A System Architecture as a Support to a Flexible Annotation Service

Maristella Agosti and Nicola Ferro

Department of Information Engineering – University of Padua,
Via Gradenigo, 6/b – 35131 Padova – Italy
{maristella.agosti, nicola.ferro}@unipd.it

Abstract. Digital Library Management Systems are systems that are able to manage collections of digital documents that form Digital Libraries and Digital Archives, and they are currently in a state of evolution. Today, most of the times they are simply places where information resources can be stored and made available, whereas for tomorrow they are becoming an integrated part of the way the user works. To cooperate towards reaching this new type of system, a digital library management system must become a tool that constitutes an active part of the intellectual production process.

Annotations are effective means in order to enable an effective interaction between users and digital library management systems, since they are a very well-established practice and are widely used. Annotations are not only a way of explaining and enriching an information resource with personal observations, but also a means of transmitting and sharing ideas in order to improve collaborative work practices. Furthermore, annotations represent a bridge between reading and writing, that facilitates the user's first approach when they begin dealing with an information resource. Thus, a service able to support annotation capabilities of collection of digital documents can be appealing to the user's needs.

This paper presents the main features of a flexible system capable of managing annotations in an automatic way in order to support users and their annotative practices. Indeed, a flexible architecture allows the design of a system with a widespread usage, so that users can benefit from its functionalities without limitations due to the architecture of a particular system. We named this system *Flexible Annotation Service Tool* and this paper is devoted to introduce most relevant design choices and characteristics of it.

1 Introduction

Nowadays, the notion of isolated information resources or applications is increasingly being replaced by a distributed and networked environment, where there is almost no distinction between local and remote information resources and applications. Indeed, a wide range of new technologies allow us to envision ubiquitous and pervasive access to information resources and applications. A wide range of wired and wireless technologies make it possible to offer almost ubiquitous connectivity; examples of such technologies are *Local Area Networks (LANs)*, *Wireless LANs (WLANs)*, *Asymmetric Digital Subscriber Line (ADSL)* and other broadband connections, *Third Generation Mobile System (3G)* networks as *Universal Mobile Telecommunication System*

C. Türker et al. (Eds.): P2P, Grid, and Service Orientation ..., LNCS 3664, pp. 147–166, 2005.

(UMTS) networks. Moreover, a variety of devices, that range from desktop computers to *Personal Digital Assistants (PDAs)*, mobile phones, and other handheld devices [1], and a series of emerging architectural paradigms, such as *Web Services (WS)*, *Peer-To-Peer (P2P)* and Grid architectures, are now available and allow us to design and develop services and systems that are more and more user-centered.

In particular, *Digital Librarys (DLs)*, as information resources, and *Digital Library Management Systems (DLMSs)*, that manage DLs, are currently in a state of evolution: today they are simply places where information resources can be stored and made available, whereas for tomorrow they will become an integrated part of the way the user works. For example, instead of simply downloading a paper and then working on a printed version, a user will be able to work directly with the paper by means of the tools provided by the DLMS and share their work with colleagues. This way, the user's intellectual work and the information resources provided by the DLMS can be merged together in order to constitute a single working context. Thus, the DL is no longer perceived as something external to the intellectual production process or as a mere consulting tool, but as an intrinsic and active part of the intellectual production process.

Annotations are effective means in order to enable this new paradigm of interaction between users and DLMSs, since they are a very well-established practice and widely used. Annotations are not only a way of explaining and enriching an information resource with personal observations, but also a means of transmitting and sharing ideas in order to improve collaborative work practices. Furthermore, annotations represent a bridge between reading and writing, that facilitates the user's first approach when they begin dealing with an information resource; thus, a DLMS offering annotation capabilities can be appealing to the user's needs. Finally, annotations allow users to naturally merge personal contents with the information resources provided by the DLMS, making it possible to embody the paradigm of interaction between users and DLs which has been envisaged above. We aim at designing a system capable of managing annotations in an automatic way in order to support users and their annotative practices.

2 Annotations

Over past years a lot of research work regarding annotations has been done [2]. All of this research work has led to different viewpoints about what an annotation is. These viewpoints are discussed in the following.

2.1 Metadata

Annotations can be considered as additional data which concern an existing content, that is annotations are metadata, because they clarify in some way the properties and the semantics of the annotated content.

[3,4] propose a data model for the composition and metadata management of documents in a distributed setting, such as a DLMS. They allow the creation of *composite documents*, that are made up of either composite documents or *atomic documents*, that can be any piece of material uniquely identifiable. A set of annotations is automatically

associated to each composite document starting from the annotations of its composing atomic documents, where [3,4] interpret annotations as terms taken from a controlled vocabulary or taxonomy to which all authors adhere.

The Annotea[1] project developed by the *World Wide Web Consortium (W3C)* [5] considers annotations as metadata and interprets them as the first step in creating an infrastructure that will handle and associate metadata with content towards the Semantic Web. Annotea uses *Resource Description Framework (RDF)* and *eXtensible Markup Language (XML)* for describing annotations as metadata and XPointer for locating the annotations in the annotated document. Annotea employs a client-server architecture based on *HyperText Transfer Protocol (HTTP)*, where annotations reside in dedicated servers and a specialized browser is capable of retrieving them upon request, when visiting a Web page.

Annotations are used also in the context of *DataBase Management Systems (DBMSs)* and, in particular, in the case of *curated databases* and *scientific databases*. SWISS-PROT[2] is a curated protein sequence database, which strives to provide a high level of annotation, such as the description of the function of a protein, its domains structure, and so on. In this case, the annotations are embedded in the database and merged with the annotated content. BIODAS[3] provides a *Distributed Annotation System (DAS)*, that is a Web–based server system for sharing lists of annotations across a certain segment of the genome. In this case, annotations are not mixed together with the content they annotate, but they are separated from it. In this context, [6] proposes an archiving technique in order to manage and archive different versions of such kind of databases, as time moves on. [6] exploits the hierarchical structure of scientific data in order to represent the content and the different versions of the database with a tree structure, and attaches annotations to the nodes of the tree, annotations that contain time–stamp and key information about the underlying data structure. Thus, these annotations are metadata about the database itself. These annotations are different from the annotations contained in the database, that are metadata about genome sequences.

[7,8] investigate the usage of annotations with respect to the *data provenance* problem, which is the description of the origins of a piece of data and the process by which it arrived in a database. Data provenance is a relevant issue in the field of curated and scientific databases, such as genome databases, because experts provide corrections and annotations to the original data, as time moves on. It is now clear that data provenance is essential to any user interested in the accuracy and timeliness of the data, especially for understanding the source of errors in data and for carrying annotations through database queries. [9] proposes and implements an extension to a relational DBMS and an extension to *Structured Query Language (SQL)*, called *propagate SQL (pSQL)*, which provides a clause for propagating annotations to tuples through queries. [9] intends annotations to be information about data such as provenance, comments, or other types of metadata; [9] envisages the following applications of annotations in DBMS: tracing the provenance and flow of data, reporting errors or remarks about a piece of data, and describing the quality or the security level of a piece of data.

