ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

Rainer Manthey and Viacheslav Wolfengagen (Eds)

Advances in Databases and Information Systems 1997

Proceedings of the First East-European Symposium on Advances in Databases and Information Systems, (ADBIS'97), St Petersburg, 2-5 September 1997

Access Control in an Object-Oriented Document Repository

Frank Buddrus

Published in collaboration with the
British Computer Society

©Copyright in this paper belongs to the author(s)

ISBN: 3-540-76227-2

Access Control in an Object-Oriented Document Repository

Frank Buddrus

Databases and Information Systems

Johann Wolfgang Goethe-University Frankfurt

Frankfurt am Main, Germany

Abstract

The approach presented in this paper aims at combining the advances of a sophisticated document repository with a flexible access control mechanism. It proposes an authorization model for hypertext documents stored in an object-oriented database system (ODBMS). The work is based on the idea of an ODBMS working as an Internet server for hypermedia documents. The approach introduced offers fine grained storage and retrieval of HTML documents, providing document maintenance, referential integrity, document integration, integration with application objects and access control. In contrast to comparable systems it acts transparently to the WWW users, i.e. without using proprietary query languages or protocols. While achieving the full benefits of a modern document repository it keeps the ease of the traditional file system it replaces. An access control mechanism establishes the support of different views on documents. This is favoured both by the need for security and the request for role dependent information presentation.

1 Introduction

Over 40 million users of more than 12 million machines are connected via the Internet (cf. [20, 34]). The Internet provides a number of services (e-mail, ftp, IRC, USENET, WWW, etc.) which are widely available and heavily used. Most of these services are aiming at distribution and exchange of information. The World Wide Web (WWW) became the most popular service and its use is rapidly growing. With just a few exceptions, WWW servers are run from standard file systems. Millions of documents, handled as normal files, build up the WWW. Almost no maintenance is provided for these documents, no consistency guaranteed, no document integration, and no real access control.

Documents available via WWW are typically either public or reside in a directory where access requires a special authentication. Moreover documents can only be retrieved as a whole. The reason for this inflexibility lies in the file systems that serve as repositories for all documents. Demands on sophisticated document maintenance go beyond the limits of file systems.

This work introduces a mechanism called **WOW** (**WWW** and **ODBMS** as a **Whole**) [8], that enhances the ODBMS O_2 [4] to a flexible WWW server offering high level access control. This is done in four steps:

1. Providing a mechanism for O_2 objects to produce HTML code (adopted from O_2 Web [25]).
2. Implementing a system with the facility to store and maintain (HTML) documents at variable granularity within O_2 .
3. Implementing a WWW server to handle requests for O_2 objects.
4. Adapting and enacting an authorization model for the HTML documents respecting their granularity in the repository.

WOW's main features are consistent documents, referential integrity (e.g. no unresolved links), reuse of document parts without copying (document integration), integration with (application) objects managed by O_2 , and a mechanism for providing access control and different views of the same document.

The remainder of this paper is organized as follows. In Section 2 the WWW and the way documents are treated are briefly introduced. Section 3 illustrates the WOW approach and some details of its architecture respectively. Section 4 finally describes the access control mechanism and the authorization model adapted. After a short summary of related work in Section 5 we conclude the paper with an outlook for future work.

2 World Wide Web

The WWW started at CERN (the European Laboratory for Particle Physics) and builds a 'distributed hypermedia system'. In practice, the WWW is a vast collection of interconnected documents, spanning the world via Internet.

The WWW is accessed through browsers which use the *HyperText Transport Protocol* (HTTP, [5]) to fetch documents from local and remote servers. Information providers set up WWW servers from which browsers can get WWW documents.

We distinguish two types of WWW documents: *bitstream* and *HTML documents*. A bitstream document contains a single piece of information which can e.g. be of one of the following types: image, animation, sound, application. HTML documents can contain bitstream documents and text and can furthermore refer to other WWW documents. Therefore, an HTML document represents both multimedia and hypermedia information. The hypermedia references are handled transparently by the browsers so that users can easily navigate through a web of interconnected documents, created by different authors and stored at different network sites. HTML documents must be specified in the *HyperText Markup Language* (HTML, [23, 26]) in order to be displayable for browsers. Each document is identified by a *Uniform Resource Locator* (URL).

