An Information Service Architecture for Annotations

Maristella Agosti and Nicola Ferro

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

Abstract. This paper presents the architecture and the features of an annotation service for digital libraries. Firstly, it describes the relevant aspects of annotations and the different way of using them. Then it shows how the characteristics of the annotations drive the design choices of the architecture. Finally it discusses the design choices and the architectural alternatives for developing the annotation service.

1 Introduction

The main objective of this research is to create an Annotation Service (AS) for Digital Libraries (DL), which deals with many aspects of annotations, such as creation, management, access and retrieval of both manual and automatically created annotations. The service at hand is innovative because in fact it is able to actively exploit annotations in order to select and retrieve documents from a digital library in response to a user query.

Over past years a lot of research work regarding annotations has been done [18]: user studies for understanding annotation practices and discovering common annotation patterns [12–14, 22]; investigation and categorization of various facets of the annotation [3, 8, 13]; employment of ad-hoc devices or handheld devices which enable reading appliances with annotation capabilities [15–17, 21]; design and development of document models and applications which support annotations in digital libraries, in the Web, in collaboratory systems and working groups [3, 8, 10, 18, 20, 23, 24]. All this research work has led to different viewpoints about what annotations are:

- *annotations are metadata*: they are additional data about an existing content. For example, the World Wide Web Consortium (W3C) [10, 23, 24] 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;
- *annotations are contents*: they are additional content about an existing content; they increase existing content and 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 [1]. For example, [13] considers annotations as a natural way of enhancing hypertexts by actively engaging users with existing content in a digital library [12, 15, 21];
- *annotations are dialog acts*: they are part of a discourse with an existing content. For example, [8] considers annotations as the document context, intended as the context of the collaborative discourse in which the document is placed.

In the following for annotation we mean *any piece of additional content associated with an existing content* and we consider that annotations and existing contents constitute a hypertext. Note that digital libraries do not usually provide a hypertext that links documents together; thus annotations can represent an effective means for creating a hypertext that links annotations and existing content together. This hypertext can be exploited not only for providing alternative navigation and browsing capabilities, but also for offering advanced search functionalities. Finally the hypertext existing between documents and annotations enables different annotation configurations, that are *threads of annotations*, i.e. an annotation made in response or comment to another annotation, and *sets of annotation*, i.e. a bundle of annotations on the same passage of text. A comprehensive presentation on annotations and their facets can be found in [3, 4].

Furthermore annotations introduce a new content layer devoted to elucidate the meaning of an underlying information resource and they can make hidden facets of the annotated information resource more explicit. Thus, we can exploit annotations for retrieval purposes, and add the evidence provided by annotations themselves in order to better satisfy the user's information needs. For example, suppose that we have the following query: "good survey grid computing". A relevant document for this query could be ranked low because it does not contain the terms "good" and "survey". On the other hand, if the following annotation "good survey, which clearly explains the topic" is linked to the document, we can exploit it in order to better rank the document. Note that the annotation itself does not clearly states what topic is explained but this information can be still obtained exploiting the hypertext and navigating the link that connects the annotation to the document. Thus we can combine two distinct sources of evidence, the one coming from the document and the one coming from the annotation. This way the annotation and the document can cooperate together in order to better satisfy the user's information need. Indeed the combining of these multiple sources of evidence can be exploited to improve the performances of an information management system, i.e. to retrieve more relevant documents and to better rank them with respect to the case of a simple query without annotations. This is what we mean for "to actively exploit annotations in order to select and retrieve documents from a DL in response to a user query".

Finally, today the notion of isolated applications or data is increasingly disappearing in favour of a distributed and networked environment with an information centric view. This allows us to provide integrated services and applications to users, without any distinction between local and remote information resources.

In this context we can envisage a scenario in which a DL can become not only a place where information resources can be stored and made available, but also a daily work tool, which can be integrated into the way the user works, so that the user's intellectual work and contents which are provided by the digital library can be merged together, constituting a single working context. Thus the digital library 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 used in enabling this paradigm of interaction between users and digital libraries for all the reasons previously explained and, in particular, because annotations allow users to naturally create an hypertext that seamlessly merges personal contents with the contents provided by the digital library.

