Analysis. Journal of the Society for Information Science, 1990, 41(6), 391-407.

[EIT 94] McGuire, J. EIT-Link-Verifier-Robot.<URL:http://wsk.eit.com/wsk/dist/doc/admin/Webtest/verify_links.html>

[Fielding 94] Fielding R. Maintaining Distributed Hypertext Infostructures: Welcome to MOMspider's Web. Proceedings of the First International Conference on the World Wide Web, May 1994.

[Kappe & Pani 91] Kappe, F., Pani, G. The Architecture of a Massively Distributed Hypermedia System. IIG Report 341, IIG, Graz University of Technology, Austria, Sept. 1991.<URL:ftp://iicm.tu-graz.ac.at/pub/Hyper-G/doc/>

[Kappe 94] Kappe F., Andrews K., Faschingbauer J., Gaisbauer M., Pichler M., Schipflinger J.: "Hyper-G: A New Tool for Distributed Hypermedia". <URL:ftp://iicm.tu-graz.ac.at/pub/Hyper-G/doc/>

[Kernighan 84] Kernighan, B. W., Pike, R. The UNIX Programming Environment. Prentice-Hall Software Series,1984.

[Lavenant 94] Lavenant, M. G., Kruper, J. A. The Phoenix Project: Distributed Hypermedia Authoring. In Proceedings of the First International Conference on the World-Wide Web, May 1994.

[Martin 95] Martin, V. There Can be Only One: A Summary of the Unix Standardization Movement, Crossroads, February 1995 <URL:http://info.acm.org/crossroads/xrds1-3/unix-standards.html>

[Mealling 94] Mealling, Michael, Specification of Uniform Resource Characteristics, April 5, 1994 <URL:ftp://cnri.reston.va.us/internet-drafts/draft-ietf-uri-urc-00.txt>

[Mukherjea 94] Mukherjea, S., Foley, J., Hudson, S. Interactive Clustering for Navigating in Hypermedia Systems, ACM European Conference on Hypermedia Technology, September 1994, Edinburgh, Scottland.

[Nebel 94] Nebel, E., Masinter, L. Form-based File Upload in HTML. Internet-draft, November 30, 1994. <URL:ftp://parcftp.xerox.com/transient/file-upload/draft-ietf-html-file-upload-01.txt>

[Pitkow 93] Pitkow, James. HTML_ANALYZER <URL:http://www.cc.gatech.edu/pitkow/html_analyzer/README.html>

[Pitkow 95] Pitkow, James. Automatic Keyword Extraction using LSI (document in preparation).

[Sollins 94] Sollins K., Masinter L., Requirements of Uniform Resource Names, March 26, 1994. <URL:ftp://cnri.reston.va.us/internet-drafts/draft-sollins-urn-00.txt>

[Spry 94] Spry, Inc. SafteyWEB - Secure HTTP Server product announcement. <URL:http://www.spry.com/secure/betalet.html>

[Weibel 94] Weibel, S., Miller, E., Godby, J., LeVan, R. An Architecture for Scholarly Publishing on the World-Wide Web. In Proceedings of the Second International WWW Conference '94. <URL:http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Pub/weibel/weibel_www_paper.html>

ACKNOWLEDGEMENTS

Thanks to all members of the Graphics, Visualization, & Usability Center and its director, Dr. Jim Foley, for their support and help. James would also like to thank Dr. Jorge Vanegas for implicit funding throughout the development and testing of the prototype.

AUTHOR INFORMATION

JAMES PITKOW received his B.A. in Computer Science Applications in Psychology from the University of Colorado Boulder in 1993. He is a Graphics, Visualization, & Usability graduate student in the College of Computing at Georgia Institute of Technology. His research interests include user modelling, adaptive interfaces, and usability.

KIPP JONES received his B.S. in Computer Science from the University of Nebraska at Kearney in 1988. Currently pursuing a Master's degree at the Graphics, Visualization, & Usability center in the College of Computing at Georgia Tech. Research interests include such diverse topics as distributed publishing, 3D distributed object based environments, and physically realistic animation of human motion.

ments may necessitate the issuing of a new URN, though, we leave this up to the author to decide, while providing hints about how much has changed structurally, i.e. the number of links which have changed.