Simple WWW documents are static, i.e. no interaction with WWW users is possible. To improve the situation, various mechanisms have been introduced: The *Common Gateway Interface* (CGI) is a standard for interfacing external applications from WWW servers. In contrast to CGI scripts where all computation is done by the server, the possibility of embedding real applications into HTML documents has been added recently by several approaches: Java [27, 18], JavaScript [32], MMM [28], Web* [2], etc.

3 WOW

Figure 1: The WOW architecture

In contrast to *O₂Web* and similar approaches for database access via Internet, WOW supplements *O₂* with the facility for integrating existing or creating new HTML documents and their embedded files (images, Java Applets, etc.). This is done by parsing HTML documents, transforming them into *O₂* objects, and storing them in an *O₂* database. A given HTML document will be dissected according to its syntactic structure (head/body/lists/tables etc.) and stored in appropriate objects connected to each other. These objects can then be accessed via an HTTP daemon (WOW-Server) without the usage of complex queries or protocols (see

Figure 1). As in O_2 Web, *o2report* methods and the possibility to query objects are provided to allow integration with application objects stored in O_2 .

The WOW Server (written in Java) redirects HTTP requests to the runtime-interface of the WOW-Shell. A request can either include a query (O_2 Web) or the name of a WWW document. The WOW-Shell identifies the corresponding object for the requested document, calls its unparse method (*o2report*) to recursively reconstruct the HTML code returned in order to serve the request. Requests concerning non-HTML documents are treated in the same way. The *o2report* method for those documents however will simply return the bitstream document belonging to it (e.g. images, animations, etc.).

Figure 2 shows the increments of an HTML document maintained under WOW.

Figure 2: An HTML document stored in WOW

3.1 WOW Characteristics

3.1.1 Transparency & Maintenance

There are several types of links within a typical HTML document. Links can either be local or global hypertext-references specified by the `<href>` tag or images, animation etc. specified by special tags like `<img src>`. While populating and running the WOW document repository it has to be kept track of all these links in order to provide high level consistency. WOW combines the ease of traditional HTML coding with the advantages of automatically gained consistency. While importing an existing document no link will be changed. WOW remembers file system type links and resolve them whenever needed. In this way the maintenance of documents stored is supported on different levels. This includes adding, removing and modifying documents as well as sharing document parts without copying.

3.1.2 Integration with Application Objects

The integration with application objects stored in O_2 is done by supporting the *o2report* method and query facility as introduced by O_2 Web. In this way any object can be mapped to a textual/graphical (HTTP compliant) description. This makes it possible both to retrieve an arbitrary number of objects and to refer to them from within any HTML document.

3.2 The WOW Architecture

WOW is highly flexible. The HTML parser, as well as the semantic machine (link maintenance, access control etc.), which together build main parts of the WOW Server, have been generated. In the same way (i.e. by generation), any other type of document, e.g. Modula-2 source code, can be integrated to WOW. The WOW generator input mainly consists of two files, one containing the syntax of the documents concerned and one the semantics for integrating documents with the WOW system (see Figure 3).

3.2.1 Document Syntax

WOW is able to store any (textual) document whose concrete syntax can be described by a special BNF (Backus Naur Form [21]). The level of granularity of the documents maintained by WOW is up to the granularity of the documents' BNF description. A WOW-BNF is a sequence of production rules each of which consists of a nonterminal, the production sign ('::='), and an expression of substitution. In order to get an expressive schema some restrictions have to be considered. A WOW BNF rule can either be:

- **Aggregation:** A sequence of terminals (enclosed by apostrophes) and nonterminals.

```
html_body ::= '<BODY>' html_list '</BODY>';
```

- **Generalization:** A sequence of nonterminals, separated by the pipe symbol.

```
formatting ::= heading | bold | italic | typed;
```

- **List:** One nonterminal representing the elements of the list and an optional separation symbol (enclosed in brackets).

```
html_list ::= html_element [];
```

- **Regular Expressions:** A sequence of symbols (enclosed by quotation marks) defining the appropriate regular expression.

```
identifier ::= "[a-zA-Z][a-zA-Z_0-9]*";
```