[1] `http://www.w3.org/2001/Annotea/`
[2] `http://www.expasy.org/sprot/`
[3] `http://biodas.org/`

2.2 Contents

Annotations are additional contents which concern an existing content [2]; indeed, they increase existing content by providing an additional layer of content that elucidates and explains the existing one. This viewpoint about annotations entails an intrinsic dualism between annotation as content enrichment and annotation as stand-alone document [10]:

- *annotation as content enrichment*: in this view annotations are considered as mere additional content regarding an existing document and so they are not autonomous entities but in fact they rely an already existing information resource in order to justify their existence;
- *annotation as stand-alone document*: in this view annotations are considered as real documents and are autonomous entities that maintain some sort of connection with an existing document.

This twofold nature of the annotation is clear if we think about the process of studying a document: firstly, we can start annotating some interesting passages that require an in depth investigation, which is an annotation as content enrichment; then we can reconsider and collect our annotations and we can use them as a starting point for a new document, covering the points we would like to explain better which is an annotation as a stand-alone document. In this case the annotation process can be seen as an informal, unstructured elaboration that could lead to a rethinking of the annotated document and to the creation of a new one. Systems like COLLATE [11,12] or IPSA [13,14,15] support this task through annotations.

Different layers of annotations can coexist on the same document: a private layer of annotations accessible only by the annotations authors themselves, a collective layer of annotations, shared by a team of people, and finally a public layer of annotations, accessible to all the users of the digital library. In this way, user communities can benefit from different views of the information resources managed by the DLMS [16,17]. A DLMS can encourage cooperative work practices, enabling the sharing of documents and annotations, also with the aid of special devices, such as XLibris [18]. Finally, as suggested in [19,20], searching, reading and annotating a DL can be done together with other activities, for example working with colleagues. This may also occur in a mobile context, where merging content and wireless communication can foster ubiquitous access to DLMSs, improving well established cooperative practices of work and exploiting physical and digital resources. The wireless context and the small form factor of handheld devices challenge our technical horizons for information management and access and require specialized solutions in order to overcome the constraints imposed by such kinds of devices, as analysed in [1].

As a further example, *Multimedia Annotation of Digital Content Over the Web (MADCOW)* is based on a client-server architecture as Annotea is. Servers are repositories of annotations to which different client can connect, while the client is a plug-in for a standard Web browser [21,22]. MADCOW employs HTTP in order to annotate Web resources and allows both private and public annotations. Moreover, it allows different pre-established types of annotations, such as explanation, comment, question, solution, summary, and so on.

2.3 Hypertext

Annotations allow the creation of new relationships among existing contents, by means of links that connect annotations together and with existing content. In this sense we can consider that existing content and annotations constitute a hypertext, according to the definition of hypertext provided in [23]. This hypertext can be exploited not only for providing alternative navigation and browsing capabilities, but also for offering advanced search functionalities. Furthermore, [24] considers annotations as a natural way of creating and growing hypertexts that connect information resources in a DLMS by actively engaging users. Finally, the hypertext existing between information resources and annotations enables different annotation configurations, that are *threads of annotations*, i.e. an annotation made in response to another annotation, and *sets of annotation*, i.e. a bundle of annotations on the same passage of text [10,25].

2.4 Dialog Acts

Annotations are part of a discourse with an existing content. For example, [11,26] consider annotations as the document context, intended as the context of the collaborative discourse in which the document is placed. Also [27] agree, to some extent, with this viewpoint about annotations. Indeed, they interpret annotations as a means that allow a "two way exchange of ideas between the authors of the documents and the documents users".

3 Architectural Approach

Annotations have a wide range of usages in different *Information Management Systems (IMSs)*, ranging from DBMSs to DLMSs and corresponding to the different viewpoints about annotations, introduced in Section 2. Annotations are a key technology for actively involving users with an IMS and this technology should be available for each IMS employed by the user. Indeed, the user should benefit from a uniform way of interaction with annotation functionalities, without the need of changing their annotative practices only because a user works with different IMSs. Furthermore, annotations create an hypertext that allows users to merge their personal content with the information resources provided by diverse IMSs, according to the scenario envisaged in Section 1: this hypertext can span and cross the boundaries of a single IMS, if users need to interact with diverse IMSs. The possibility of having a hypertext that spans the boundaries of different IMSs is quite innovative because up to now such hypertext is usually confined within the boundaries of a single IMS. Moreover, IMSs do not usually offer hypertext management functionalities; for example, DLMSs 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 [25]. Finally, there are many new emerging architectural paradigms, such as P2P or WS architectures, that have to be taken into account.

Thus, our architectural approach is based on *flexibility*, because we need to adopt an architecture which is flexible enough to support both various architectural paradigms

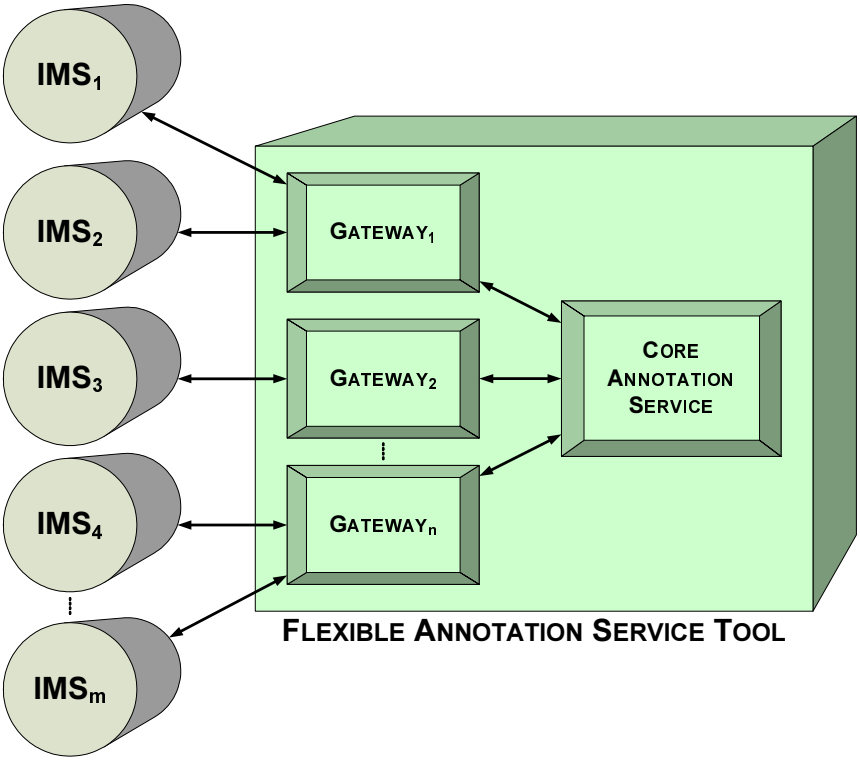

Fig. 1. Overview of the architecture of FAST with respect to different IMSs

and a wide range of different IMSs. Indeed, a flexible architecture allows the design of a system with a widespread usage, so that users can benefit from its functionalities without limitations due to the architecture of a particular IMS. Since our target system is flexible, we named it *Flexible Annotation Service Tool (FAST)*. In order to fulfil the requirements introduced above, our architectural approach is twofold:

1. to make FAST a stand-alone system, i.e. FAST is not part of any particular IMS;
2. to separate the core functionalities of the annotation service, from the functionalities needed to integrate it into different IMSs.