The presentation of the findings is structured as follows: Section 2 introduces the design choices and the features of the architecture of the annotation service, while Section 3 describes each component of the annotation service and its functionalities.

2 Annotation Service Architecture

In the context introduced in Section 1, architectural choices become a key factor for enabling the design and development of an advanced annotation service capable of both modelling the different facets of the annotation and effectively exploiting annotations for search and retrieval purposes.

We need to design an architecture with a twofold aim: firstly, it has to model the behaviour of the AS in a modular way, so that we can easily add new functionalities to the AS without the need of redesigning the architecture of the AS. Secondly, the architecture has to be flexible enough to be implemented according to different architectural paradigms, such as Web Services (WS) or Peer–to–Peer (P2P) architectures (figure 1). Indeed a flexible architecture allows the AS to have a great reach and a widespread usage, so that users can benefit from its functionalities without limitations due to the architecture of a particular DL, allowing a strict interaction between users and digital libraries.

Thus the annotation service has to comply with the following constraint: it must be able to work with diverse DL systems.

We introduce this constraint for three reasons:

- annotations are a key technology for actively engaging users with a DL. Users need to have an annotation service easily available for each DL they work with and they should not change their annotative practices only because they work with a different DL system. Thus the annotation service must be capable to be integrated into various DL systems in order to guarantee a uniform way of interaction to the users;
- annotations create a hypertext that allow users to merge their personal content with the content provided by diverse DL systems, according to the scenario envisage above. This hypertext can span and cross the boundaries

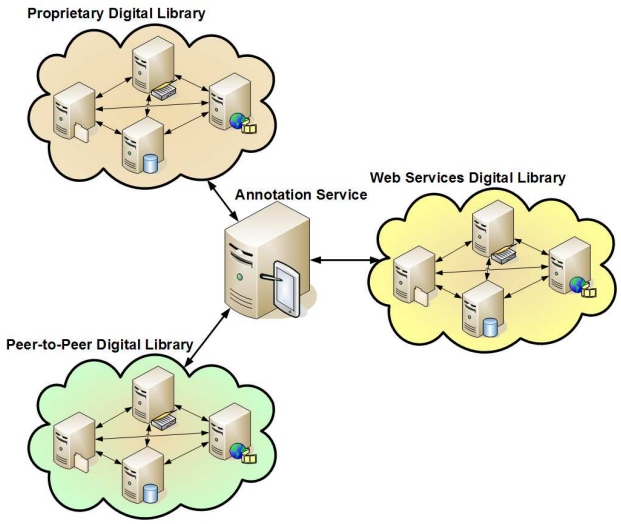

Fig. 1. Overview of the annotation service architecture.

of a single DL and thus the annotations service must be able to interact with diverse DL systems;
- annotations and information resources managed by the DL provide multiple sources of evidence for answering to a user's query. Since these sources of evidence can come from different DL systems and we need to combine them in order to answer the query, the annotation service must be able to work with diverse DL systems.

To comply with the constraint just defined, we have decided to adopt an architecture using a gateway which mediates between the DL and the core functionalities of the AS. Furthermore we have mapped this choice into a three layer architecture, which allows us to have a better modularity. Figure 2 demonstrates this architecture, where the AS is depicted on the right, and in the cloud the external DL system is represented.

The architecture is organized along two dimensions:

- *vertical decomposition* (from bottom to top): consists of three layers – the data, application and interface logic layers – and it is concerned with the organization of the core functionalities of the AS.
  This decomposition allows us to achieve a better modularity within the AS and to properly describe the behaviour of the AS by means of isolating specific functionalities at the proper layer. Moreover it makes it possibile to clearly define the workflow within the AS by means of communication paths that connect the different components of the AS itself. This way we can achieve the first aim of our architecture, that is to model the behaviour of the AS in a modular way.