Figure 3: Generating a document schema

Here is an extract of the HTML-BNF used by the the WOW prototype (whitespaces like \n are inserted for parsing/unparsing purposes):

```
document ::= '<HTML>\n\n'
           html_head
           html_body
           '</HTML>\n\n';
```

```

html_body ::= '<BODY>\n' html_list '</BODY>\n';
...

// HTML-List

html_list ::= html [];
html ::= paragraph | horiz_line | break | complex;

complex ::= href | formating | image | enum |
           bordered_table | table | text | label;

formating ::= heading | bold | italic | typed |
            quote | preform | address;

heading ::= h1 | h2 | h3 | h4 | h5 | h6;
enum     ::= ulist | dlist | nlist;

// Simple Tags

paragraph ::= '<P>\n';
horiz_line ::= '<HR>\n';
break      ::= '<BR>\n';

// Special Tags

href ::= '<A HREF=\"' ref '\">'
        html_list '</A>\n';
label ::= '<A NAME=\"' lab_name '\">'
        html_list '</A>\n';
image ::= '<IMG SRC=\"' src '\">\n';
ref    ::= url | src_label | src;
url    ::= http | ftp;
src_label ::= pre_path '#' lab_name;
...

// Complex Tags - Lists

nlist  ::= '<OL>\n' li_list '</OL>\n';
li_list ::= list_item [];
...

// Complex Tags - Formating

italic ::= '<I>' html_list '</I>';
h1      ::= '<H1>' html_list '</H1>\n';
...

// Regular Expressions

src      ::= "[^\"]*";
pre_path ::= "[a-z_A-Z/.0-9-+]*";

```

```

identifier ::= "[a-zA-Z][a-zA-Z_0-9]*";
...

```

WOW generates an O_2 schema describing all increments of the language given by the BNF (one class for each nonterminal). Each increment gets a method *parse* to instantiate this increment from a given string (populating the database by providing already existing documents or document parts). Analogously for each increment an *o2report* method will be generated responsible for retrieving documents/increments from the database.

3.2.2 Semantics

After having defined the BNF of documents to be stored in O_2 some semantics have to be given in order to allow an integration with the WOW infrastructure. This is done by providing a set of control structures and methods that allow keeping track of any action concerning documents stored, e.g. adding, modifying, etc. The detailed description of WOW's semantic programming goes beyond the scope of this paper. However, the definition of an authorization model providing access control for documents maintained by WOW has been constructed within this framework.

4 Providing Access Control for WOW

WOW's ability to maintain any level of document granularity allows the realization of several advanced features demanded by modern document repositories. One of them is high level access control.

Current WWW technology is only capable of restricting access to complete files (usually directories). No other level of authorization is supported. Documents accessible can only be retrieved as a whole. This inflexibility mainly results from file systems serving as repositories. However the benefits of sophisticated access control can be many:

- Information protection: Depending on their identity users get different types of access to documents, i.e. document parts can be hidden and navigation/usage can be limited.
- Information presentation: Depending on the user's role information is differently viewed, e.g. managers get diagrams, whereas accountants get number columns.
- Information consistency: Concurrent editing of documents/document parts can be supported, e.g. by high level locking using dynamic authorizations.
- Information accounting: Based on user accounts access to information can be monitored and dynamically granted/denied.

In the last decade a lot of research has been done in the area of authorization models (cf. [9]) for database and operating systems. The area of hypertext/hypermedia systems has been relatively neglected. In [30] and [13] basic issues concerning authorization models for hypertext documents have been investigated. Nevertheless the restrictions resulting from file system limitations have not been resolved. The approach introduced in this paper combines the idea of an authorization model for hypertext documents with the possibility of a fine grained storage of documents offered by WOW. This is accomplished employing an access control mechanism for the ODBMS O_2 called SecOO [7] (see Figure 4). SecOO uses Prolog-like statements to describe authorization and implication rules and a trigger mechanism for controlling access to data stored in O_2 . By administrating rules and triggers different authorization models can be applied.

The following describes the authorization model adapted for WOW. It is derived from [30] using the terminology provided by [12]. Nevertheless other models (e.g. [19]) could also be instantiated within the same framework.

Figure 4: Access control mechanism in WOW

4.1 Authentication

HTTP provides a simple challenge-response authentication mechanism [6] which is used by the HTTP daemon to challenge a client request and by a client to provide authentication information. This mechanism is supported by most of the WWW browsers available. When requesting a file (usually directories) which need authentication information browsers typically ask the users to type in a *username* and a *password* to be returned to the HTTP daemon in order to check them against a master list of approved users. The WOW-Server forces this authentication for each non-public document requested. For subsequent requests to the same protected information (directories) browsers typically handle temporary lists of associations with server names and authentication information. As the user name and password are passed over the network as clear text this mechanism is not considered to be a secure method of user authentication. At least it provides user information for the purpose of access control. For more advanced authentication mechanisms refer to [16] and [31].