Figure 1 shows the general architecture of the FAST system and its integration with different IMSs: the *Core Annotation Service (CAS)* is able to interact with different gateways, that are specialised for integrating the CAS into different IMSs. From the standpoint of an IMS the FAST system acts like any other distributed service of the IMS, even if it is actually made up of two distinct modules, the gateway and the CAS; on the other hand, the FAST system can be made available for another IMS by creating a new gateway. Note that the additional layer introduced by the gateway allows the integration of the CAS also with legacy systems, that may benefit from the availability of annotation functionalities. The choice of making FAST a stand-alone system

is coherent with the approach adopted by different systems: for example, Annotea by the W3C, MADCOW, and BIODAS rely on stand-alone servers, that store and manage annotations separated from the annotated objects. On the other hand, the choice of separating the core functionalities of the annotation service, from the functionalities needed to integrate it into the different IMSs is quite new. In fact, you will not be able to find an architecture like this in the literature about annotation systems, to the best of our knowledge.

As a consequence of this architectural choice, it is worth pointing out that the FAST system knows everything about annotations, however it cannot do any assumption regarding the information resources provided by the IMS, being that it needs to cooperate with different IMSs.

This situation is very different from what is commonly found today. For example, both Annotea and MADCOW are stand-alone systems but they are targeted to work with Web pages. Indeed, they assume that the annotated object has a structured compliant with *HyperText Markup Language (HTML)*, as an example, and that they can use HTTP to transport annotations. On the contrary, FAST cannot assume that it is dealing with either HTML documents or the HTTP protocol, but it 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. This assumption is coherent with the assumption made by [3,4] who refer to and compose documents only by identifiers and annotate them with metadata from a taxonomy of terms.

Over the past years, various syntaxes, mechanisms, and systems have been developed in order to provide handles or identifiers for information resources. The mechanisms and the standards discussed in the following are all suitable to be used as handles, according to the assumption made above.

URI - URN - URL. The *Internet Engineering Task Force (IETF)*[4] defines: *Uniform Resource Identifier (URI)*, *Uniform Resource Name (URN)*, and *Uniform Resource Locator (URL)*. An URI [28] is a compact string of characters for identifying an abstract or physical resource. URIs are characterized by the following definitions:

- *uniform*: it allows different types of resource identifiers to be used in the same context, even when the mechanisms used to access those resources may differ; it allows uniform semantic interpretation of common syntactic conventions across different types of resource identifiers; it allows introduction of new types of resource identifiers without interfering with the way that existing identifiers are used; and, it allows the identifiers to be reused in many different contexts, thus permitting new applications or protocols to leverage a pre-existing, large, and widely-used set of resource identifiers;
- *resource*: a resource can be anything that has identity. Not all resources are network "retrievable"; e.g., human beings, corporations, and library books can be considered

[4] `http://www.ietf.org/`

resources as well. The resource is the conceptual mapping to an entity or set of entities. Thus, the resource does not necessarily have to correspond to the mapped entity at any given time, instead it is the conceptual mapping itself. In conclusion, a resource can remain constant even when its content—the entities to which it currently corresponds—changes over time, provided that the conceptual mapping is not changed in the process;
- *identifier*: an identifier is an object that can act as a reference to something that has an identity. In the case of URI, the object is a sequence of characters with a restricted syntax.

The term URL refers to the subset of URIs that identify resources via a representation of their primary access mechanism (e.g., their network "location"), rather than identifying the resource by name or by some other attribute(s) of that resource. The term URN refers to the subset of URI that are required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable.

DOI. The *International DOI Foundation (IDF)*[5] defines the *Digital Object Identifier (DOI)*, which is an *actionable identifier* for intellectual property on the Internet. Firstly, the IDF defines an identifier from different viewpoints:

(1) an identifier is *an unambiguous string or "label" that references an entity*. An example of such an identifier is the *International Standard Book Number (ISBN)*[6], which is a unique number assigned to a title or edition of a book or other monographic publication (serial publications excluded) published or produced by a specific publisher or producer;
(2) an identifier is *a numbering scheme*, such as a formal standard, an industrial convention, or an arbitrary internal system. This numbering scheme provides a consistent syntax for generating individual labels or identifiers, as stated in (1), that denote and distinguish separate members of a class of entities; we can still use the ISBN, as an example. The intention is establishing a one–to–one correspondence between the members of a set of labels (numbers), and the members of the set counted and labelled. An important point is that the resulting number is simply a label string, but it does not create a string that is "actionable" in a digital or physical environment without further steps being taken;
(3) an identifier is *an infrastructure specification*: a syntax by which any identifier as stated in (1) can be expressed in a suitable form for use with a specific infrastructure, without necessarily specifying a working mechanism; an example of such an identifier is the URI. This is sometimes known as creating an "actionable identifier" which means that in the context of that particular piece of infrastructure, the label can now be used to perform some action;
(4) an identifier is *a system for implementing labels (identifiers as stated in (1)) through a numbering scheme (identifiers as stated in (2)) in an infrastructure using a specification (identifiers as stated in (3)) and management policies*. This sense of "identifier" denotes a fully implemented identification mechanism that includes the ability

[5] `http://www.doi.org/`
[6] `http://www.isbn-international.org/`

to incorporate labels, conforms to an infrastructure specification, and adds to these practical tools for the implementation such as registration processes, structured interoperable metadata, and an administrative mechanism.

The DOI is a system which provides a mechanism to interoperably identify and exchange intellectual property in the digital environment. It is an identifier as stated in (4) above. One of the components is a syntax specification (identifier as stated in (2)). DOI conforms to a URI (identifier as stated in (3)) specification. It provides an extensible framework for managing intellectual content based on proven standards of digital object architecture and intellectual property management, and it is an open system based on non-proprietary standards.

