Annotation as a Support to User Interaction for Content Enhancement in Digital Libraries

Maristella Agosti
agosti@dei.unipd.it

Nicola Ferro
ferro@dei.unipd.it

Department of Information Engineering – University of Padua
Via Gradenigo, 6/b – 35131 Padova, Italy

Emanuele Panizzi
panizzi@di.uniroma1.it

Rosa Trinchese
trinchese@di.uniroma1.it

Department of Computer Science – University of Rome "La Sapienza"
Via Salaria, 113 – 00198 Roma, Italy

ABSTRACT

This work describes the interface design and interaction of a generic annotation service for *Digital Library Management Systems (DLMSs)*, called *Digital Library Annotation Service (DiLAS)*, that has been designed and is currently undergoing development and user test in the framework of the DELOS European Network of Excellence. The objective of DiLAS is to design and develop an architecture and a framework able to support and evaluate a generic annotation service, i.e. a service that can be easily used into different DLMSs enhancing their *User Interfaces (UIs)* in order to offer to *Digital Library (DL)* users a set of uniform, user-tested (under certain required conditions), and recognizable functionalities.

Categories and Subject Descriptors

H.3.7 [**Information Storage and Retrieval**]: Digital Libraries—*System issues*; H.5.1 [**Multimedia Information Systems**]: Evaluation/methodology; H.5.2 [**User Interfaces**]: Graphical user interfaces

General Terms

Design, Human Factors

Keywords

annotation, annotation service, multimedia document, user interface, digital library, digital library management system

1. INTRODUCTION

In most contemporary DLMSs the contents are conveyed to the user as a "collection of information items" which can be searched or browsed. However, this paradigm is often not sufficient to cope with embedded usages, for which access to the contents is not seen as an isolated activity, but as part of a larger work process, where interaction with other users, editing and annotating documents, needs to be integrated. Up to now, annotations have been - in most cases - stored together with the documents they refer to in a central DL repository. With the advent of decentralized DL architectures in Grid or *Peer-To-Peer (P2P)* environments, but also in *Service-oriented Architectures (SoA)*, these design choices need to be revised by technical solutions which allow us to manage annotations independently from a particular DLMS.

One of the goals of the *Digital Library Annotation Service (DiLAS)* project [1], which is an ongoing project in the framework of DELOS European Network of Excellence on Digital Libraries[1], is to design and develop a generic annotation service, that is a service that can be easily used into different DLMSs. The other goal is to define a set of *Application Program Interfaces (APIs)* to allow both the access to this service from different DLMSs and the creation of different annotation clients and *User Interfaces (UIs)* embedded in various DLMSs.

The paper is organized as follows: Section 2 provides background information about our research on annotation systems; Section 3 introduces the architecture of the DiLAS service and describes the design and the work conducted to integrate *Flexible Annotation Service Tool (FAST)* and *Multimedia Annotation of Digital Content Over the Web (MADCOW)*; Section 4 explains how the proposed APIs drive the design and development of UIs for DLMS; finally, Section 5 draws some conclusion and outlooks future works.

2. BACKGROUND IN ANNOTATION RESEARCH

2.1 FAST

Flexible Annotation Service Tool (FAST) is a flexible system designed to support different architectural paradigms, such as P2P or *Web Services (WS)* architectures, and a wide range of different DLMSs. The flexibility of FAST and its

[1] `http://www.delos.info/`

AVI '06, May 23–26, 2006, Venezia, Italy.

independence from any particular DLMS is a key feature to provide users with a uniform way of interaction with annotation functionalities, without the need of changing their annotative practices only because a user works with different DLMSs. Furthermore, annotations create an hypertext that allows users to merge their personal content with the information resources provided by diverse DLMSs: this hypertext can span and cross the boundaries of a single DLMS, if users need to interact with diverse DLMSs [2]. The possibility of having a hypertext that spans the boundaries of different DLMSs is quite innovative because up to now such hypertext is usually confined within the boundaries of a single DLMS. Moreover, DLMSs do not usually offer hypertext management functionalities, since they do not normally have a hypertext connecting information resources with each other. Thus, annotations can be a way of associating a hypertext to a DL in order to enable an active and dynamic usage of information resources [3].

Differently from today's systems, FAST has to avoid any constraints concerning both the annotated information resource and the available protocols. The only assumption about information resources that FAST can make is that each information resource is uniquely identified by a *handle*, which is a name assigned to an information resource in order to identify and facilitate the referencing to it, such as a *Uniform Resource Identifier (URI)* or a *Digital Object Identifier (DOI)*.