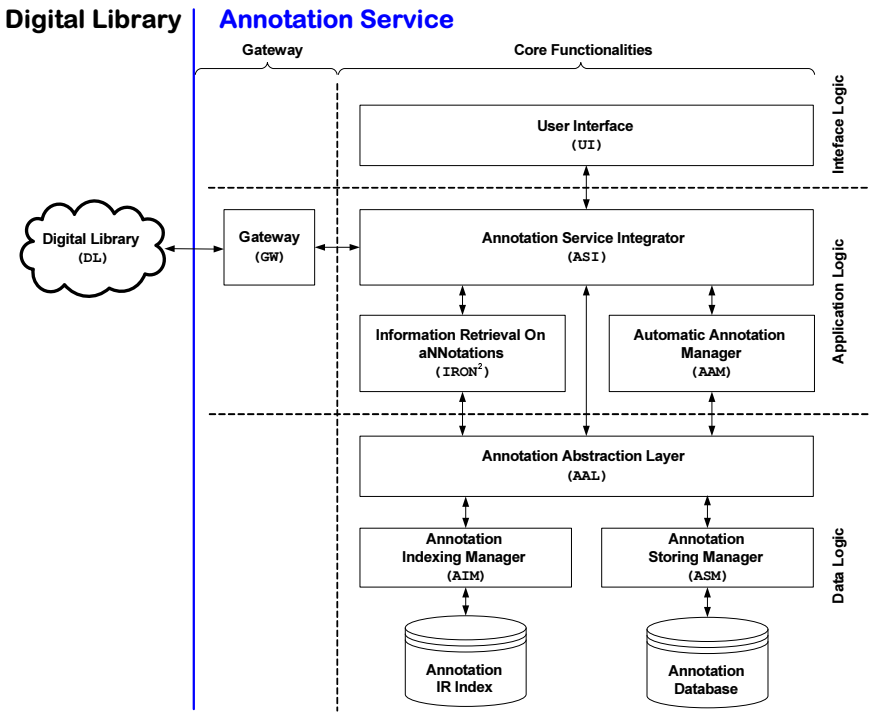

Fig. 2. Annotation Service Architecture.

– *horizontal decomposition* (from left to right): consists of the DL system, the gateway and the core services of the AS. It separates the core functionalities of the AS from the problem of integrating the AS into a specific DL system. The horizontal decomposition allows us to achieve the second aim of our architecture, since we can integrate the AS with a different DL system simply by changing the gateway. Furthermore the architecture is presented in the context of a DL, but it is suitable for integration also with other systems that manage documents and offer information retrieval capabilities, such as a search engine or a collaboratory system [4].

In order to achieve a great flexibility, we design the architecture of the AS at a high level of abstraction, that is we define the functionalities of each component of the AS in terms of abstract Application Program Interface (API). This way we can model the behaviour and the workflow within the AS 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 AS.

On the whole, we have two levels of abstraction: the first level is the separation of the core features of the AS from the integration with the DL and the organization of the AS into different layers in order to clearly determine the

different functionalities of the AS; the second level is built on top of the first one and makes it possible to describe at a higher level of abstraction the behaviour of each component of the AS, separating its behaviour from the actual implementation of the AS. This way we can have multiple implementations of the AS all referring to the same architecture and API.

We achieve the abstraction levels described above by means of a set of interfaces, which define the behaviour of each component of the AS in abstract terms. Then, a set of abstract classes partially implement the interfaces in order to define the behaviour of each component. This way this behaviour becomes common to all of the implementations of that component. Finally, the actual implementation is left to the concrete classes, inherited from the abstract ones, that fit the AS into a given architecture, such as a WS or P2P architecture. Furthermore, we apply the *abstract factory* design pattern [9], which uses a factory class for providing concrete implementations of a component, compliant with its interface, in order to guarantee a consistent way of managing the different implementations of each component.

The AS is developed using the Java[1] programming language, which provides us great portability across different hardware and software platforms.

3 Components of the Annotation Service

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

3.1 Data Logic Layer

Annotation Storing Manager (ASM) manages the actual storage of the annotations and provides a persistence layer for storing the `Annotation` objects, that are used in the upper layers for representing annotations.

The ASM provides a set of basic operations for storing, retrieving, deleting and searching annotations in a SQL–like fashion. Furthermore it takes care of mapping `Annotation` objects into their equivalent representation for the actual storage, according to the Data Access Object (DAO)[2] and the Transfer Object (TO)[3] design patterns. This way all of the other components of the AS deal only with `Annotation` objects, which represent the TO of our system, without worrying about the details related to the persistence of such objects.