OpenURL. The *National Information Standards Organization (NISO)* Committee AX[7] defines the OpenURL framework, which aims at standardizing the construction of "packages of information" and the methods by which they may be transported over networks. The intended recipients of these packages are networked service providers that deliver context-sensitive services. To enable such services, each package describes not only the resource for which services are needed, but also the network context of a reference to the resource in question. Thus, OpenURL is a standard syntax for transporting information (metadata and identifiers) about one or multiple resources within URLs, i.e. it provides a syntax for encoding metadata and identifiers, limited to the world of URLs.

PURL. The *Online Computer Library Center (OCLC)* defines the *Persistent URL (PURL)*[8], which is an URL from a functional standpoint. However, instead of pointing directly to the location of an Internet resource, a PURL points to an intermediate resolution service, that associates the PURL with the actual URL and returns that URL to the client as a standard HTTP redirect. The client can then complete the URL transaction in the normal fashion.

The *PURL-based Object Identifier (POI)*[9] is a simple specification for resource identifiers based on the PURL system, closely related to the use of the *Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)* defined by the *Open Archives Initiative (OAI)*[10]. The POI is a relatively persistent identifier for resources that are described by metadata "items" in OAI-compliant repositories. Where this is the case, POIs are not explicitly assigned to resources – a POI exists implicitly because an OAI "item" associated with the resource is made available in an OAI-compliant repository. However, POIs can be explicitly assigned to resources independently from the use of OAI repositories and the OAI-PMH, if desired.

Lexical Signatures [29] makes a proposal for identifying Web documents that is different from what has been discussed up to now. Indeed, the *Lexical Signatures (LSs)* aim at uniquely identifying a Web document by means of a signature extracted from its content and not by means of using some identifiers, as in the case of URLs.

[7] `http://www.niso.org/committees/committee_ax.html`
[8] `http://purl.oclc.org/`
[9] `http://www.ukoln.ac.uk/distributed-systems/poi/`
[10] `http://www.openarchives.org/`

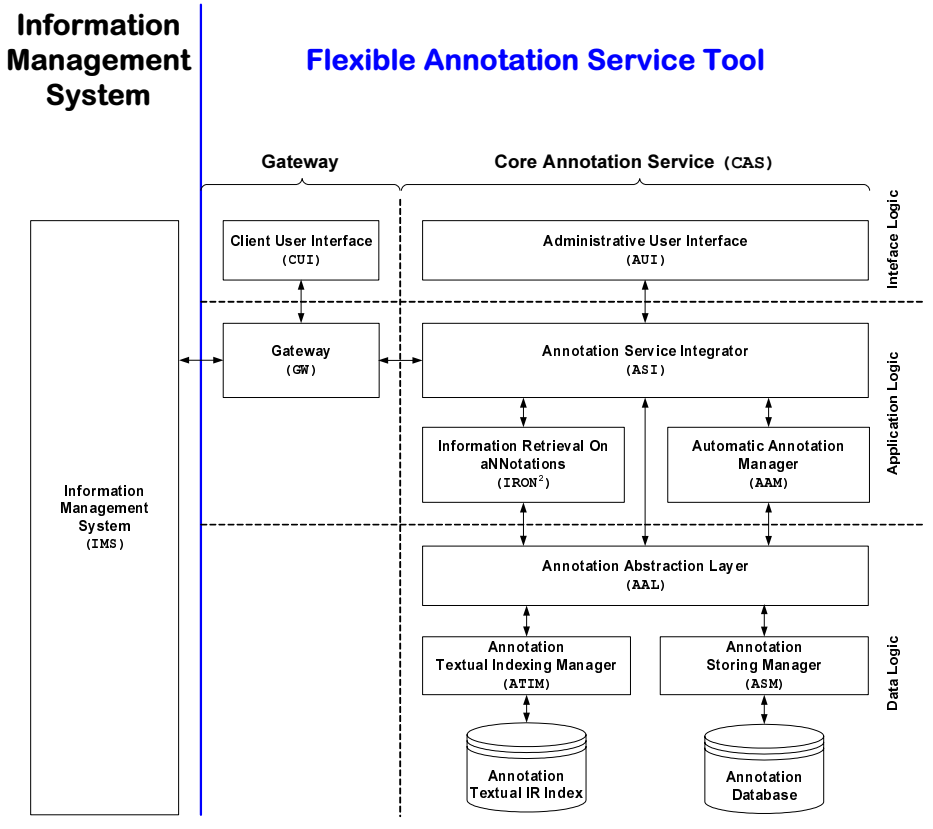

Fig. 2. Detailed architecture of the FAST system

LSs are a bunch of keywords extracted from a Web document that are used as query for a *Search Engine (SE)*. In this way, if a Web document cannot be found by means of its URL, then the LS of the document can be submitted to a SE in order to search and locate the document anyway. In conclusion, LSs represent an interesting alternative with respect to various kinds of identifiers and handles, due to the fact that LSs offer the possibility to almost uniquely identify a Web document by exploiting its own content.

4 FAST Conceptual Architecture

Figure 2 demonstrates the complete conceptual architecture of FAST, where FAST is depicted on the right, and the generic IMS is represented on the left. On the whole, the architecture is organized along two dimensions:

– *horizontal decomposition* (from left to right): consists of the IMS, the gateway and the CAS. It separates the core functionalities of FAST from the problem of integrating FAST into a specific IMS.

The horizontal decomposition allows us to accomplish the first two requirements of our architecture, since FAST is a stand-alone system that can be integrated with different IMSs by changing the gateway;

– *vertical decomposition* (from bottom to top): consists of three layers – the data, application and interface logic layers – and it is concerned with the organization structure of the CAS.
This decomposition allows us to achieve a better modularity within FAST and to properly describe the behaviour of FAST by means of isolating specific functionalities at the proper layer. Moreover, this decomposition makes it possibile to clearly define the functioning of FAST by means of communication paths that connect the different components of FAST itself. In this way, the behaviour of the FAST system is designed in a modular and extensible way.

The conceptual architecture of FAST is designed at a high level of abstraction in terms of abstract *Application Program Interfaces (APIs)* using an *Object Oriented (OO)* approach. 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, such as a WS or a P2P architecture. Furthermore, we apply the *abstract factory* design pattern [30], which uses a factory class that provides concrete implementations of a component, compliant with its interface, in order to guarantee a consistent way of managing the different implementations of each component. Java is the programming language in use for developing FAST. Java ensures us great portability across different hardware and software platforms, thus providing us with a further level of flexibility.

In the following sections we describe each component of FAST, according to figure 2, from bottom to top.

5 Data Logic Layer

5.1 Annotation Storing Manager

The *Annotation Storing Manager (ASM)* 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.

