically) and the use of “precanned” pages, as it is in the current version.

  An optimization of the code is also required to ensure the proper level of performance.

These modifications and enhancements are due by the end of 98.

5. Bibliography

[1] V. Balasubramanian, Bang Min Ma, Joonhee Yoo, "A Systematic Approach to Designing a WWW Application", Communications of the ACM, Vol. 38, N. 8, pp.47-48, 1995

[2] *Aphrodite: a Form Based Approach to the Object Oriented Analysis and Design*, Corinto internal paper, Bari, Italy, 1997.

[3] *Aphrodite Project Manual*, Corinto internal paper, Bari, Italy, 1997.

[4] D.Schwabe, G.Rossi, The Object-Oriented Hypermedia Design Model, Communications of the ACM, Vol. 38, N. 8, pp.45-46, Aug. 1995

[5] F.Garzotto, P.Paolini and D.Schwabe, “HDM - A Model Based Approach to Hypermedia Application Design”, ACM Transactions on Information Systems, 11, 1 (Jan. 1993), 1-26

[6] U.Cavallaro, F.Garzotto, P.Paolini and D.Totaro, “HIFI: Hypertext Interface for Information Systems”, IEEE Software 10, 6 (Nov. 1993), 48-51.

[7] F.Garzotto, L.Mainetti and 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.

[8] F.Garzotto, L.Mainetti and P.Paolini, “Navigation in Hypermedia Applications: Modeling and Semantics”, Journal of Organizational Computing (in press)

[9] *Corinto Tutorial: Project Manual*, Corinto internal paper, Bari, Italy, 1997.

[10] http://www.ibm.com/Java/Sanfrancisco.

[11] *Corinto Tutorial: User Manual*, Corinto internal paper, Bari, Italy, 1997.

[12] A.Dix, J.Finlay, G.Abowd and R.Beale, *Human Computer Interaction*, Prentice Hall, 1993.

[13] J.Preece, Y.Rogers, H.Sharp, D.Benyon, S.Holland and T.Carey, *Designing the User Interface*, Addison Wesley, 1994.

[14] G.Hamilton, R.Cattal and M.Fisher, *JDBC Database Access from Java: a Tutorial and Annotated Reference*, Addison Wesley, 1997.

[14] H.Benz, S.Bessler; S.Fischer; M.Hager; R.Mecklenburg, "DIANE: A Multimedia Annotation System", Proceedings of the Second European Conference on Multimedia Applications, Services and Techniques (ECMAST'97).