DISCUSSION

The focus of our initial environment is the publishing of hypertext documents. This is a notable contrast to some recent progress in the area of digital libraries which provide facilities for static document service, but do not facilitate the database maintenance process [Davis, 1994]. We firmly believe that the continued development of tools to facilitate management of dynamic information ecologies, like the Web, remain paramount to their long-term viability.

Ancillary to the act of publishing is the abstraction of the database management system from the author. This abstraction should be maintained throughout the publishing cycle in a consistent manner. A first pass at this process is represented in our prototype work by removal of the burden of link maintenance and hyperlink integrity within the database from the author. We note that this abstraction parallels our initial design goal of maximizing automation while preserving the authority of the author.

Recent work in authoring has primarily focused on the wide-area authentication and authorization support and the front-end WYSIYWIG HTML editors [Lavenant 94]. Our approach has been somewhat different. We are not so much concerned about the actual creation of the document, but have focused more on the publishing aspects. While direct and centralized publishing scenarios rely on the underlying security and authentication mechanism provided by the underlying operating systems, we have relied upon the 'basic' authentication provided by most Web browsers, combined with out-of-band negotiation for access. We feel that there are several available solutions which provide additional security, including available RSA based client and servers, any of which could be used in combination with our environment. Practically speaking, authentication and secure transmissions are client-server protocol issues.

Key to the success of the Web and our publishing environment is a methodolgy for external link maintenance. Two scenarios exist that can be incorporated into our publishing environment with a simple protocol. Specifically, when a document is published that contains a link to an external document, an "ADD_LINK" message can be sent to the corresponding publishing environment of the external document. This message contains the URL of the citing document and the document being linked. Thus, each link database contains complete in-link information on a *global* scale. Similarly, when a document is deleted, the publishing environments of all referencing documents can be notified via a "DELETE_LINK" message. This message contains the anchor content and link to remove as well as the document to remove the link from. Note that this solves *all* referential integrity problems in a non-centralized manner and thus initially seems to scale well.

CONCLUSION

In this paper, we presented an environment for WWW publishing intended to ease the task of publishing a document as well as that of disseminating the information about the published document. Building on the work of others, we were able to automate a significant portion of the publishing task. We recognize that HTML publishing is only a portion of the document publishing activity on the Web. This has provided motivation for providing as extensible a framework as possible. A significant addition to authoring tools will be the provision for disseminating the published information to the proper information lookup source or indexing service. Our prototype is intended to work with the developing WHOIS++ service to provide for resource discovery.

REFERENCES

[Arena 94] Arena <URL:http://info.cern.ch/hypertext/WWW/Arena/Welcome.html>

[Berners-Lee 92] Berners-Lee, T. J., Calliau, R., Groff, J-F., Pollermann, B. World-Wide Web: An Information Infrastructure for High-Energy Physics. Presented at "Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics", January 1992.

[Berners-Lee 94] Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H., and Secret, A. (1994). The World-Wide Web. *Communications of the ACM*, 37(8):76-82.

[Bowman 94] Bowman, C., Danzig, Peter B., Manber, Udi and Schwartz, Michael F (1994). Scalable Internet Resource Discovery: Research Problems and Approaches. *Communications of the ACM*, 37(8):98-107.

[Connolly 94] Connolly, D. W., Gaither, M. HaLsoft HTML Validation Service. <URL:http://www.hal.com/users/connolly/html-test/service/validation-form.html>

[Davis 94] Davis, J. and Lagoze, C. "Drop-in" publishing with the World-Wide Web. In Proceedings of the Second International WWW Conference '94.

[Deutsch 94] Deutsch P., Schoutlz R., Flatstrom P., and Weider C., Architecture of the WHOIS++ service, April 6, 1994. <URL:ftp://cnri.reston.va.us/internet-drafts/draft-ietf-wnils-whois-arch-00.txt>

[Deerwester 90] Deerwester S., Dumais S., Landauer T., Furnas G., Harshman, R. Indexing by Latent Semantic

Security with respect to file upload is not covered in detail in this paper. We note however that the RSA secure servers and clients are now available which preclude the need for adding this functionality to the publishing environment [Spry 95].

The Publish program takes as input a file name as well as the desired URL to be used to identify the file. An HTML form is used to receive input from the author regarding the document. A recent internet-draft [Nebel 94] specifies a method to be used for form-based file uploading on the client side. This allows the author to specify a file not accessible to the server, i.e. as in the distributed publishing scenario. Additional parameters can also be sent including naming scheme to use for URN creation (DNS or IANA). Upon submission, Publish first verifies the authority of the publisher via either: 1) lookup in the publisher database, which is maintained by the local Webmaster or 2) operating system supported user identification. Once the verification process is completed, the program extracts all URLs contained within the file as well as other document attributes.