The ASM relies on a *Relational DBMS (RDBMS)* in order to store annotations. The database schema is given by the mapping to the relational data model of the *Entity–Relationship (ER)* schema for modelling annotations, which has been proposed in [10]. Thus, the ASM provides a set of basic operations for storing, retrieving, updating, deleting and searching annotations in a SQL–like fashion. Furthermore, it takes care of mapping the objects which represent the annotations into their equivalent representation in

the relational model, according to the *Data Access Object (DAO)*[11] and the *Transfer Object (TO)*[11] design patterns.

The DAO implements the access mechanism required to work with the underlying data source, i.e. it offers access to the RDBMS using the *Java DataBase Connectivity (JDBC)* technology. The components that rely on the DAO are called *clients* and they use the interface exposed by the DAO, which completely hides the data source implementation details from its clients. Because the interface exposed by the DAO to clients does not change when the underlying data source implementation changes, this pattern allows the DAO to adapt to different storage schemes without affecting its clients. Essentially, the DAO acts as an adapter between the clients and the data source. The DAO makes use of TOs as data carriers in order to return data to the client. The DAO may also receive data from the client in a TO in order to update the data in the underlying data source.

In conclusion, all of the other components of FAST deal only with objects representing annotations, which are the TOs of our system, without worrying about the details related to the persistence of such objects.

5.2 Annotation Textual Indexing Manager

The *Annotation Textual Indexing Manager (ATIM)* provides a set of basic operations for indexing and searching annotations for *Information Retrieval (IR)* purposes.

The ATIM is a full-text *Information Retrieval System (IRS)* and deals with the textual content of an annotation. It is based on the experience acquired in developing *Information Retrieval ON (IRON)*, the prototype IRS which has been used for participating in the *Cross-Language Evaluation Forum (CLEF)*[12] evaluation campaigns since 2002 [31,32,33,34]. CLEF is an international evaluation initiative aimed at providing an infrastructure for evaluating IRSs in a multilingual context.

Figure 3 shows the architecture of the last version of IRON, which has been used during the CLEF 2004 evaluation campaign [34]. Please note that in figure 3 the Logging component has been duplicated to make the figure more easily legible, but there actually is only one Logging component in the system.

IRON is a Java multi-threaded program, which provides textual IR functionalities and enables concurrent indexing and searching of document collections for both monolingual and bilingual tasks. IRON is made up of the following components:

- **Lexer:** implements an efficient lexer using JFlex 1.4[13], a lexer generator for Java. The current lexer is able to process any multilingual CLEF collection in a transparent way with respect to the document structure and to different character encodings, such as ISO 8859-1 or UNICODE[14].
- **IR engine:** is built on top of the Lucene 1.4 RC4[15] library, which is a high-performance text search engine library written entirely in Java. Lucene implements

[11] `http://java.sun.com/blueprints/corej2eepatterns/Patterns/`
[12] `http://clef.isti.cnr.it/`
[13] `http://www.jflex.de/`
[14] The lexer has been designed and developed by G. M. Di Nunzio [33,34].
[15] `http://jakarta.apache.org/lucene/docs/index.html`

the vector space model, and a $(\mathrm{tf} \times \mathrm{idf})$–based weighting scheme [35]. Some parts of the Lucene library were completely rewritten, i.e. a set of parallel classes has been written without modifying the original source code of Lucene, so that IRON remain compatible with the official Jakarta distribution of Lucene. In particular, those parts of Lucene concerning the text processing, such as tokenization, elimination of stop words, stemming, and the query construction, have been modified. Furthermore Lucene has been adapted to the logging infrastructure of IRON;

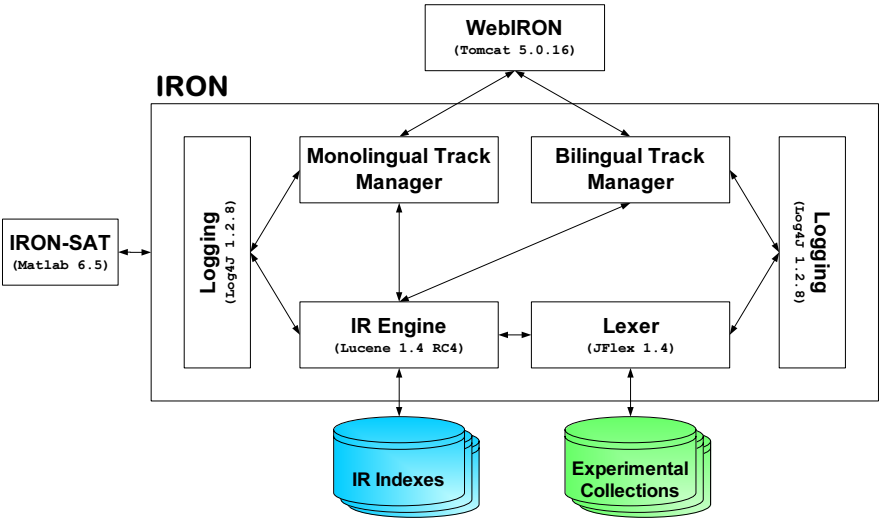

Fig. 3. Architecture of IRON

- **Monolingual Track Manager:** drives the underlying IR engine and provides high-level indexing and searching functionalities in order to carry out monolingual tasks. It provides a high-level API that allows us to easily plug together the different components of an IRS. This API can be further used to create a front-end application to IRON: for example we can develop a command-line application, a *Graphical User Interface (GUI)* for a stand-alone application, or a Web based *User Interface (UI)* to IRON;
- **Bilingual Track Manager:** drives the underlying IR engine and provides high-level indexing and searching functionalities in order to carry out the bilingual tasks. As the Monolingual Track Manager, also the Bilingual Track Manager provides a high-level API that can be used to develop different kinds of UIs for IRON;
- **Logging:** provides a full-fledged logging infrastructure, based on the Log4J 1.2.8[16] Java library. Each other component of IRON sends information about its status to the logging infrastructure, thus allowing us to track each step of the experiment.

[16] `http://logging.apache.org/log4j/docs/`

IRON is partnered with two other tools:

- **WebIRON:** is a Java servlet based Web interface. WebIRON is based on the Tomcat 5.0.16[17] Web server, making IRON a Web application. It provides a set of wizards which help the user to set all the parameters and choose the IR components, which are needed in order to conduct a run or, more generally, an IR experiment.
- ***IRON - Statistical Analysis Tool (IRON-SAT):*** is a Matlab program that interacts with IRON in order to carry out the statistical analysis of the experimental results. IRON-SAT parses the experimental data and stores the parsed information into a data structure suitable for the following processing. It is designed in a modular way, so that new statistical tests can be easily added to the existing code. The statistical analysis is performed using the Statistics Toolbox 4.0 provided by Matlab 6.5[18].

5.3 Annotation Abstraction Layer

The *Annotation Abstraction Layer (AAL)* abstracts the upper layers from the details of the actual storage and indexing of annotations, providing uniform access to the functionalities of the ASM and the ATIM.