4.2 Authorization Rules

Authorization rules are given by an administrator as triples of the form:

```
access( subject, mode, set_of_objects )
```

This defines that a *subject* owns the *mode* of access for *set_of_objects*. If there is no corresponding authorization rule, access is denied by default.

Subjects are single users or groups which are identified during the authentication process. Groups¹ are regarded as sets of users and other groups. A user inherits all the rights of the groups he belongs to (Group Policy, see below). The most general group called *public* requires no password.

Since WOW allows variable grained document structures *Objects* of authorization could be any increments maintained². Regarding the complexity of administration though a reasonable limitation is strongly required. In this way authorization models should be considered as a trade-off between complexity and maintainability.

In our model objects of authorization are restricted to increments describing the logical structure of HTML documents. In contrast increments concerning the layout of documents, e.g. italic text or horizontal lines, are not regarded appropriate for authorization.

Increments considered by the WOW authorization are:

¹For an overview of user group structures refer to [14].

²[10] discusses security classifications for documents in detail.

- (complete) Documents (marked with <html>)
- Titles (<title>)
- Headers (<head>)
- Bodies (<body>)
- Hypertext-links ()
- (embedded) bitstream_documents ()
- Sections (which are areas marked with <a name>)
- Tables (<table>)
- Applications (e.g. <form> or <applet ...>)
- Lists (e.g. or)
- Footers (marked with <address>)

Authorization to a piece of information (increment) implies the same kind of authorization for all information embedded (Completeness Policy, see below), e.g the right to view a document implies the right to view all of its sections etc. without exception.

[30] distinguishes basic objects (analogous to WOW's embedded bitstream documents) and nodes (analogous to a generalization of all other increments listed above), whereas [13] suggests authorization objects directly derived from document object models.

Modes of access are developed from the activities that can be applied to HTML documents. Similar to [30] we distinguish authorization for *browsing*, *usage*, and *authoring* (see Figure 5). Browsing is related to any request for information, usage for embedding pieces of information to documents outside of WOW, and authoring for adding/updating documents/data to WOW. There are several activities related to WWW technology that can hardly be monitored, like printing, copying etc. Although [13] includes them to the modes (types) of access they are not considered meaningful in our model.

Figure 5: Modes of access in WOW

For browsing the mode of access can be one of *view*, *navigate*, and *run*, for usage either of *refer* and *share*, and for authoring either of *insert* and *modify*.

These modes have the following meaning:

- **Browsing:**

- **view:** The piece of information (increment) concerned can be accessed for viewing (only).
- **navigate:** The information can be accessed for viewing and following included hypertext-links. Note that this simply enables (marked) hypertext-links. It does not imply any authorization to the information referred to.
- **run:** The information can be accessed for viewing, navigating, and running both applications (Java, JavaScript, etc.) and Forms.

- **Usage:**

- **refer:** Information can be referred to via hypertext-links (possible for (bitstream) documents and sections). Currently this can only be monitored within the WOW framework. If public information is referred to without *refer* permission, access will be denied.
- **share:** Information can be shared in other documents. Note that this is a feature within the WOW framework and can only be applied to sections (see Figure 6).

- **Authoring:**

- **insert:** Information can be added. This is a feature available within the WOW framework.
- **modify:** Already existing information can be modified (and deleted). This as well is a feature only available within the WOW framework.

Figure 6: Sharing sections in WOW

4.3 Implication Rules/Authorization Policies

In order to keep the number of authorization rules as small as possible several implication rules have been defined that together build up the authorization policies for the model proposed. Implication rules in WOW are provided by Prolog-like statements. Policies are given as sets of implication rules belonging together. In the following the most important policies, which are already mentioned above, are explained.