At this point, the integrity of the embedded hyperlinks is verified and the HTML validated. Capitalizing on the work of others, we utilize the services offered by HAL and EIT's HTML analyzer. The author is informed of any warnings that occur. It may be that the document being published contains hyperlinks to documents not yet published. For this scenario, the author is allowed to override the warnings and mandate final document publication.

In accordance with the latest URC and URN specifications, the author name, file size, file creation time, last modified time, current capabilities as per the permissions field in UNIX, and MIME content type are determined via a combination of inode, "/etc/passwd" and "/etc/group" mining (see above URC example). We note that more sophisticated meta-information generation software can be inserted into this phase of processing, e.g. automatic keyword extraction [Pitkow 95]. The URN is composed based upon either a DNS or IANA naming scheme as determined by the author submitting the document. All of the above information is then formatted into HTML and returned to the user for editing, though the editing of certain fields is not enabled, e.g. file size, creation time, etc.

When the author approves the document for publication, the document is parsed and all hyperlinks are extracted and entered into the document-outlink database and an entry is made in the document-inlink database as well. Then, the file is copied over (direct/centralized publishing) or extracted from the passed in parameters via the file upload method (distributed publishing) into the server's document area. The file does not maintain ownership by the author; it is now owned by the UID running the http server. This prevents authors from indirectly manipulating documents. Our current prototype dumps the URC to a file, which is transferred to the local WHOIS++ server manually. We note that this stage will be replaced with the issuing of the CREATE TEMPLATE command to the local WHOIS++ server once the latter's security code stabilizes. Finally, an entry is made to the version control system that notes the publication of the document. Our implementation uses SCCS for version control, though we note the ease of other system inclusion.

Deletion also follows a two phase process. During the initial phase, the author enters the URL of the file to be deleted. The Delete code resolves the URL to the corresponding URC and URN. The code also checks the publisher database for authority as well as the permissions of the file. In order for the deletion to be viewed as valid, the author information in the URC must match the UID of the user making the request for deletion. This resolution of author name to UID is explicitly provided by standard Perl system calls in both the direct and centralized publishing scenarios. Author name resolution for distributed maintenance is handled via a user lookup within the database. Once verified, the URC is passed back to the author for final confirmation of deletion.

The Delete code is destructive. Not only is the intended file deleted, but all links to the now deleted URL within the server's document space are removed. This process is simplified by the document-inlink database since it contains the list of files that point to the recently deleted file. For each file in the list, the HTML is parsed and the anchor identified and removed. The new version of the file is checked into the version control system with the comment field stating which link was removed. A new URC is not generated for the file as the structural changes are currently assumed to be minimal. Since the publishing environment contains a database of publisher information, the author of the changed file is notified via e-mail that their file has been changed. Furthermore, a weighting scheme that determines the amount of structural change that resulted by the removal of the anchor could be calculated and used to suggest the appropriateness of a new version. Neither of the two latter functionalities are currently implemented in our prototype. As with the publishing code, all changes are logged to the file which is manually transferred to the WHOIS++ server, though we plan for this communication to be performed automatically.

Updates to existing documents are relatively simple. The author is presented with a screen on their WWW browser that enables them to enter the URL they wish to update, the filename of the document to replace the URL, and a radio button asking whether they want to create a new version within the URC that already exists for the document. Once the permissions of the author have been checked, the new document is parsed and its set of links added to the document-outlink database, overwriting the previous values. Commands are issued to the version control system which update the document and the change is noted to file for transfer to the WHOIS++ server. Note that updates to docu-

system and user errors. Current WWW environments do not support the author in this manner. This lack of support by the environment is less than optimal, requiring the author to handle tasks which are not in direct support of their primary goal - providing information. Group access to documents can be supported by using a version control system. This can be handled by allowing only one person to edit the document at a time, locking out all others. Alternatively, different resource sharing collaborative schemes could be implemented.

An ideal system would protect documents from unauthorized activities. In particular, only desired authors should be allowed to control document content and existence. In addition, document transfer and copying, whether centralized or distributed, need to be secure.