Annotations are modeled according to the Entity–Relationship (ER) schema of figure 3, which is described in detail in [4]. Briefly, the ER schema of figure 3 represents the fact that an Annotation must Annotate one and only one digital object, identified by its handle DOHandle, while it can Relate To one

[1] `http://java.sun.com`

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

[3] `http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html`

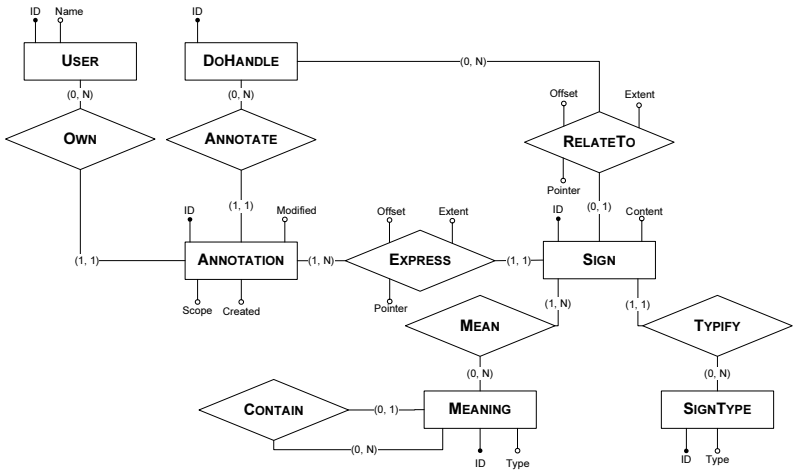

Fig. 3. Entity–Relationship schema for modelling annotations.

or more digital objects. Moreover one or more SIGNs, e.g. a piece of text or a graphic mark, contribute to EXPRESS an ANNOTATION and each SIGN can have one or more MEANINGs, that define its semantics.

Note that the ER schema of figure 3 can be easily mapped to different designing models, such as a relational schema, a Resource Description Framework (RDF) schema or a eXtensible Markup Language (XML) schema. This way it gives us great flexibility with respect to different architectural choices and allows us to provide different implementations of the ASM in a completely transparent manner for the other components of the AS.

Annotation Indexing Manager (AIM) provides a set of basic operations for indexing and searching annotations for information retrieval purposes.

The AIM is a full-text information retrieval system and deals with the textual content of an annotation. It is based on the experience acquired in developing IRON (Information Retrieval ON), the prototype information retrieval system that we have developed to be used in the Cross Language Evaluation Forum (CLEF) evaluation campaigns since 2002; the main functionalities of IRON are described in [2, 7].

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 ASI and the ASM.

The AAL provides the typical Create–Read–Update–Delete (CRUD) data management operations, taking care of coordinating the work of the ASM and the ASI together. For example when we create a new annotation, we need to put it in both the ASM and in the ASI or, when we delete an annotation, we need to remove it from both the ASM and the ASI.

Furthermore the AAL provides search capabilities forwarding the queries to the ASM or to the AIM. At the moment there is only the AIM for providing full text search capabilities but, in the future, other specialised information retrieval engines can be added to the system, for example for indexing and searching the graphical content of an annotation and other types of digital media. In any case the addition of other information retrieval engines becomes transparent for the upper layers, since the AAL provides uniform access to them.

Note that both the ASM and the AIM are focused on each single annotation in order to properly store and index it, but they do not have a comprehensive view of the relationships that exist between documents and annotations, that is they do not take into consideration the hypertext mentioned in Section 1. On the contrary, the AAL has a global view of the annotations and their relationships and exploits it for managing purposes. 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 AIM alone would not be able to answer this question but, on the other hand, the AAL can drive the ASM and the AIM to perform the correct operations.

Thus, on the whole, the AAL, the ASM and the AIM constitute an information management system specialised in managing annotations, as a database management system is specialised in managing structured data.

3.2 Application Logic Layer

Information Retrieval on aNNotations (IRON^2) provides advanced search capabilities based on annotations, such as those previously introduced.

As an important consequence of our architecture, we know everything about annotations but we have no knowledge about documents managed by the DL. This is due to the fact that we directly manage annotations while documents and information pertaining to them are provided by the DL.

