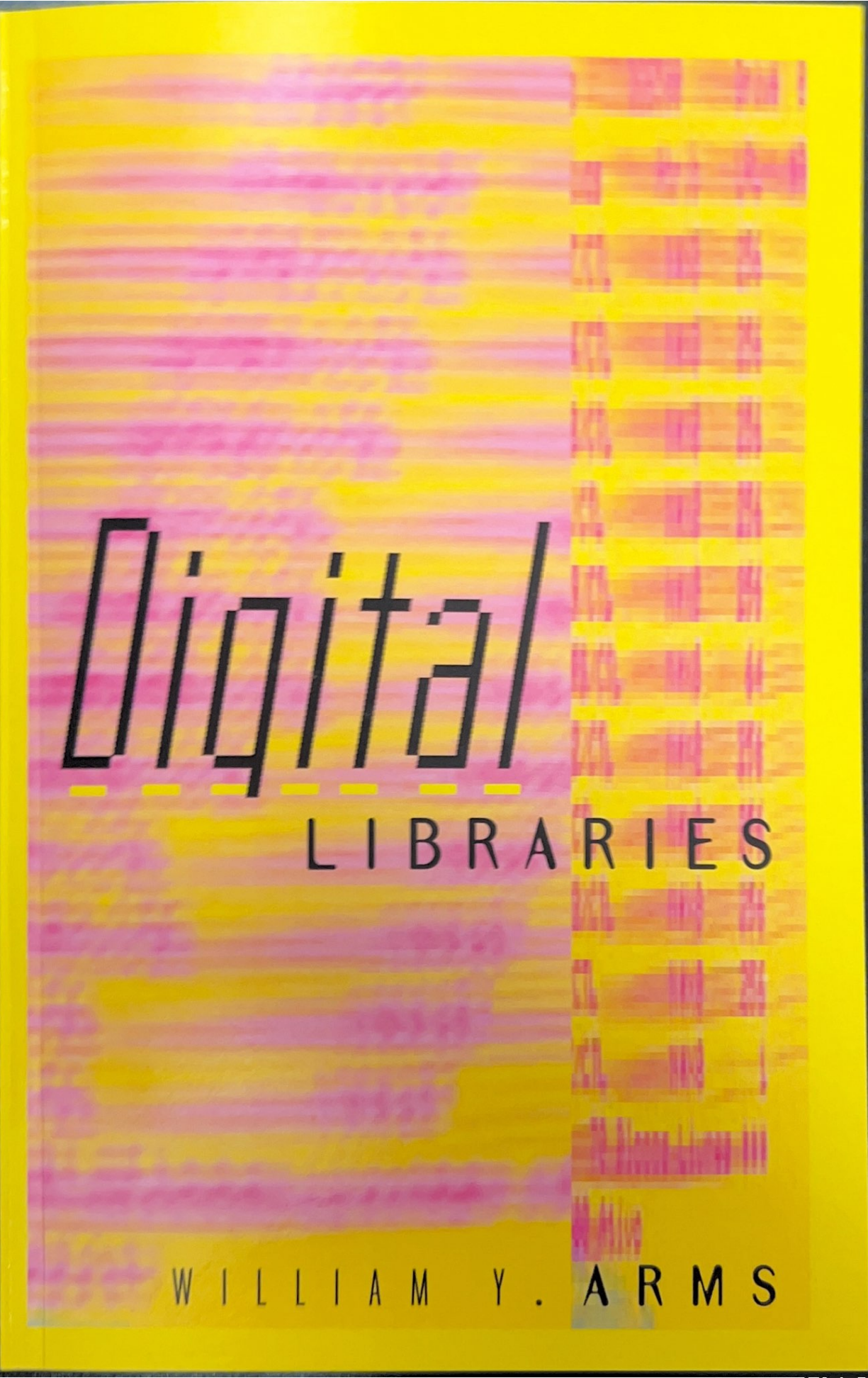




Digital

LIBRARIES

WILLIAM Y. ARMS

Digital Libraries

William Y. Arms

The MIT Press
Cambridge, Massachusetts
London, England

© 2000 Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

Set in Sabon by The MIT Press.

Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Arms, William Y.

Digital libraries / William Y. Arms.

p. cm. — (Digital libraries and electronic publishing)

Includes bibliographical references and index.

ISBN 0-262-01880-8 (alk. paper)

1. Libraries—United States—Special collections—Electronic information resources. 2. Digital libraries—United States.

I. Title. II. Series.

Z692.C65A76 2000

025'.00285—dc21

99-14773

CIP

Libraries, Technology, and People

This is a fascinating period in the history of libraries and publishing. For the first time, it is possible to build large-scale services where collections of information are stored in digital formats and retrieved over networks. The materials are stored on computers. A network connects the computers to personal computers on the users' desks. In a completely digital library, nothing need ever reach paper.

This book provides an overview of this new field. Partly it is about technology, but equally it is about people and organizations. Digital libraries bring together facets of many disciplines, and experts with different backgrounds and different approaches. The book describes the contributions of these various disciplines and how they interact. It discusses the people who create information and the people who use it—their needs, motives, and economic incentives. It analyzes the profound changes that are occurring in publishing and libraries. It describes research into new technology, much of it based on the Internet and the World Wide Web. Among the topics are technical aspects of computers and networks, librarianship and publishing, economics, and law. The constant theme is change, with its social, organizational, and legal implications. One book cannot cover all these topics in depth, and much has been left out or described at an introductory level. Most of the examples come from the United States, with prominence given to universities and the academic community; however, the development of digital libraries is worldwide, with contributions coming from many sources. Specialists in big American universities are not the only developers of digital libraries, though they are major contributors. There is a wealth and diversity of innovation in almost every discipline, in countries around the world.

An informal definition of a digital library is a managed collection of information, with associated services, where the information is stored in digital formats and accessible over a network. A crucial part of this definition is that the information is managed. A stream of data sent to earth from a satellite is not a library. The same data, when organized systematically, becomes a digital library collection. Most people would not consider a database containing financial records of one company to be a digital library, but would accept a collection of such information from many companies as part of a library. Digital libraries contain diverse collections of information for use by many different users. Digital libraries range in size from tiny to huge. They can use any type of computing equipment and any suitable software. The unifying theme is that information is organized on computers and available over a network, with procedures to select the material in the collections, to organize it, to make it available to users, and to archive it. In some ways digital libraries are very different from traditional libraries, yet in other ways they are remarkably similar. People do not change because a new technology is invented. They still create information that has to be organized, stored, and distributed. They still need to find information that others have created, and to use it for study, reference, or entertainment. However, the form in which the information is expressed and the methods that are used to manage it are greatly influenced by technology, and this creates change. Every year, the quantity and variety of collections available in digital form grows, while the supporting technology continues to improve steadily. Cumulatively, these changes are stimulating fundamental alterations in how people create information and how they use it.

To understand these forces requires an understanding of the people who are developing the libraries. Technology has dictated the pace at which digital libraries have been able to develop, but the manner in which the technology is used depends on people. Two important communities are the source of much of this innovation. One group, the information professionals, includes librarians, publishers, and a wide range of information providers, such as indexing and abstracting services. The other community contains computer science researchers and their offspring, the Internet developers. Until recently these two communities had disappointingly little interaction; even now it is commonplace to find a computer scientist who

knows nothing of the basic tools of librarianship, or a librarian whose concepts of information retrieval are years out of date. Since the introduction of the web, however, there has been much more collaboration and understanding. Partly this is a consequence of digital libraries becoming a recognized field for research, but an even more important factor is the greater involvement of users. Low-cost equipment and simple software have made electronic information directly available to everybody. Authors no longer need the services of a publisher to distribute their works. Readers can have direct access to information without going through an intermediary. Many exciting developments come from academic or professional groups who develop digital libraries for their own needs. Medicine has a long tradition of creative developments. The pioneering legal information systems were developed by lawyers for lawyers. The World Wide Web was initially developed by physicists for physicists.