DESIGN DECISIONS

We had one basic design goal throughout all areas of implementation: ***automate as much as possible.*** This does not translate to moving the loci of control away from the publisher, but rather allowing the publisher to have final editing authority over as much automatically generated information as possible. That is, why should a user have to enter all the fields of a URC, when most can be extracted by an intelligent system? The same approach towards balancing tasks between author and system applies to hyperlink maintenance. If a document is about to be deleted, the author can be prompted for final authority to remove all occurrences of links to this document within the local document space and have the systems update all the appropriate files automatically.

One of the most important contributions with respect to the Web may be the refinement and inclusion of architectures that support global information location and global resource discovery. One such architecture, the WHOIS++ Index Service, is being developed by members of the Integration of Internet Information Resources Working Group and the Whois Network Information Lookup Service Working Group areas of the IETF. Coupled with the work of the Uniform Resource Identifiers Working Group on URN and URC, the WHOIS++ architecture holds much promise for both information location and shallow discovery tasks. Our publishing environment was explicitly designed to readily integrate and expand upon the basic functionality of these efforts.

Similar to the Harvest [Bowman 94], the WHOIS++ architecture attempts to handle the resource location and discovery problems. The WHOIS++ scheme utilizes URC's and URNs for name permenance and location independent name resolution. Below is an example of a URC with an embedded URN as generated by our prototype (please refer to references for a complete description and explanation):

```
URN:DNS:urn.cc.gatech.edu:/tmp/test.html
Author: World Wide Web Maintainer
{:
 Signature: not available
 Title: Georgia Tech: College of Computing Home Page
 Created: Thu Dec  8  0:55:02 1994
 Publisher: Georgia Tech College of Computing
 Keywords: Univversity Computing Information
 content-type: text/html
 Size: 3052
 Version: 1.0
 Pub+Date: Thu Dec  8  0:55:02 1994
 LIFN: not available
 Abstract: not available
 Last_Modified: Thu Dec  8  0:55:02 1994
 Notes: none
}:
URL:http://www.cc.gatech.edu/CoC.test4.html
```

IMPLEMENTATION

Our prototype was written in Perl 4.36 running on Sun OS 4.1.3 within a networked community of over 250 machines. Being a centralized publishing environment, all machines have access to our HTTP server and its document filesystem. In an attempt to increase the availability and scalabiltiy of our efforts, we chose to use a WWW browser interface to facilitate the publishing process rather than a customized GUI. Additionally, we chose to hand code the databases needed for link and author record keeping so as not to limit the prototype to proprietary databases. As a result, the databases used in our prototype are by no means overly efficient and do not offer the expressiveness of high level query languages like SQL. Still, fulfillment of these design goals essentially enables most direct and centralized publishing environments to readily use our prototype. The prototype is available via <URL:ftp://ftp.cc.gatech.edu/pub/software/publishing/ipe.tar>.

The prototype consists of several programs along with a library of general purpose routines (see Figure 3). These programs are supplemented by three customized databases: a publisher database that maintains records of registered authors, a database that maps a given URL to files that link to it (document-inlink relation), and a database that maps a given file to the URLs contained within its content (document-outlink relation).

"Basic" authentication between the client and server is used to provide a means to access the database. A slight modification to the server allowed the verified user id to be passed on to the authoring environment, which then performed a simple lookup within the access control list. The access control list consists of individual and group authority for both individual documents as well as directory level access. The negotiation for this access is handled "out of band," presumably using a something akin to e-mail and system accounts. The local Webmaster is responsible for maintaining the access control list.