From an architectural point of view, FAST adopts a three-layers architecture – the data, application and interface logic layers – and is designed at a high level of abstraction in terms of abstract APIs [2]. In this way, we can model the behaviour and the functioning of FAST without worrying about the actual implementation of each component. Different alternative implementations of each component could be provided, still keeping a coherent view of the whole architecture of the FAST system. We achieve this abstraction level by means of a set of interfaces, which define the behaviour of each component of FAST in abstract terms. Then, a set of abstract classes partially implement the interfaces in order to define the actual behaviour common to all of the implementations of each component. Finally, the actual implementation is left to the concrete classes, inherited from the abstract ones, that fit FAST into a given architecture. Java[2] is the programming language in use for developing FAST.

2.2 MADCOW

Multimedia Annotation of Digital Content Over the Web (MADCOW)[3] [5] has a client-server architecture, where the client is a plug-in for a standard Web browser and the servers are repositories of annotations to which different clients can login. The possibility of digitally annotating digital documents introduces three important novelties: (i) the original document can be left unaltered and the content of the annotation can be stored in a different document; (ii) the document (web-note) resulting from the annotation activity can still be held privately or with restricted circulation within an organization; (iii) the content of the annotation is not restricted to text, but can include any form of multimedia material.

[2]`http://java.sun.com`
[3]`http://www.web-notes.com`

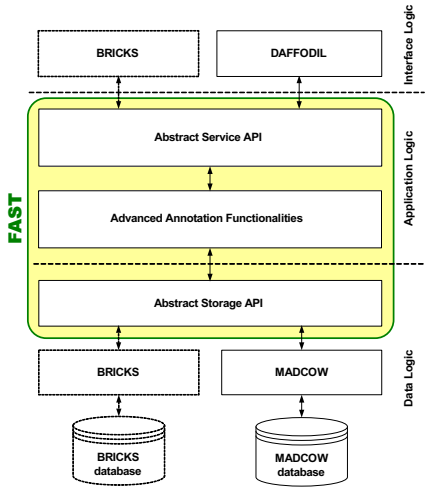

Figure 1: Architecture of the DiLAS service.

Specific UIs for retrieving and filtering annotations have been defined, as well as for establishing a default annotation server for a document. A webnote is a digital annotation consisting of two main components: metadata and content. The first is a set of attributes like, type, author, title, creation date, modification date, location, *Uniform Resource Locator (URL)*, public or private visibility. The second component is the multimedia content, which is coded into an *HyperText Markup Language (HTML)* document containing, besides text written by the user, links to other multimedia documents chosen by the same user to integrate the annotation; that can contain textual information, video, image or audio files, as inserted by the user. MADCOW allows different pre-established types of annotations, such as explanation, comment, question, solution, summary, and so on. The type of the annotation is one from an enumeration of values which describe the different functional roles that the annotation plays with respect to the annotated object [4]. When a document is loaded in a browser, presenting the WebNotes toolbar, the placeholders (icons showing the annotation type) corresponding to the annotations associated to the document are automatically shown on the document itself, thus providing links to HTML pages visualizing the annotation. Since webnotes are presented as HTML pages which can be annotated in turn and a discussion track may be produced, where different positions are reflected.

3. DILAS SYSTEM ARCHITECTURE

The architecture of the DiLAS system, shown in Figure 1, consists of three layers – the data, application and interface logic layers.

The data logic layer manages the actual storage of the annotations and provides a persistence layer for storing the objects which represent the annotation and which are used by the upper layers of the architecture. In order to make it as flexible as possible, the *Abstract Storage API* for the functionalities of the storage has been defined. This API, in turn, allows for accessing different system to perform the actual storage of the annotations. In the first prototype of the DiLAS system we use the MADCOW system as actual storage for the annotations, but for the final prototype we