This architectural choice influences the way in which our search strategy is carried out and retrieving documents by using annotations involves a complex strategy. Firstly, the AS receives a query from the end-user, the query is used to select all the relevant annotations, an annotation hypertext can now be built and used to identify the related documents. Now we aim to combine the source of evidence coming from annotations with the one coming for the documents managed by the DL, as previously explained. Since the source of evidence concerning the documents is completely managed by the DL, the AS has to query the DL, which gives back a list of relevant documents. Only after that the AS has acquired this information from the DL, it can combine it with the source of evidence coming from annotations in order to create a fused list of result documents that are presented to the users to better satisfy his information needs.

The Unified Modeling Language (UML) sequence diagram [19] of figure 4 shows how this search strategy involves many of the components of the AS.

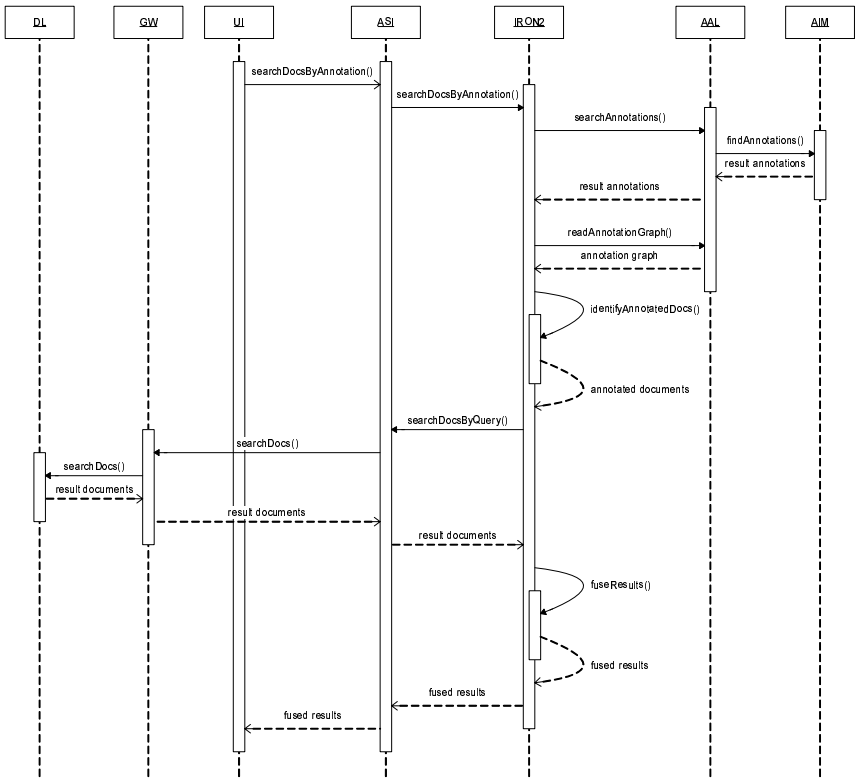

Fig. 4. Sequence diagram for searching documents exploiting annotations.

Automatic Annotation Manager (AAM) creates automatic annotations for a given document.

Automatic annotations can be created using topic detection techniques to associate each annotation with its related topic, which forms the context of the annotation. In this way a document can be re-organized and segmented into topics, whose granularity can vary, and annotations can present a brief description of those topics. Then by applying automatic hypertext construction techniques, similar to those presented in [5], document topics and annotations can be linked together, proposing an alternative way of navigating the content of a digital library.

Annotation Service Integrator (ASI) integrates the underlying components and provides uniform access to them. It represents the entry point to the core functionalities of the AS for both the gateway and the user interface, dispatching their requests to underlying layers and collecting the responses from the underlying layers.

The ASI can be further exploited for creating a network of P2P annotation services that cooperate together. In this scenario, when the ASI receives a request from the gateway or from the user interface – for example a search request – it forwards the request also to the other ASI peers and then collects their answers in order to provide access to the whole network of P2P annotation services.

Thus, our architecture allows us to implement the annotation service not only as a stand–alone service, that can be integrated into different DL systems, but also as a network of P2P annotation services that cooperate in order to provide advanced annotation functionalities to different DL systems.