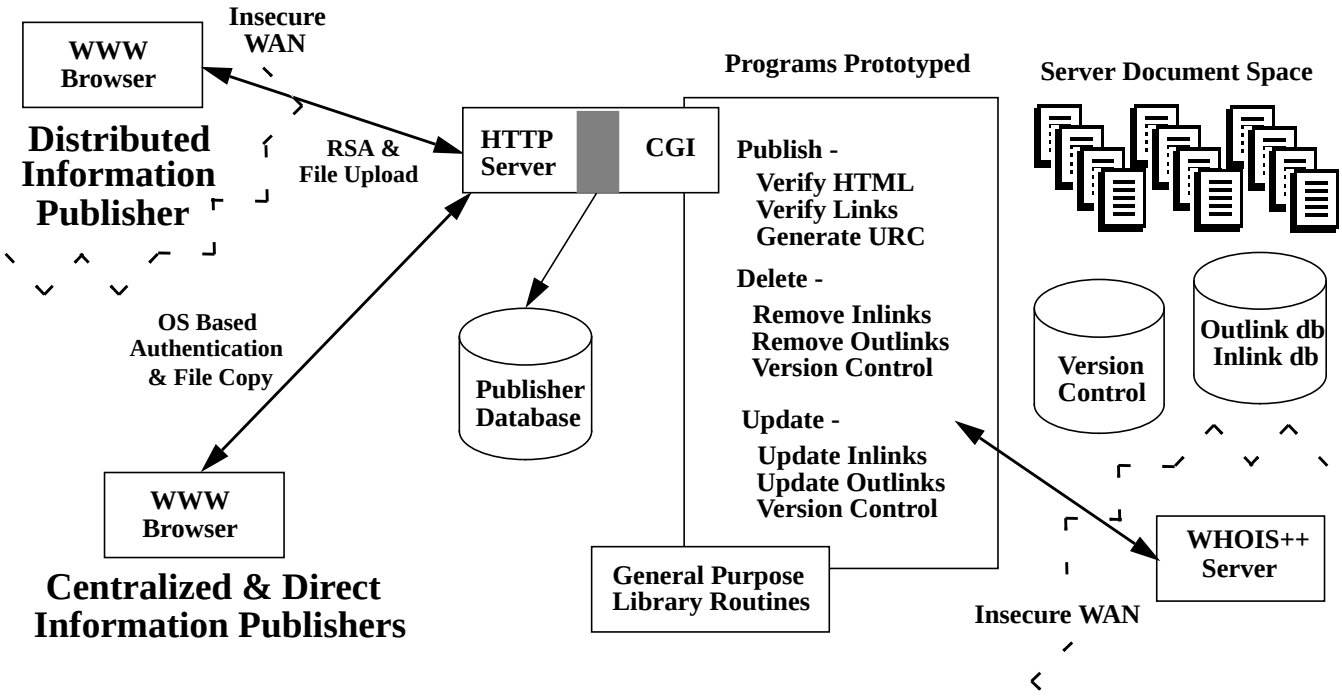

Figure 3. The above diagram overviews the architecture used for the implementation of the publishing environments. Distributed publishing is shown as occurring across an insecure WAN. Both centralized and direct publishing occurs on secure filesystems. Additionally, the document space, version control, and database facilities exist on the server filesystem. Communication to WHOIS++ is assumed to be insecure.

AN IDEALIZED PUBLISHING ENVIRONMENT

Currently, document creation and editing suffers from the lack of robust WYSIWIG (what you see is what you get) editors, though this seems to be a short-lived problem. Of more importance to the document creation process is assisting the author in adding documents to large repositories. In other words, how does an author add hyperlinks to known and unknown documents that are semantically related *in the same database*. The possibility of linking unknown documents could occur if a new author is making contributions to an existing repository, or if there are numerous authors contributing to the same area. Visualization systems that display the structure of hypermedia [Mukherjea 94] and semantically based clustering algorithms [Deerwester 90] are viable initial solutions.

For Web documents, checking refers to the validity of the HTML as well as the availability of hyperlinks. Several packages exist that facilitate hyperlink maintenance and availability [Pitkow 93, Fielding 94, EIT 94], as well as HTML correctness [Connely 94, Arena 94]. Few environments exist that support version control, i.e. controlling multiple revisions of documents.

Publishing environments can perform several tasks during this stage. First, link databases can be updated to reflect both the internal and external links (links to documents on local and other servers respectively) contained in the document. Second, document meta-information can be extracted and presented to the author for final approval. Third, the document can be copied into the server area in a controlled, secure manner. This entails user authorization as well as version control. Finally, the appropriate scalable architecture can be notified of the newly available document. An ideal system would directly support all of the above functionalities[4].

Document maintenance occurs when either information needs to be changed, or a redesign of the document is necessitated. Both potentially involve content as well as structural modifications. Ideally, an environment would support the maintenance process by pushing the database maintenance tasks away from the author, requiring the publishing environment and/or the underlying database management system to assume responsibility for these tasks. This includes the maintenance of the referential integrity of hyperlinks within the database, the verification of the document in so far as possible, and the capability to gracefully recover by offering versioning control.