The AAL provides the typical *Create–Read–Update–Delete (CRUD)* data management operations, coordinating the work of the ASM and the ATIM together. For example, when we create a new annotation, we need to put it into both the ASM and the ATIM.

Furthermore, the AAL provides search capabilities by properly forwarding the queries to the ASM or to the ATIM. Our modular architecture allows us to partner the ATIM, which is specialised for providing full text search capabilities, with other IRSs, which are specialised for indexing and searching other kinds of media. In any case, the addition of other specialised IRSs is transparent for the upper layers, due to the fact that the AAL provides the upper layers with an uniform access to those IRSs.

Note that both the ASM and the ATIM are focused on each single annotation in order to properly store and index it. On the other hand, both the ASM and the ATIM do not have a comprehensive view of the relationships that exist between documents and annotations. On the contrary, the AAL has a global knowledge of the annotations and their relationships by using the hypertext existing between documents and annotations. For example, if we delete an annotation that is part of a thread of annotations, what policy do we need to apply? Do we delete all the annotations that refer to the deleted one or do we try to reposition those annotations? The ASM and the ATIM alone would not be able to answer this question but, on the other hand, the AAL can drive the ASM and the ATIM to perform the correct operations by exploiting the hypertext between documents and annotations.

In conclusion, the AAL, the ASM and the ATIM constitute an IMS specialised in managing annotations, as a DBMS is specialised in managing structured data.

[17] `http://jakarta.apache.org/tomcat/index.html`
[18] `http://www.mathworks.com/`

6 Application Logic Layer

6.1 Automatic Annotation Manager

The *Automatic Annotation Manager (AAM)* automatically creates annotations for a given document. Automatic annotations can be created by using topic detection techniques in order to associate each annotation with its related topic, which constitutes the context of the annotation. In this way, a document can be re-organized and segmented into topics, whose dimension can range in many different sizes, and annotations can present a brief description of those topics.

6.2 Information Retrieval On aNNotations

Annotations introduce a new content layer aimed at elucidating the meaning of underlying documents, so that annotations can make hidden facets of the annotated documents more explicit. Thus, we can image to exploit annotations for retrieval purposes in order to satisfy better the user's information needs.

We need to develop a search strategy which is able to effectively take into account the multiple sources of evidence coming from both documents and annotations. Indeed, the combining of these multiple sources of evidence can be exploited in order to improve the performances of an information management system. We aim to retrieve more relevant documents and to rank them better than the case of a query without using annotations.

The *Unified Modeling Language (UML)* sequence diagram of figure 4 shows how searching for documents by exploiting annotations involves many components of FAST. Remember that we aim at combining the source of evidence which comes from annotations, managed by FAST, with the source of evidence which comes from documents, managed by the IMS. Thus, the search strategy requires the cooperation of both FAST and the IMS in order to acquire these two sources of evidence. Firstly, FAST receives a query from the end-user, which is dispatched from the user interface to *Information Retrieval On aNNotations (IRON^2)*. Secondly, the query is used to select all the relevant annotations, that is IRON^2 asks the *Annotation Service Integrator (ASI)* to find all the relevant annotations. Then, the hypertext between documents and annotations can be built and used to identify the documents that are related to the found annotations. Now we aim to combine the source of evidence which comes from the documents identified by the annotations with the one which comes from the documents managed by the IMS, as previously explained. Since the source of evidence concerning the documents is completely managed by the IMS, the FAST system has to query the IMS, which gives us back a list of relevant documents. Finally, once the FAST system has acquired this information from the IMS, it can combine this information with the source of evidence which comes from the documents identified by annotations in order to create a list of fused result documents that are presented to the users.

The reader, which is interested in knowing how FAST exploits annotations in order to search and retrieve relevant documents to answer to a user query, can refer to [36] where it is provided a formal framework which is used in facing the searching for documents from digital collections of annotated documents.

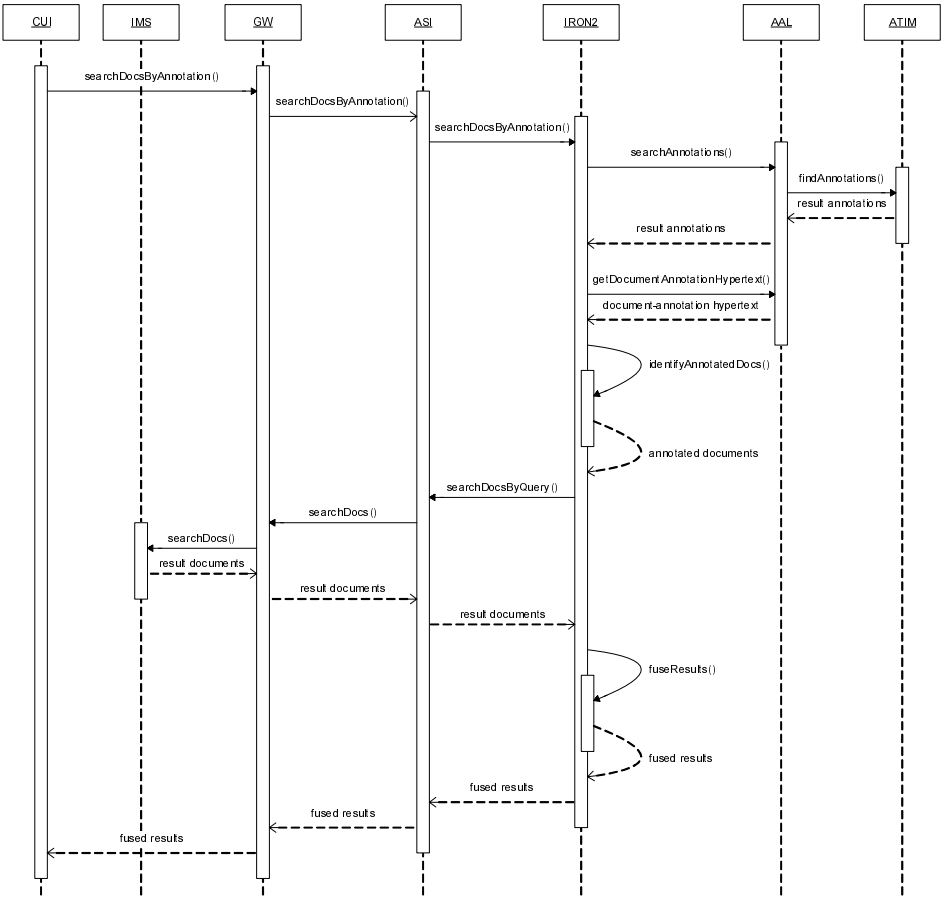

Fig. 4. Sequence diagram for searching documents by exploiting annotations

6.3 Annotation Service Integrator

The ASI integrates the underlying components and provides uniform access to them. It represents the entry point to the CAS for both the gateway and the user interface, dispatching their requests to underlying layers and then collecting the responses from the underlying layers.