4.3.1 Group Policy

In WOW users are identified by a name and a password. Groups are special users that can be considered as sets of other users. A single user can be member of several groups while cyclic membership relations are prohibited. WOW maintains a base of all existing groups and users, given by facts (rules without body) of the form *user(name,password)* and *member(user,group)*:

```
user('public','').
user('frank','knarf').
user('admins','snimda').
user('assistants','stnatsissa').
...
member('frank','assistants').
member('assisans','admins').
member(_, 'public').
```

Now users inherit all access rights of any groups they are (transitive) members of (the arrow <- represents the implication direction):

```
policy Group_Policy
  access(AUser,AMode,Objects) <-
    member(AUser,GroupX),
    access(GroupX,AMode,Objects).
end.
```

4.3.2 Completeness Policy

In order to decide whether an increment is a part of another increment WOW supports a predicate called *includes* which can be used defining policies. Here access to an increment implies the same kind of access to all increments included:

```
policy Completeness_Policy
  access(AUser,AMode,ObjectA) <-
    includes(ObjectB,ObjectA),
    access(AUser,AMode,ObjectB).
end.
```

4.3.3 Mode Domination Policy

Some modes of access imply others. The right to navigate an increment (follow hypertext-links) for instance should also include (imply) the right to view it. This is expressed by a simple policy:

```
policy Mode_Domination_Policy
  implies(run,navigate).
  implies(navigate,view).
  implies(modify,insert).

  access(AUser,ModeA,Objects) <-
    implies(ModeB,ModeA),
    access(AUser,ModeB,Objects).
end.
```

4.3.4 Collecting Objects

The identification of objects (increments) for defining authorization rules can be cumbersome in some cases, especially when addressing a large number of documents. For this purpose a mechanism for building object sets is provided. This is accomplished by supporting a number of primitives:

- *objects(struct)* for collecting all increments of a certain structure (similar to class/type extensions). *objects(table)* for instance identifies the set of all objects included by all tables maintained by WOW.
- *docs(dir)* for collecting all documents residing in a certain area (directory). WOW maintains a so called *table of contents* containing each document stored and a name for referencing and identifying it. This name is typically related to a path name and contains a sequence of '/'-separated directory names and a trailing filename. *docs* gets a name pattern as input allowing wildcards (*, ?) and returns all documents having names matching this pattern. *docs('/dira/dirb/*')* for instance identifies all documents residing (having names) in the subdirectory '/dira/dirb'.
- set operations:
 - *diff(setA,setB)* for building the difference between two object sets
 - *union(setA,setB)* for building the union of two object sets
 - *intersect(SetA,SetB)* for building the intersection of two object sets

diff(objects(document),objects(footer)) for instance identifies the set of all available documents excluding their footers.

4.3.5 Examples

- Example 1a: Traditional WWW

To allow the traditional public (unrestricted) access to all documents in the repository the following authorization rules have to be given:

```
access('public',run,objects('document')).
access('public',refer,objects('document')).
```

Note, that the authorization to *refer* to documents implies the authorization to *refer* to all embedded bitstream documents as well as all sections (the latter only within the WOW framework).

- Example 1b: Restricted Areas

Allowing public access to all documents except for those residing in an area named '/private':

```
access('public',run,diff(objects('document'),
                        docs('/private/*'))).
access('public',refer,diff(objects('document'),
                        docs('/private/*'))).
access('admins',run,docs('/private/*')).
```

- Example 2: Anonymous publishing

Navigational access to documents without presenting authors' information authorization requires rules of the following form:

```

access('public', navigate, diff(objects('document'),
                                objects('address'))).
access('public', refer, diff(objects('document'),
                              objects('address'))).

```

5 Related Work

Several approaches have been proposed to connect (relational) databases to the Internet. Most of them use CGI scripts working as gateways between an HTTP daemon and the database's API (DB2WWW, KE Texpress, Oracle WebServe, R:WEB, Sapphire/Web, Sibylla, Software Engine, Spider (cf. [29])). Similar systems come up in the area of ODBMS (*O₂Web*, GemStone WWW Gateway). However, all of these systems use proprietary protocols or complex query languages to maintain information and handle links. WOW demands little from WWW authors and administrators. Documents and links are created in the usual manner.

In the area of document repositories there are some approaches that are comparable to WOW. STREAT/HyperStorM [1, 33] provides sophisticated management for SGML documents based on the VODAK ODBMS. However, integration of application objects and non SGML document types is not supported. The same applies for LINCKS [22] and several commercial products like Hyper-G/HyperWave [15]. Most of the commercial products moreover simply handle documents as flat files attached to hyperlink information or vice versa. In this way Hyper-G for instance or [19] restrict access for certain user groups only to documents, links, and bitstream information (content).