Economics

Technology influences the economic and social aspects of information, and vice versa. The technology of digital libraries is developing fast, and so are the financial, organizational, and social frameworks. The various groups that are developing digital libraries bring different social conventions and different attitudes to money. Publishers and libraries have a long tradition of managing physical objects—notably books, but also maps, photographs, sound recordings, and other artifacts. They evolved economic and legal frameworks that are based on buying and selling these objects. Their natural instinct is to transfer to digital libraries the concepts that have served them well for physical artifacts. Computer scientists and scientific users, such as physicists, have a different tradition. Their interest in digital information began in the days when computers were very expensive. Only a few well-funded researchers had computers on the first networks. They exchanged information informally and openly with colleagues, without payment. The networks have grown, but the tradition of open information remains. The economic framework that is developing for digital libraries shows a mixture of these two approaches. Some digital libraries mimic traditional publishing by requiring a form of payment before users may access

the collections and use the services. Other digital libraries use a different economic model. Access to their material is open to all. The costs of creating and distributing the information are borne by the producer, not the user of the information. This book describes many examples of both models and attempts to analyze the balance between them. Almost certainly, both have a long-term future, but the final balance is impossible to forecast.

Why Digital Libraries?

The fundamental reason for building digital libraries is a belief that they will provide better delivery of information than was possible in the past. Traditional libraries are a fundamental part of society, but they are not perfect. Can we do better? Enthusiasts for digital libraries point out that computers and networks have already changed the ways in which people communicate with each other. In some disciplines, they argue, a professional or a scholar is better served by sitting at a personal computer connected to a communications network than by making a visit to a library. Information once available only to the professional is now directly available to all. From a personal computer, the user is able to consult materials that are stored on computers around the world. Conversely, all but the most diehard enthusiasts recognize that printed documents are so much a part of civilization that their dominant role in storing and conveying information cannot change except gradually. Though some important uses of printing may be replaced by electronic information, not everybody considers a large-scale movement to electronic information desirable, even if it is technically, economically, and legally feasible. Here are some of the potential benefits of digital libraries.

The Digital Library Brings the Library to the User

To use a traditional library, a reader must go there. At a university this may take only a few minutes, but most people are not at universities and do not have a library nearby. Many engineers and many physicians have depressingly poor access to the latest information.

A digital library brings the information to the user's desk, either at work or at home. With a digital library on the desk top, the reader need never

The Internet and the World Wide Web

The Internet

As its name suggest, the Internet is an interconnected group of independently managed networks. Each network supports the technical standards needed for interconnection—the TCP/IP family of protocols and a common method for identifying computers—but in many ways the separate networks are very different. The various sections of the Internet use almost every kind of communications channel that can transmit data. They range from fast and reliable to slow and erratic. They are privately owned or operated as public utilities. They are paid for in different ways. The Internet is sometimes called an “information highway.” A better comparison would be to the international transportation system, which includes everything from airlines to dirt tracks.

The networks that make up the Internet originated in two ways. One line of development was the local-area networks that were created to link computers and terminals within a department or an organization. Many of the original concepts came from the Xerox Corporation’s Palo Alto Research Center (PARC). In the United States, universities were pioneers in expanding small local networks into campus-wide networks. The second source of network developments were the national wide-area networks. The best known of these was the ARPAnet, which by the mid 1980s linked about 150 computer science research organizations. Other major networks, such as Bitnet, have been almost forgotten. Their former users are now served by the Internet.

Since the early networks used differing technical approaches, linking them together was hard. During the late 1980s, the universities and the

Both of the aforementioned services were well established before the emergence of the web. The web has been so successful that many people forget that there are other effective ways to distribute information on the Internet. Both the Los Alamos archives and the RFC series now use web methods, but they were originally based on electronic mail and file transfer.

The World Wide Web

The World Wide Web (colloquially “the web”) has been one of the great successes in the history of computing. It ranks with the development of word processors and spreadsheets as a definitive application of computing. The web and its associated technology have been crucial to the rapid growth of digital libraries. This section gives a basic overview of the web and its underlying technology. Details of specific aspects are spread throughout the book.