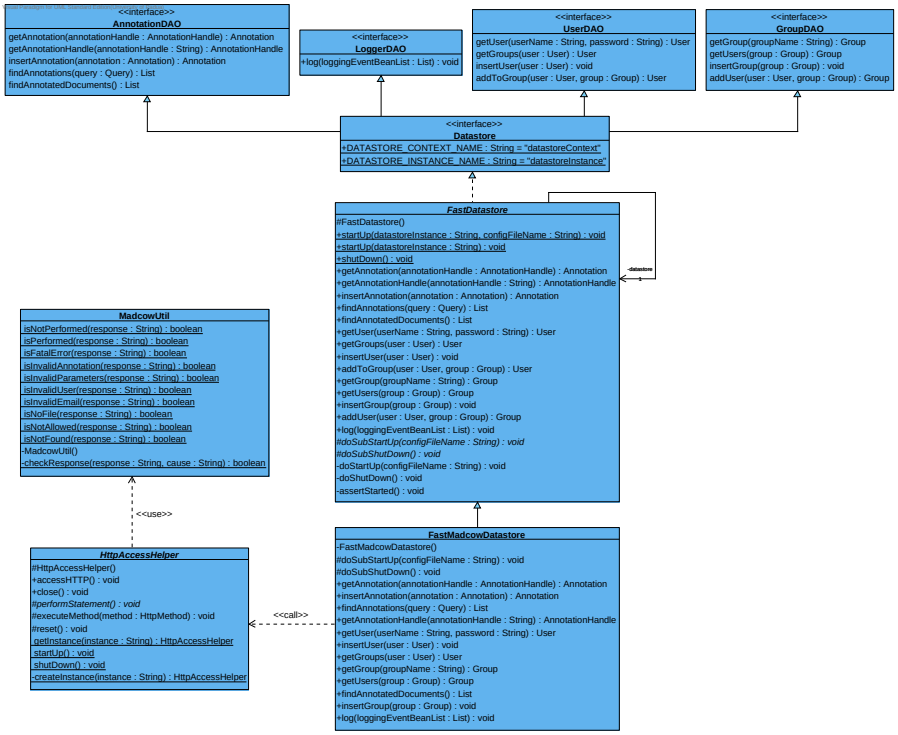

Figure 2: UML class diagram of the designed API and its prototype implementation.

are going to integrate also the *Building Resources for Integrated Cultural Knowledge Services (BRICKS)* system as storage provider.

The application logic layer provides advanced functionalities that make use of annotations, such as for example the search and retrieval of annotations described above.

Note that the application logic and the upper part of the data logic correspond to the respective layers in the FAST system. As already said, within FAST these layers are defined by means of interfaces, as well as the business objects exchanged in these layers. Thus, the integration of FAST and MADCOW requires a new implementation of these layers and business objects, in order to fit them to the needs of both systems.

The interface logic layer is devoted to manage the interaction with the end-user. It depends on the system into which DiLAS is going to be used and relies on the *Abstract Service API* in order to provide the functionalities described above to the end user. For the first prototype of DiLAS we use the *Distributed Agents for User-Friendly Access of Digital Libraries (DAFFODIL)* system in order to carry out the desired user-level use cases

With respect to the architecture of Figure 1, the integration of FAST and MADCOW requires on the one hand to provide an implementation of the "Abstract Storage API" suitable for communicating with MADCOW and, on the other hand, to adapt MADCOW to exchange the messages expected by the storage API.

According to the architectural approach adopted in FAST and described in Section 2.1, Figure 2 shows the *Unified Modeling Language (UML)* class diagram of both the designed API and its first implementation, developed with the Java programming language. The proposed API corresponds to the `Datastore` interface, which is a façade for different interfaces: `AnnotationDAO`, `UserDAO`, `GroupDAO`, and `LoggerDAO`. These interfaces are designed according to the *Data Access Object (DAO)* design pattern[4], which implements the access mechanism required to work with the underlying data source. Each of the interfaces introduced above takes care of defining the operation needed to ensure the persistence of the different objects managed by the system, that is `Annotation`, `User`, `Group` of users, and `Logger` for logging system activities. Figure 2 shows all the different functionalities offered by the `Datastore` interface, which is initially implemented by an abstract class, called `FastDatastore`, which provides the basic functionalities and the input parameters checks common to all the concrete implementations.

The proposed API has been implemented by creating a subclass of `FastDatastore`, called `FastMadcowDatastore`, which provides an implementation of the `Datastore` interface, able to communicate with the MADCOW system. As discussed in Section 2.2, MADCOW is a web-based system which communicates with its own protocol over *HyperText Transfer Protocol (HTTP)*. Thus, `FastMadcowDatastore` translates the functionalities described by the `Datastore` interface into the HTTP calls supported by MADCOW and exchanges with it *eXtensible Markup Language (XML)* encoded messages, according to the XML schemas defined for MADCOW. To this end, `FastMadcowDatastore` relies on an helper class, called `HttpAccessHelper`, which makes use of the *Jakarta Commons HttpClient*[5] version 3.0rc4, in order

[4] `http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html`

[5] `http://jakarta.apache.org/commons/httpclient/`

Figure 3: Annotate dialog window.

to hide the details of the communication with MADCOW over HTTP, and utilized the `MadcowUtil` class for interpreting the responses received by MADCOW.