The ESPRIT project GOODSTEP [17] resulted in an integrated software development environment, consisting of a process control component and a set of generated syntax directed tools. These tools were specified using the GTSL description language [11]. WOW follows a similar approach but is adapted to the needs of WWW.

DCE Web [24] is a project that intends to combine OSF's Distributed Computing Environment with WWW needs for authorization. Requiring specialized browsers, DCE Web adds to WWW the advantages of DCE security, which includes distributed authentication, consistent group administration across domains, protection of nodes using access control lists (ACLs), and remote administration of ACLs. However DCE Web is focussed on low level authorization neglecting any documents' structures. The same holds for Sessioneer [3] and the work proposed by [19], that focus on authorization problems arising from the distribution of interconnected HTML documents.

6 Conclusion and Future Work

With standard file systems WWW developers are getting problems to maintain and update the big number of WWW documents, data processed by various applications, and connections among all of them. This information moreover is mostly given completely or not at all, which prevents the WWW from being used with maximum effectivity and ease.

WOW solves this problem in a powerful way. The *O₂* based WOW system allows a flexible and efficient handling of a large set of WWW documents, offering facilities for consistent updates of the repository. The WOW Generator and WOW Server provide the facility to process any type of document, to store and maintain them with *O₂*, and to let them being retrieved via Internet by WWW clients. Document integration is supported by adapting *O₂Web*'s idea of generic unparse methods. Access control is provided by the instantiation of an authorization model based on variable grained document structures maintained by WOW.

Still some security issues, also heavily demanded for business purposes, have to be investigated. Among these a reasonable user authentication and encryption are the most important. From the scientific point of view the meaning of negative authorizations and extensions to the authorization model concerning advanced aspects (e.g. distributed repositories, distributed publishing, decentralized administration) have to be worked out. Furthermore the maintenance of HTML-Forms has to be discussed. Since Forms typically go beyond the

scope of file system structures, i.e. by connecting to databases and using system resources, some efforts have been made in order to integrate them within the WOW framework.

References

- [1] K. Aberer, K. Böhm, and C. Hüser. The prospects of publishing using advanced database concepts. *Electronic Publishing Origination, Dissemination, and Design*, 6(4):469–480, Dec. 1993.
- [2] G. Almasi et al. Web*: A Technology to Make Information Available on the Web. In *Proc. of the 4th IEEE Workshop on Enabling Technology: Infrastructure for Collaborative Enterprises*, Berkley Springs, WV, 1995. IEEE Computer Society Press.
- [3] S. Anderson and R. Garvin. Sessioneer: Flexible Session Level Authentication with off the Shelf Servers and Clients. In *Proc. of the Third World-Wide Web Conference*, pages 1047–1053, Apr. 1995.
- [4] F. Bancilhon, C. Delobel, and P. Kanellakis, editors. *Building an object-oriented database system, the story of O₂*. Morgan Kaufmann, 1992.
- [5] T. Berners-Lee, R. T. Fielding, and H. F. Nielsen. *Hypertext Transfer Protocol – HTTP/1.1*, Jan. 1996. Internet Draft.
- [6] T. Berners-Lee, R. T. Fielding, and H. F. Nielsen. RFC 1945: Hypertext transfer protocol — HTTP/1.0, May 1996.
- [7] F. Buddrus. Enacting authorization models for object-oriented databases. In *Proc. of the 7th Int. Conf. on Database and Expert System Applications (DEXA)*, Zurich, Switzerland, Sept. 1996. IEEE Computer Society. Workshop Proc.
- [8] F. Buddrus and M. Bellavia. Surfing an ODBMS (maintaining WWW documents with O₂). In *Proc. of the 4th Int. Conf. on Document Analysis and Recognition (ICDAR)*, Ulm, Germany, Aug. 1997.
- [9] S. Castano, M. G. Fugini, G. Martella, and P. Samarati. *Database Security*. Addison Wesley - ACM Press, Wokingham, England, 1995.
- [10] J. H. P. Eloff, R. Holbein, and S. Teufel. Security classification for documents. *Computer & Security*, 15:55–71, 1996.
- [11] W. Emmerich, P. Kroha, and W. Schäfer. Object-oriented database management systems for construction of CASE environments. *Lecture Notes in Computer Science*, 720, 1993.
- [12] E. B. Fernandez, E. Gudes, and H. Song. A model for evaluation and administration of security in object-oriented databases. *IEEE Transactions on Knowledge and Data engineering*, pages 275–291, Apr. 1994.
- [13] E. B. Fernandez, K. R. Nair, M. M. Larrondo-Petrie, and Y. Xu. High-level security issues in multimedia/hypertext systems. In P. Horster, editor, *Communications and Multimedia Security II, Proc. of the IFIP TC6/TC11 Int. Conf.* Chapman & Hall, London, 1996.
- [14] E. B. Fernandez, J. Wu, and M. H. Fernandez. User group structures in object-oriented database authorization. In J. Biskup, M. Morgenstern, and C. Landwehr, editors, *8th IFIP WG 11.3 Working Conference on Database Security*, 1994.
- [15] U. Flohr. Hyper-G organizes the Web — this could be the next big thing on the Web: a new hypermedia architecture that makes it easier to find what you’re looking for. *Byte Magazine*, 20(11), Nov. 1995.
- [16] J. Franks, P. Hallam-Baker, J. Hostetler, P. A. Luotonen, and E. L. Stewart. RFC 2069: An extension to HTTP: Digest access authentication, Jan. 1997.