The web is a linked collection of information on many computers on the Internet around the world. These computers are called *web servers*. Some of the web servers and the information on them are maintained by individuals and some by small groups, including university departments and research centers; others are large corporate information services. Some sites are consciously organized as digital libraries; others (some of them excellent) are managed by individuals who would not consider themselves librarians or publishers. Some web servers have substantial collections of high-quality information; others are used for short-term or private purposes, are informally managed, or are used for purposes (such as marketing) that are outside the scope of libraries.

The web technology was developed around 1990 by Tim Berners-Lee and colleagues at the Centre Européenne pour la Recherche Nucléaire (CERN), in Switzerland. It was made popular by Mosaic, a user interface developed by Marc Andreessen and others at the University of Illinois at Urbana-Champaign. Mosaic was released in 1993. Within a few years, numerous commercial versions of it followed. Of those, the most widely used are the Netscape Navigator and Microsoft’s Internet Explorer. These user interfaces are called *web browsers*, or simply *browsers*.

The basic reason for the success of the web can be summarized succinctly: It provides a convenient way to distribute information over the

Libraries and Publishers

Libraries have a proud tradition as early adopters of new technologies, including microfilm, online information services, and CD-ROMs. (The second typewriter imported into Brazil was for the National Library of Brazil to type library catalog cards.) The Internet and the web are the latest examples of libraries' embracing new technologies and adapting it to their needs.

Mainstream commercial products do not always fit the needs of libraries. The computing industry aims its products at the large markets provided by commercial businesses, science, medicine, and the military. In the United States, much of the basic research behind these products has been funded by the Department of Defense or by companies such as IBM. Until recently, these large markets had little interest in the management of the kinds of information that are found in digital libraries, and the products sold by the computing industry did not address important needs in information management. Therefore, libraries are accustomed to taking core technologies from other fields and tailoring them to their needs.

Resource Sharing and Online Catalogs

In libraries, the word *networking* has two meanings. Before modern computer networks, libraries had a wide variety of arrangements to share resources. These included interlibrary lending, reciprocal reading privileges between institutions, and the exchange of photocopies. The term used for these arrangements was *networking* or *resource sharing*. Resource-sharing services, such as the photocopying and document delivery provided by the British Library, enhanced the availability of information around the world.

Mounting large amounts of information online and keeping it current is expensive. Although the costs hardware fall continually, they are still noticeable. The big costs are associated with the licensing of data and the people who handle both the business aspects and the large data files. To reduce these costs, libraries have formed consortia so that one set of online data serves many libraries.

Information Retrieval

Information retrieval is a central topic for libraries. A user—perhaps a scientist, a doctor, or a lawyer—is interested in information on some topic and wishes to find the objects in a collection that cover the topic. This requires specialized software. During the mid 1980s, libraries began to install computers with software that allowed full-text searching of large collections. Usually the MARC records of a library's holdings were the first data to be loaded onto this computer, followed by standard reference works. Full-text searching meant that a user could search using any words that appeared in the record; the user did not have to be knowledgeable about the structures of the records or the rules used to create them.

Research in this field is at least 30 years old, but the basic approach has changed little. A user expresses a request as a *query*. This may be a single word (such as *cauliflower*), a phrase (such as *digital libraries*), or a longer query (such as “In what year did Darwin travel on the Beagle?”). The task of information retrieval is to find objects in the collection that match the query. Since a computer does not have the time to go through the entire collection for each search, looking at every object separately, the computer must have an index of some sort that enables it to retrieve information by looking up entries in indexes.

As computers have grown more powerful and as the price of storage has declined, methods of information retrieval have moved from carefully controlled searching of short records (such as catalog records or those used by abstracting and indexing services) to searching the full text of every word in large collections. In the early days, the expense of computer storage and processing power stimulated the development of compact storage methods and efficient algorithms. More recently, web search programs have intensified research into methods for searching large amounts of information distributed across many computers.

User Interfaces and Usability

A person who uses a library is conventionally called a *patron* or a *reader*. In computing, the usual term is *user* or *end user*. Whatever word is chosen, digital libraries are of little value unless they are easy to use effectively.