The UML sequence diagram of figure 4 shows how the ASI plays a central role in coordinating the different components of FAST. In the example of figure 4, the ASI forwards the user query to IRON^2; it dispatches the request for relevant documents of IRON^2 to the *Gateway (GW)* in order to submit this query to the IMS; then, it passes the results provided by the IMS back to IRON^2; finally, it gives the fused result list produced by IRON^2 back to the GW in order to return this list to the user interface.

6.4 Gateway

As already discussed in Section 3, the GW provides functionalities of mediator between the CAS and the IMS. By changing the gateway, we can share FAST with different IMSs. In this way, we can provide a wide range of different architectural choices: firstly, the CAS could be connected to a specific IMS which uses proprietary protocols and data structures and, in this case, the gateway can implement them, as we did in the case of the OpenDLib[19] digital library [10]; secondly, we could employ WS to carry out the gateway, so that FAST is accessible in a more standardized way; finally, the gateway could be used to adapt FAST to a P2P network of IMSs.

7 Interface Logic Layer

7.1 Administrative User Interface

The *Administrative User Interface (AUI)* is a Web-based UI for the administration of FAST. It provides the different functionalities needed to configure and run FAST, such as the choice of the gateway to be used, the creation and management of the users granted by the system, and so on.

7.2 Client User Interface

The *Client User Interface (CUI)* provides end–users with an interface for creating, modifying, deleting and searching annotations.

The CUI is connected to, or even directly integrated into, the gateway, so that it represents a user interface tailored to the specific IMS for which the gateway is developed. In this way, the gateway forwards the requests from the CUI to the ASI, as it is shown in the example of figure 4.

8 Conclusions and Future Work

This paper discussed the conceptual architecture of the FAST system, which separates core functionalities from their integration in any particular IMS. In this way, FAST acts as a bridge between different IMSs and allows the hypertext to cross the boundaries of a single IMS, in order to exploit annotations as an active and effective collaboration tool for users.

We plan to enhance our conceptual architecture in order to support a network of P2P FAST systems. In this way, we will be able to implement FAST not only as a stand–alone system, that can be integrated into different IMSs, but also as a P2P network of FASTs that cooperate in order to provide advanced annotation functionalities to different IMSs.

[19] `http://www.opendlib.com/`

Acknowledgements

This work is partially funded by the ECD project, which is a joint program between the Italian National Research Council (CNR) and the Ministry of Education (MIUR), with regards to law 449/97-99. The work is also 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).

References

1. Agosti, M., Ferro, N.: Chapter X: Managing the Interactions between Handheld Devices, Mobile Applications, and Users. In Lim, E.P., Siau, K., eds.: Advances in Mobile Commerce Technologies, Idea Group, Hershey, USA (2003) 204–233
2. Nagao, K.: Digital Content Annotation and Transcoding. Artech House, Norwood (MA), USA (2003)
3. Rigaux, P., Spyratos, N.: Metadata Inference for Document Retrieval in a Distributed Repository. In Maher, M.J., ed.: Proc. 9th Asian Computing Science Conference – Advances in Computer Science (ASIAN 2004) – Higher Decision Making. Dedicated to Jean-Louis Lassez on the Occasion of His 5th Cycle Birthday, Lecture Notes in Computer Science (LNCS) 3321, Springer, Heidelberg, Germany (2004) 418–436
4. Gueye, B., Rigaux, P., Spyratos, N.: Taxonomy-Based Annotation of XML Documents: Application to eLearning Resources. In Vouros, G.A., Panayiotopoulos, T., eds.: Proc. 3rd Helenic Conference on AI – Methods and Applications of Artificial Intelligence (SETN 2004), Lecture Notes in Computer Science (LNCS) 3025, Springer, Heidelberg, Germany (2004) 33–42
5. Kahan, J., Koivunen, M.R.: Annotea: an open RDF infrastructure for shared Web annotations. In Shen, V.Y., Saito, N., Lyu, M.R., Zurko, M.E., eds.: Proc. 10th International Conference on World Wide Web (WWW 2001), ACM Press, New York, USA (2001) 623–632
6. Buneman, P., Khanna, S., Tajima, K., Tan, W.C.: Archiving Scientific Data. ACM Transactions on Database Systems (TODS) **29** (2004) 2–42
7. Buneman, P., Khanna, S., Tan, W.C.: Why and Where: A Characterization of Data Provenance. In Van den Bussche, J., Vianu, V., eds.: Proc. 8th International Conference on Database Theory (ICDT 2001), Lecture Notes in Computer Science (LNCS) 1973, Springer, Heidelberg, Germany (2001) 316–330
8. Buneman, P., Khanna, S., Tan, W.C.: On Propagation of Deletions and Annotations Through Views. In Abiteboul, S., Kolaitis, P.G., Popa, L., eds.: Proc. 21st ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems (PODS 2002), ACM Press, New York, USA (2002) 150–158
9. Bhagwat, D., Chiticariu, L., Tan, W.C., Vijayvargiya, G.: An Annotation Management System for Relational Databases. In Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B., eds.: Proc. 30th International Conference on Very Large Data Bases (VLDB 2004), Morgan Kaufmann (2004) 900–911
10. Agosti, M., Ferro, N.: Annotations: Enriching a Digital Library. In Koch, T., Sølvberg, I.T., eds.: Proc. 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2003), Lecture Notes in Computer Science (LNCS) 2769, Springer, Heidelberg, Germany (2003) 88–100