- [17] GOODSTEP Team. The GOODSTEP Project: General Object-Oriented Database for Software Engineering Processes. In *Proc. of the Asia-Pacific Software Engineering Conf., Tokyo, Japan*, pages 410–420. IEEE Computer Society Press, 1994.
- [18] J. Gosling and H. McGilton. The Java Language Environment – A Whitepaper. Technical report, Sun Microsystems, Oct. 1995.
- [19] J. Kahan. A capability-based authorization model for the world-wide-web. In *Proc. of the 3rd World-Wide-Web Conf.*, pages 1055–1064, Apr. 1995.
- [20] G. Kessler and S. Shepard. A Primer On Internet and TCP/IP Tools. Request for Comments (Informational) RFC 1739, Internet Engineering Task Force, Dec. 1994.
- [21] D. E. Knuth. Backus Normal form vs. Backus Naur form. *Communications of the ACM*, 7(12):735–736, Dec. 1964.
- [22] P. Lambrix, M. Sjölin, and L. Pagdham. LINCKS - a platform for cooperative information systems. Technical report, Departement of Computer and Information Science, LiTH, Sweden, 1993.
- [23] L. Lemay. *Teach Yourself Web Publishing with HTML in a Week*. Howard W. Sams, Indianapolis, IN 46268, USA, 1995.
- [24] S. Lewontin. The DCE Web Toolkit: enhancing WWW protocols with lower layer services. *Computer Networks and ISDN Systems*, 27(6):765–771, 1995.
- [25] O2 Technology, 7 rue du Parc de Clagny, 78035 Versailles Cedex, France. *O₂Web User Manual*, 4.6 edition, May 1996.
- [26] D. Raggett, J. Lam, and I. Alexander. *The Definitive Guide to HTML 3.0*. Addison-Wesley, Reading, MA, USA, 1996.
- [27] T. Ritchey. *Java!* New Riders, 1995.
- [28] F. Rouaix. A web navigator with applets in caml. In *Proc. of the 5th WWW Conf., Paris*, page 12, May 1996.
- [29] J. Rowe. Accessing a Database Server via the World Wide Web. Technical report, COMSO, Inc., 1995.
- [30] P. Samarati and E. Bertino. An authorization model for a distributed hypertext system. *IEEE Transactions on Knowledge and Data engineering*, pages 555–562, Aug. 1996.
- [31] E. H. Spafford and S. A. Weeber. User authentication and related topics: An annotated bibliography. Technical Report Purdue Technical Report CSD-TR-91-086, 1991.
- [32] T. Tessier. Using JavaScript to create interactive Web pages. *Dr. Dobb's Journal of Software Tools*, 21(3):84, 86–89, 96–97, Mar. 1996.
- [33] M. Volz, K. Aberer, and K. Böhm. Applying a Flexible OODBMS-IRS-Coupling to Structured Document Handling. In *Proc. of the 12th Int. Conf. on Data Engineering*, pages 10–19, 1996. New Orleans.
- [34] N. Wizards. Internet Domain Survey, July 1996. Survey, Network Wizards, July 1996.