The relationship between people and computers is a subject of intensive research drawing on fields as diverse as cognitive science, graphic design, rhetoric, and mathematical modeling of computer systems. Some of the research aims to develop a theoretical understanding of how people interact with computers so that models of human information processing can be used to design appropriate computer systems. Other research helps the user comprehend the principles behind a computer system, thereby stimulating productive use of the services and the information that the system provides. This chapter is less ambitious. It concentrates on the methods that are in wide use today, offering some illustrations of experimental systems that show promise.

Change is one of the themes of digital libraries, and change is one of the problems in designing user interfaces. Traditional libraries are not easy to use effectively; however, they change slowly, and over the years their users develop expertise. Digital libraries evolve so quickly that every month brings new services, new collections, new interfaces, and new headaches. Users do not enjoy having to relearn basic skills, but change appears to be a necessary evil.

Partly because of the rapidity with which digital libraries change, their users' levels of expertise vary greatly. Much of the development of digital libraries has come out of universities, where there are many experts. Colleagues and librarians are at hand to help users; system administrators configure computers, install software, and track changes in the marketplace.

But with the spread of the Internet, digital libraries are being used by people who do not have access to such expertise and do not want to spend their own time learning techniques that may be of only transitory value. This creates tension: Advanced features are valuable to specialists, and they enable skilled users to work faster and be more effective, but digital libraries must be usable by individuals with minimal training.

Usability and the Design of User Interfaces

In discussing the usability of a computer system, it is easy to focus on the design of the interface between the user and the computer. But usability is a property of the entire system. All the components of a digital library must work together smoothly to make the library effective and convenient for patrons, for librarians, and for systems administrators.

Figure 8.1 shows a way to think about usability and the design of user interfaces. Any computer system's user interface is built on a *conceptual model* that describes the manner in which the system is used. Here are some typical conceptual models that are used to design digital libraries. (In practice, most digital libraries combine concepts from several such models.)

- In the classical library model, the user searches a catalog or index, selects objects from the results, then retrieves the objects from a repository.
- In the basic web model, the user follows hyperlinks between files.
- The Z39.50 protocol supports a conceptual model of searching a collection and then storing sets of results for subsequent manipulation and retrieval.

The right side of figure 8.1 illustrates the layers of design needed to implement any conceptual model. *Interface design* encompasses what

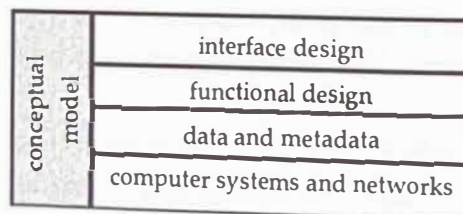


Figure 8.1

appears on the screen and how the user manipulates it; among its aspects are fonts, colors, logos, keyboard controls, menus, and buttons. *Functional design* specifies the functions that are offered to the user. Typical functions include selecting parts of a digital object, searching a list or sorting results, obtaining help, and manipulating objects that have been rendered on the screen. These functions are made possible by the *data and metadata* that are provided by the digital library, and by the underlying *computer systems and networks*. Panel 8.1 details how these aspects of usability apply to an important application.

Almost all present-day personal computers have a user interface of the style made popular on Apple's Macintosh computers, which was derived from earlier research at Xerox PARC. It uses the metaphor of files and folders on a desktop. Its characteristics include overlapping windows, menus, and a pointing device such as a mouse. Despite numerous attempts at improvement, this style of interface continues to dominate the computer market. An early Macintosh user transported through fifteen years and presented with a computer running Microsoft's latest system would find the interface familiar. Some new conventions have been introduced, and hardware has been improved greatly, but the same basic metaphor is still used.

The conceptual model is the desktop metaphor; files are thought of as documents that can be moved to the desktop, placed in folders, or stored on disks. Every windows-based user interface has this same conceptual model, although the interface designs vary (for example, Apple uses a mouse with one button, Microsoft uses two buttons, and Unix systems usually have three). The functions that support this model include open and closing files and folders, selecting them, moving them from one place to another, and so on. These functions differ little from one manufacturer to another, but the systems differ in the metadata that they use to support the functions. The desktop metaphor requires that applications be associated with data files. Microsoft and Unix systems use file-naming conventions; for example, the names of files that are to be used with a PDF viewer end in .pdf. Apple stores such metadata in a separate data structure, hidden from the user. Differences in the underlying computer systems permit some user interfaces to carry out several tasks simultaneously.