Gateway (GW) provides functionalities of mediator between the core functionalities of the AS and the DL system. Simply by changing the gateway, we can share the same AS with different DL systems. We can envisage three kinds of gateway: firstly the AS could be connected to a DL which uses a proprietary protocol and in this case the gateway can implement it. This is the case, for example, of the OpenDLib digital library [6], with which the AS is going to cooperate [3]. Secondly we could employ Web Services to carry out the gateway, so that the AS is accessible in a more standardized way. Finally the gateway can be used to adapt the AS to a P2P digital library.

Note that the decoupling of the core functionalities of the AS from its integration within a specific DL is independent from the possibility of implementing the AS as a stand–alone service or as a network of P2P services. Indeed we can adapt both of these implementations to any kind of DL system by means of a proper gateway: in both cases the gateway represents the unique access point for the DL to the functionalities of the stand–alone AS or of the P2P network of AS.

3.3 Interface Logic Layer

User Interface (UI) provides an interface to end–users for creating, modifying, deleting and searching annotations. As show in figure 2 the UI is connected to the ASI, so that it represents the user interface of AS itself and it is independent of any particular DL system.

On the other hand, we can connect or integrate the UI directly in the gateway, so that it represents a user interface tailored to the specific DL for which the gateway is developed. In this case the gateway forwards the request of the UI to the ASI, for which the gateway acts also as user interface. For example, this choice has been adopted in integrating the AS into the OpenDLib digital library in order to obtain a user interface more coherent with those of the other OpenDLib services.

The design choice of connecting the UI to the gateway or to the ASI gives interesting possibilities for programmatically accessing the functionalities of the AS. As an example, if the gateway is implemented using Web Services, another service can programmatically access the AS in order to create more complex applications that exploit also annotations.

Acknowledgements

This work was partially funded by ECD (Enhanced Contents Delivery), a joined program between the Italian National Research Council (CNR) and the Ministry of Education (MIUR), under the law 449/97-99.

References

1. M. Agosti. An Overview of Hypertext. In M. Agosti and A. Smeaton, editors, *Information Retrieval and Hypertext*, pages 27–47. Kluwer Academic Publishers, Norwell (MA), USA, 1996.
2. M. Agosti, M. Bacchin, N. Ferro, and M. Melucci. Improving the Automatic Retrieval of Text Documents. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Evaluation of Cross–Language Information Retrieval Systems, Third Workshop of the Cross–Language Evaluation Forum, CLEF 2002, Revised Papers*, pages 279–290. Lecture Notes in Computer Science (LNCS) 2785, Springer, Heidelberg, Germany, 2003.
3. M. Agosti and N. Ferro. Annotations: Enriching a Digital Library. In Koch and Sølvberg [11], pages 88–100.
4. M. Agosti, N. Ferro, I. Frommholz, and U. Thiel. Annotations in Digital Libraries and Collaboratories – Facets, Models and Usage. In R. Heery and L. Lyon, editors, *Proc. 8th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2004)*. Lecture Notes in Computer Science (LNCS), Springer, Heidelberg, Germany (in print), 2004.
5. M. Agosti and M. Melucci. Information Retrieval Techniques for the Automatic Construction of Hypertext. In A. Kent and C.M. Hall, editors, *Encyclopedia of Library and Information Science*, volume 66, pages 139–172. Marcel Dekker, New York, USA, 2000.
6. D. Castelli and P. Pagano. OpenDLib: a Digital Library Service System. In M. Agosti and C. Thanos, editors, *Proc. 6th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2002)*, pages 292–308. Lecture Notes in Computer Science (LNCS) 2458, Springer, Heidelberg, Germany, 2002.
7. G. M. Di Nunzio, N. Ferro, M. Melucci, and N. Orio. Experiments to Evaluate Probabilistic Models for Automatic Stemmer Generation and Query Word Translation. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Evaluation of Cross–Language Information Retrieval Systems, Fourth Workshop of the Cross–Language Evaluation Forum, CLEF 2003, Revised Papers*. Lecture Notes in Computer Science (LNCS), Springer, Heidelberg, Germany (in print), 2004.
8. I. Frommholz, H. Brocks, U. Thiel, E. Neuhold, L. Iannone, G. Semeraro, M. Berardi, and M. Ceci. Document-Centered Collaboration for Scholars in the Humanities – The COLLATE System. In Koch and Sølvberg [11], pages 434–445.
9. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object–Oriented Software*. Addison-Wesley, Reading (MA), USA, 1995.
10. J. Kahan and M.-R. Koivunen. Annotea: an open RDF infrastructure for shared Web annotations. In V. Y. Shen, N. Saito, M. R. Lyu, and M. E. Zurko, editors, *Proc. 10th International Conference on World Wide Web (WWW 2001)*, pages 623–632. ACM Press, New York, USA, 2001.
11. T. Koch and I. T. Sølvberg, editors. *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.