Upon completion of the editing and testing cycle, the author re-publishes the document. As with initial document publication, the author should be presented with information regarding the validity of the document submitted. Additionally, information from the original document, namely the resource name and the resource characteristics, could be presented to the author for update. An interesting issue at this point involves the determination of sufficient modification to require a new resource name. That is, at what point does a document change sufficiently to warrant a new resource name?

Version control ought to be provided to allow for multiple revisions of documents as well as graceful recovery from

4. Additionally, URLs could be resolved to URNs during this phase for external and local hyperlinks. We note that our system does not handle this case as the nature of including URNs into HTML has not been finalized.

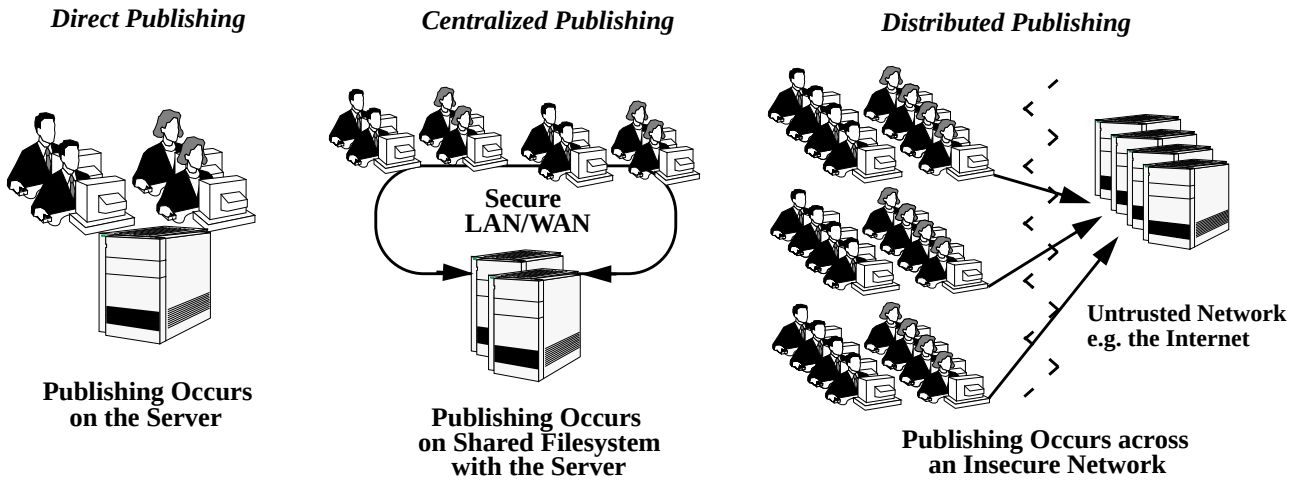

Figure 2. Three different publishing scenarios.

Our prototype attempts to combine the advantages of link database systems with the Web. Specifically, we developed a prototype environment that:

1. explicitly supports the notion of publishing
2. automatically generates a URC and a URN
3. provides versioning facilities for published documents
4. provides hyperlink database and maintenance facilities
5. propagates all activities to a local WHOIS++ server

The act of publishing on the Web (See Figure 1) does not drastically differ methodologically from traditional publishing. Electronic publishing does differ in document testing and version control. With the intent of explicating the motivations behind our prototype, we briefly discuss Web publishing scenarios and the problems associated with each scenario with respect to current Web server and publishing systems. The paper will then focus on the prototype's implementation as well as outline our position on many of the issues encountered during development. Finally, we discuss areas not covered in our initial implementation.

PUBLISHING SCENARIOS

Once a document has been tested, essentially three types of publishing scenarios exist for the information provider. The first occurs when the provider is publishing from the filesystem of the intended server. We refer to this case as *direct publishing* since the filesystem being utilized is directly accessible to the publisher and to the server. No new security issues are introduced above and beyond those already in existence for the filesystem and server. Note that one or many publishers may seek access to the server's repository.

The second scenario occurs when the publishers and the sever are accessible via a secure local area network[1] (LAN). Security of file transfer is not an issue since the operating and file management systems are assumed to be mutually trusted. In other words, the server-side publishing environment ought to be able to issue a copy file command without abnormal threat of intruder. Most cases in this class involve multiple publishers with potentially several servers. We refer to this scenario as *centralized publishing* since the publishers and servers share a central view of the filesystem.