11. Frommholz, I., Brocks, H., Thiel, U., Neuhold, E., Iannone, L., Semeraro, G., Berardi, M., Ceci, M.: Document-Centered Collaboration for Scholars in the Humanities – The COLLATE System. In Koch, T., Sølvberg, I.T., eds.: Proc. 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2003), Lecture Notes in Computer Science (LNCS) 2769, Springer, Heidelberg, Germany (2003) 434–445
12. Thiel, U., Brocks, H., Frommholz, I., Dirsch-Weigand, A., Keiper, J., Stein, A., Neuhold, E.J.: COLLATE – A collaboratory supporting research on historic European films. International Journal on Digital Libraries **4** (2004) 8–12
13. Agosti, M., Benfante, L., Orio, N.: IPSA: A Digital Archive of Herbals to Support Scientific Research. In Sembok, T.M.T., Zaman, H.B., Chen, H., Urs, S.R., Myaeng, S.H., eds.: Proc. 6th International Conference on Asian Digital Libraries. Digital Libraries – Digital Libraries: Technology and Management of Indigenous Knowledge (ICADL 2003), Lecture Notes in Computer Science (LNCS) 2911, Springer, Heidelberg, Germany (2003) 253–264
14. Agosti, M., Ferro, N., Orio, N.: Annotations as a Support to Research Users. In Catarci, T., Christodoulakis, S., Del Bimbo, A., eds.: Proc. 7th International Workshop of the EU Network of Excellence DELOS on Audio-Visual Content and Information Visualization in Digital Libraries (AVIVDiLib'05), Centromedia, Viareggio, Italy (2005) 117–120
15. Agosti, M., Ferro, N., Orio, N.: Annotating Illuminated Manuscripts: an Effective Tool for Research and Education. In Proc. 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2005), ACM Press, New York, USA (2005) (in print)
16. Marshall, C.C.: Annotation: from Paper Books to the Digital Library. In Allen, R.B., Rasmussen, E., eds.: Proc. 2nd ACM International Conference on Digital Libraries (DL 1997), ACM Press, New York, USA (1997) 233–240
17. Marshall, C.C., Brush, A.J.B.: Exploring the Relationship between Personal and Public Annotations. In Chen, H., Wactlar, H., Chen, C.C., Lim, E.P., Christel, M., eds.: Proc. 4th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2004), ACM Press, New York, USA (2004) 349–357
18. Schilit, B.N., Price, M.N., Golovchinsky, G.: Digital Library Information Appliances. In Witten, I., Akscyn, R., Shipman, F.M., eds.: Proc. 3rd ACM International Conference on Digital Libraries (DL 1998), ACM Press, New York, USA (1998) 217–226
19. Marshall, C.C., Golovchinsky, G., Price, M.N.: Digital Libraries and Mobility. Communications of the ACM **44** (2001) 55–56
20. Marshall, C.C., Ruotolo, C.: Reading-in-the-Small: A Study of Reading on Small Form Factor Devices. In Hersh, W., Marchionini, G., eds.: Proc. 2nd ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2002), ACM Press, New York, USA (2002) 56–64
21. Bottoni, P., Civica, R., Levialdi, S., Orso, L., Panizzi, E., Trinchese, R.: MADCOW: a Multimedia Digital Annotation System. In Costabile, M.F., ed.: Proc. Working Conference on Advanced Visual Interfaces (AVI 2004), ACM Press, New York, USA (2004) 55–62
22. Bottoni, P., Civica, R., Levialdi, S., Orso, L., Panizzi, E., Trinchese, R.: Storing and Retrieving Multimedia Web Notes. In Bhalla, S., ed.: Proc. 4th International Workshop on Databases in Networked Information Systems (DNIS 2005), Lecture Notes in Computer Science (LNCS) 3433, Springer, Heidelberg, Germany (2005) 119–137
23. Agosti, M.: An Overview of Hypertext. In Agosti, M., Smeaton, A., eds.: Information Retrieval and Hypertext. Kluwer Academic Publishers, Norwell (MA), USA (1996) 27–47
24. Marshall, C.C.: Toward an Ecology of Hypertext Annotation. In Akscyn, R., ed.: Proc. 9th ACM Conference on Hypertext and Hypermedia (HT 1998): links, objects, time and space-structure in hypermedia systems, ACM Press, New York, USA (1998) 40–49
25. Agosti, M., Ferro, N., Frommholz, I., Thiel, U.: Annotations in Digital Libraries and Collaboratories – Facets, Models and Usage. In Heery, R., Lyon, L., eds.: Proc. 8th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2004), Lecture Notes in Computer Science (LNCS) 3232, Springer, Heidelberg, Germany (2004) 244–255

26. Frommholz, I., Thiel, U., Kamps, T.: Annotation-based Document Retrieval with Four-Valued Probabilistic Datalog. In Baeza-Yates, R., Maarek, Y., Roelleke, T., de Vries, A.P., eds.: Proc. 3rd XML and Information Retrieval Workshop and the 1st Workshop on the Integration of Information Retrieval and Databases (WIRD2004), http://homepages.cwi.nl/~arjen/wird04/wird04-proceedings.pdf [last visited 2004, November 22] (2004) 31–38
27. Fogli, D., Fresta, G., Mussio, P.: On Electronic Annotation and Its Implementation. In Costabile, M.F., ed.: Proc. Working Conference on Advanced Visual Interfaces (AVI 2004), ACM Press, New York, USA (2004) 98–102
28. Berners-Lee, T., Fielding, R., Irvine, U.C., Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396 (1998)
29. Park, S.T., Pennock, D., Giles, C.L., Krovetz, R.: Analysis of Lexical Signatures for Improving Information Persistence on the World Wide Web. ACM Transactions on Information Systems (TOIS) **22** (2004) 540–572
30. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object–Oriented Software. Addison-Wesley, Reading (MA), USA (1995)
31. Agosti, M., Bacchin, M., Ferro, N., Melucci, M.: Improving the Automatic Retrieval of Text Documents. In Peters, C., Braschler, M., Gonzalo, J., Kluck, M., eds.: Advances in Cross-Language Information Retrieval, Third Workshop of the Cross–Language Evaluation Forum (CLEF 2002) Revised Papers, Lecture Notes in Computer Science (LNCS) 2785, Springer, Heidelberg, Germany (2003) 279–290
32. Bacchin, M., Ferro, N., Melucci, M.: A Probabilistic Model for Stemmer Generation. Information Processing & Management **41** (2005) 121–137
33. Di Nunzio, G.M., Ferro, N., Melucci, M., Orio, N.: Experiments to Evaluate Probabilistic Models for Automatic Stemmer Generation and Query Word Translation. In Peters, C., Braschler, M., Gonzalo, J., Kluck, M., eds.: Comparative Evaluation of Multilingual Information Access Systems: Fourth Workshop of the Cross–Language Evaluation Forum (CLEF 2003) Revised Selected Papers, Lecture Notes in Computer Science (LNCS) 3237, Springer, Heidelberg, Germany (2004) 220–235
34. Di Nunzio, G.M., Ferro, N., Orio, N.: Experiments on Statistical Approaches to Compensate for Limited Linguistic Resources. In Peters, C., Clough, P., Gonzalo, J., Jones, G., Kluck, M., Magnini, B., eds.: Fifth Workshop of the Cross–Language Evaluation Forum (CLEF 2004) Revised Selected Papers, Lecture Notes in Computer Science (LNCS), Springer, Heidelberg, Germany (in print) (2005)
35. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, New York, USA (1983)
36. Agosti, M., Ferro, N.: Annotations as Context for Searching Documents. In Crestani, F., Ruthven, I., eds.: Proc. 5th International Conference on Conceptions of Library and Information Science - Context: nature, impact and role, Lecture Notes in Computer Science (LNCS) 3507, Springer, Heidelberg, Germany (2005) 155–170