12. C. C. Marshall. Annotation: from Paper Books to the Digital Library. In R. B. Allen and E. Rasmussen, editors, *Proc. 2nd ACM International Conference on Digital Libraries (DL 1997)*, pages 233–240. ACM Press, New York, USA, 1997.
13. C. C. Marshall. Toward an Ecology of Hypertext Annotation. In R. Akscyn, editor, *Proc. 9th ACM Conference on Hypertext and Hypermedia (HT 1998): links, objects, time and space-structure in hypermedia systems*, pages 40–49. ACM Press, New York, USA, 1998.
14. C. C. Marshall and A. J. B. Brush. From Personal to Shared Annotations. In L. Terveen and D. Wixon, editors, *Proc. Conference on Human Factors and Computing Systems (CHI 2002) – Extended Abstracts on Human Factors in Computer Systems*, pages 812–813. ACM Press, New York, USA, 2002.
15. C. C. Marshall, M. N. Price, G. Golovchinsky, and B.N. Schilit. Introducing a Digital Library Reading Appliance into a Reading Group. In N. Rowe and E. A. Fox, editors, *Proc. 4th ACM International Conference on Digital Libraries (DL 1999)*, pages 77–84. ACM Press, New York, USA, 1999.
16. C. C. Marshall, M. N. Price, G. Golovchinsky, and B.N. Schilit. Designing e-Books for Legal Research. In E. A. Fox and C. L. Borgman, editors, *Proc. 1st ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2001)*, pages 41–48. ACM Press, New York, USA, 2001.
17. C. C. Marshall and C. Ruotolo. Reading-in-the-Small: A Study of Reading on Small Form Factor Devices. In W. Hersh and G. Marchionini, editors, *Proc. 2nd ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2002)*, pages 56–64. ACM Press, New York, USA, 2002.
18. K. Nagao. *Digital Content Annotation and Transcoding.* Artech House, Norwood (MA), USA, 2003.
19. Object Management Group (OMG). OMG Unified Modeling Language Specification – March 2003, Version 1.5, formal/03-03-01. `http://www.omg.org/technology/documents/formal/uml.htm`, last visited 2004, May 25.
20. T. A. Phelps and R. Wilensky. Multivalent Annotations. In C. Peters and C. Thanos, editors, *Proc. 1st European Conference on Research and Advanced Technology for Digital Libraries (ECDL 1997)*, pages 287–303. Lecture Notes in Computer Science (LNCS) 1324, Springer, Heidelberg, Germany, 1997.
21. B. N. Schilit, M. N. Price, and G. Golovchinsky. Digital Library Information Appliances. In I. Witten, R. Akscyn, and F. M. Shipman, editors, *Proc. 3rd ACM International Conference on Digital Libraries (DL 1998)*, pages 217–226. ACM Press, New York, USA, 1998.
22. F. Shipman, M. N. Price, C. C. Marshall, and G. Golovchinsky. Identifying Useful Passages in Documents based on Annotation Patterns. In Koch and Sølvberg [11], pages 101–112.
23. World Wide Web Consortium (W3C). Annotea Project. `http://www.w3.org/2001/Annotea/`, last visited 2004, May 25.
24. World Wide Web Consortium (W3C). EMMA: Extensible MultiModal Annotation markup language – W3C Working Draft 18 December 2003. `http://www.w3.org/TR/2003/WD-emma-20031218/`, last visited 2004, May 25.