Panel 8.1
Page Turning

Page turning illustrates the aspects of user-interface design shown in figure 8.1. Conversion projects such as JSTOR and American Memory have collections of digital objects, each consisting of page images scanned from a book or some other printed material. The *conceptual model* is that the user interacts with an object in much the same manner as with a book. Often the pages will be read in sequence, but the reader may also go back a page or jump to a different page. Some pages, such as a table of contents or an index, may be identified as special. Since many personal computers have screens that are smaller than printed pages and have lower resolution, the conceptual model includes zooming and panning across a single page.

The *interface design* defines the actual appearance on the screen, such as the choice of frames, icons, colors, and visual clues. It also includes decisions about how the individual functions are offered to the user. The interface design determines the appearance of icons, the wording on the buttons, and their position on the screen. The design also specifies whether panning and zooming are continuous or in discrete steps. When we built a page turner at Carnegie Mellon University, the interface design maximized the area of the screen that was available for displaying page images. Most manipulations were controlled from the keyboard; the arrow keys were used for panning around an individual page, with the tab key used for going to the next page. An alternative design would have these functions controlled by buttons on the screen, but with less space on the screen for the page images.

The *functional design* must provide services that are equivalent to turning the pages of a book, including going to the first, the next, the previous, or the last page. There will be functions that relate to the content of a specific page, such as going to the page that has a specific page number printed on it and going to the table of contents. To support panning and zooming within a page, other functions of the user interface move the area displayed up or and down one screen, and zoom in or out.

The functions offered by a user interface depend on the digital objects in the collections and especially on the *structural metadata*. Typically, page images are stored as compressed files that can be retrieved in any sequence. To allow pages to be turned in sequence, structural metadata must identify the first page image and list the sequence of the other images. Going to the page with a specific page number printed on it requires structural metadata that relates the page numbers to the sequence of page images, since it is rare that the first page of a set is the page numbered 1. Zooming and panning require metadata that states the dimensions of each page.

A user interface is only as good as the performance of the underlying *computer system*. Delays in transmitting a page image across a network can be annoying to the user. One possible way to reduce this annoyance is to anticipate the user's demands by sending pages from the repository to the user's computer before the user requests them, so that at least the next page in a

sequence is ready in memory. This is known as *pre-fetching*. In the case of the Carnegie Mellon page turner, priority was given to achieving quick response—about a second to read and display a page image, even one transmitted over the busy campus network. This led to a “pipelined” implementation in which the first part of a page was being rendered on the user’s computer even before the final section was read into the repository computer from disk storage.

Browsers

The introduction of browsers (notably Mosaic, introduced in 1993) stimulated the design of better user interfaces for networked applications. Although browsers were designed for the web, they are so flexible that they are used as interfaces to almost every type of application on the Internet, including digital libraries. Before the emergence of general-purpose browsers, developers had to provide a separate user interface for each type of computer and each computing environment. These interfaces had to be modified whenever the operating systems changed—a monumental task that few attempted and almost none did well. By relying on web browsers for the actual interaction with the user, the designer of a digital library can now focus on how to organize the flow of information to the user, leaving complexities of hardware and operating systems to the browser.

Basic Functions

The basic function of a browser is to retrieve a remote file from a web server and render it on the user’s computer.

- To locate the file on the web server, the browser needs a Uniform Resource Locator. The URL can be typed in by the user, or it can be a link within an HTML page, or it can be stored as a bookmark.
- From the URL, the browser extracts the protocol. If it is HTTP, the browser then extracts from the URL the domain name of the computer on which the file is stored. The browser sends a single HTTP message, waits for the response, and closes the connection.
- If all goes well, the response consists of a file and a MIME type. To render the file on the user’s computer, the browser examines the MIME type and invokes the appropriate routines. These routines may be built into the browser or may be an external program invoked by the browser.

A simple web browser would offer little more than support for the HTTP protocol and routines to render pages in the HTML format. Since both HTTP and HTML are straightforward, a web browser need not be a complex program.