4. INTERFACE AND INTERACTION

The DiLAS specification drives the design of the interface and of the interaction with the provided services in the DLMS. In fact, from one point of view, the API specification defines the functionalities that the interface can offer to the DL user; on the other hand, the results of user study we are undergoing will give hints on the possible use of annotations by DL users, on their behaviour and on their typical errors, and will thus act as a guideline to the design of the interaction of the new features that will be integrated and will enhance the existing DLMS interface.

The most important constraints that must be taken into account in designing the UI are:

- the need to identify univocally each document;
- the modalities of selection of the source document or of a part of it; this can take advantage from user studies related to Web pages conducted in MADCOW;
- the metadata to be filled in, with their relative importance and priority, e.g. only the most relevant and used metadata fields can be showed in dialog windows;
- the need to univocally identify annotations in the DLMS, so that they can be reannotated in turn;
- the metadata used by the annotation search service;
- the user authentication constraints.

Thus, the above constraints and the guidelines that will result from the user study, will limit and drive the interface design, but will also allow for different choices in the design and implementation of the UI functionalities, also according to the existing interaction designed for each specific DLMS.

The first interface prototype, DAFFODIL, is the test bench for the API, for the usability study and refinement of such criteria, while the implementation of the BRICKS UI, which will be prepared later, will let us test the guidelines and the whole process of interface integration.

When users browse the web and want to create an annotation, they first select an object in the web page and then fill the dialog window showed up (see figure 3). The user specifies all the metadata that describe the annotation (title of annotation, RST type, public or private visibility) and annotation content: text and optional attachments. Some fields (such as creation date, modification date and author name) are automatically filled in by the system. When the user saves the annotation, the icon corresponding to the selected type is inserted as a placeholder in the web page.

The system allows users to search any public annotation created by any other user. In the search annotation dialog window some parameters from the metadata appear as selection criteria, e.g. server name, annotation type, annotation kind, etc. When the user clicks on the search button, the list of results that satisfy the parameters appeared. Users can refine the search or open an annotation.

5. CONCLUSIONS AND FUTURE WORK

This paper discussed the architecture of the DiLAS system, which can be integrated into different DLMSs in order to exploit annotations as an active and effective collaboration tool for users.

The first prototype of DiLAS is built on the integration of two existing systems, FAST and MADCOW, which will provide the basic functionalities of the annotation service, so that it can be used by the target systems, BRICKS and DAFFODIL, and we will perform a user evaluation with these two system, in order to assess the design and implementation choices of DiLAS.

Acknowledgments

The work was partially supported by the DELOS Network of Excellence on Digital Libraries, as part of the Information Society Technologies (IST) Program of the European Commission (Contract G038-507618).

6. REFERENCES

[1] M. Agosti, H. Albrechtsen, N. Ferro, I. Frommholz, P. Hansen, N. Orio, E. Panizzi, A. M. Pejtersen, and U. Thiel. DiLAS: a Digital Library Annotation Service. In *International Workshop on Annotation for Collaboration. Methods, Tools, and Practices*, pages 91–101. Paris, November 24-25, 2005.

[2] M. Agosti and N. Ferro. A System Architecture as a Support to a Flexible Annotation Service. In *Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures: 6th Thematic Workshop of the EU Network of Excellence DELOS. Revised Selected Papers*, pages 147–166. LNCS 3664, Springer, Heidelberg, Germany, 2005.

[3] M. Agosti, N. Ferro, I. Frommholz, and U. Thiel. Annotations in Digital Libraries and Collaboratories – Facets, Models and Usage. In *Proc. 8th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2004)*, pages 244–255. LNCS 3232, Springer, Heidelberg, Germany, 2004.

[4] P. Bottoni, R. Civica, S. Levialdi, L. Orso, E. Panizzi, and R. Trinchese. MADCOW: a Multimedia Digital Annotation System. In *Proc. Working Conference on Advanced Visual Interfaces (AVI 2004)*, pages 55–62. ACM Press, New York, USA, 2004.

[5] P. Bottoni, R. Civica, S. Levialdi, L. Orso, E. Panizzi, and R. Trinchese. Digital Library Content Annotation with the MADCOW System. In *Proc. 7th International Workshop of the EU Network of Excellence DELOS on Audio-Visual Content and Information Visualization in Digital Libraries (AVIVDiLib'05)*, pages 111–116. Centromedia, Viareggio, Italy, 2005.