The third situation occurs when the publisher and server do not share a common file space. In this case, the document to be published must be transferred to the server in a secure and efficient manner for entry into the server's document repository. This most likely involves a large number of publishing groups as well as multiple servers. We refer to this scenario as *distributed publishing* since other publishers and servers view the filesystem as distributed. Figure 2 provides a diagram of these scenarios. Our environment provides support for all of the above scenarios.

From the UNIX[2] standpoint, the above scenarios can be viewed as: editing a document directly within the server's document space (direct publishing), copying a file into the server area (centralized publishing), or submitting files to the local Webmaster for publication via e-mail or similar methods of file transfer (distributed publishing). Regardless, much of the intelligence and functionality that does not exist in current Web-based systems can occur during the publishing phase. That is, we desire a server environment that has more information about a document than determining at the time a document is requested whether an inode[3] exists.

1. In special cases, this scenario can apply to wide area network (WAN).

2. UNIX is a registered trademark of X/Open [Martin 95].
3. A file has several components: a name, contents, and administrative information such as permissions and modification times. The administrative information is stored in what is referred to as the *inode* in UNIX systems [Kernighan 84].

Towards an Intelligent Publishing Environment

James E. Pitkow & R. Kipp Jones

Graphics, Visualization, & Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
E-mail {pitkow, kipp}@cc.gatech.edu

ABSTRACT

This paper presents an environment for publishing information on the World-Wide Web (WWW). Previous work has pointed out that the explosive growth of the WWW is in part due to the ease with which information can be made available to Web users [Weibel 94]. Yet this property can have negative impacts on the ability to find appropriate information as well as on the integrity of the information published. We present a prototype environment that facilitates the publishing of documents on the Web by automatically generating meta-information about the document, communicating this to a local scalable architecture, e.g WHOIS++, verifying the document's HTML compliance, maintaining referential integrity within the local database, and placing the document in a Web accessible area. Additionally, maintenance and versioning facilities are provided. This paper first discusses an idealized publishing environment then describes our implementation, followed by a discussion of salient issues and future research areas.

KEYWORDS

Internet tools, electronic publishing, hyperlink databases

INTRODUCTION

Though one of the original intentions of the Web was for the publication of time-sensitive information [Berners-Lee 92], very few tools exist that explicitly facilitate the notion of publishing on the Web. Such endeavors may prove critical to the long lasting success of the Web, as the amount of accessible information and the diversity of the authoring population increases.

Research on systems such as Hyper-G [Kappe 94] nearly parallel the work presented in this paper. Though a complete comparison between the Web and Hyper-G is beyond the scope of this paper, several differences exist. First, attempts have been made to conform to current accepted practices and developing standards where possible. This includes the incorporation of the latest Internet Engineering Task Force (IETF) drafts for HTML, Uniform Resource Characteristic (URC) [Mealling 94] and Uniform Resource Name (URN) [Sollins 94], and WHOIS++ [Deutsch 94] name resolution into our prototype. While Hyper-G also manages meta-information and unique document identification spaces, no design decision is evident that such choices were made in lieu of current standardization practices.

Unlike the Web, Hyper-G maintains a centralized link database in addition to a document server. This separation of document content from hyperlink structure has several advantages. Most notably, maintenance of referential integrity, i.e. ensuring that all links point to existing documents, of the links is facilitated. The link database can also facilitate visualizations of the local database as well as the automated maintenance of hyperlinks upon document deletions. A disadvantage of Hyper-G is that data fusion (combination of document content and structure) must be performed by the client in order to view the document. The Web does not separate content and structure and thus does not need to perform data fusion, but lacks link maintenance facilities.

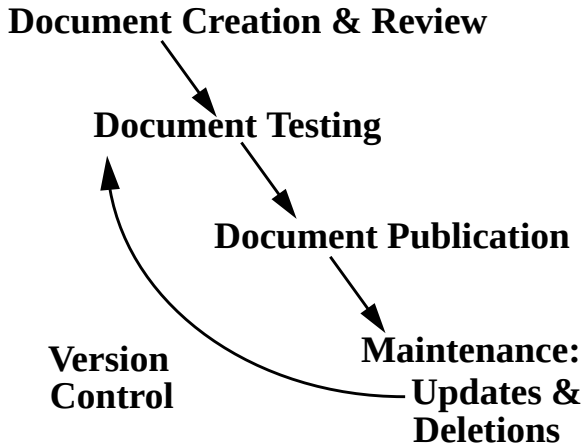

Figure 1. Web publishing processes