Extending Browsers beyond the Web

Browsers were developed for the web, and every browser supports the web's basic protocols and a few standard formats. However, browsers can be extended to provide other services while retaining the browser interface. Much of the success of browsers, of the web, and indeed of the whole Internet is due to this extensibility.

Mosaic and its successors have had the same three types of extensibility: one for data types, one for protocols, and one for the execution of programs.

Data types

With each data type, browsers associate routines to render files of that type. A few types have been built into all recent browsers, including plain text, HTML pages, and images in GIF format, but users can add additional types through mechanisms such as helper applications and plug-ins. A *helper application* is a separate program that is invoked by selected types of data. The source file is passed to the helper as data. For example, browsers do not have built-in support for files in the PostScript format, but many users have a PostScript viewer on their computer which is used as a helper application. When a browser receives a file of type PostScript, it starts this viewing program and passes to it the file to be displayed. A *plug-in* is similar to a helper application, except that it is not a separate program. It is used to render source files of non-standard formats, within an HTML file, in a single display.

Protocols

HTTP is the central protocol of the web, but browsers also support other protocols. Some, such as Gopher and WAIS, were important historically because they allowed browsers to access older information services. Others, such as NetNews, electronic mail, and FTP, remain important. A weakness of most browsers is that the list of protocols supported is fixed and does not allow for expansion. Thus, there is no natural way to add Z39.50 or other protocols to browsers.

Execution of programs

An HTTP message sent from a browser can do more than retrieve a static file of information from a server. It can run a program on a server and return the results to the browser. The earliest method of achieving this was the common gateway interface (CGI), which provides a simple way for a browser to execute a program on a remote computer. The CGI programs are often called *CGI scripts*. CGI is the mechanism that most web search programs use to send queries from a browser to the search system. Publishers store their collections in databases and use CGI scripts to provide user access. An informal interpretation of the URL

`http://www.dlib.org/cgi-bin/seek?author='Arms'`

is "On the computer with domain name `www.dlib.org`, execute the program in the file `cgi-bin/seek`, pass it the parameter string `author='Arms'`, and return the output." The program might search a database for records having the word `Arms` in the author field.

The earliest uses of CGI were to connect browsers to older databases and other information. By a strange twist, now that the web has become a mature system, the roles have been reversed. People who develop advanced digital libraries often use CGI as a method to link the old system (the web) to their newer systems.

Mobile Code

Browsers rapidly became big business, with new features added continually. Some of these features are clearly good for the users; others are marketing features. The improvements include performance enhancements, elaboration to HTML, built-in support for other formats (an early addition was JPEG images), and better ways to add new formats. Two changes that are more than incremental improvements are the steady addition of security features and the introduction of *mobile code* (which permits servers to send computer programs to the client, to be executed by the browser on the user's computer).

Mobile code gives the designer of a web site the ability to create web pages that incorporate computer programs. Panel 8.2 describes one approach, which uses *applets* (small programs) written in the Java programming language. An applet can be copied from a web site to a client program and executed on the client. Because Java is a full programming

Panel 8.2

Java

Java is a general-purpose programming language that was explicitly designed for creating distributed systems, especially user interfaces, in a networked environment.

The conventional approach to making user-interface software that runs on several types of computers (e.g., web browsers and email systems) has been to write different versions. Even when versions are written in a standard language, such as C, the differences among the Windows, Unix, and Macintosh operating systems force the creator to write several versions and to modify them continually as the operating systems change. A user who wants to run a new user interface must first find and install a version of the user interface for his or her specific type of computer. At best, this is awkward and time consuming; at worst, the new program will disrupt the operation of some existing program or will introduce viruses.

Computer programs are written in a high-level language known as the *source code*. After a program has been written, it is usually compiled into the machine language of the specific computer. A Java compiler is different, however. Rather than create machine code for a specific computer system, it transforms the Java source into an intermediate code, known as *Java bytecode*, that is targeted for a software environment called a Java Virtual Machine. To run the bytecode on a specific computer requires a second step, in which each statement in the bytecode is interpreted as machine-code instructions for that computer as it is executed. Modern browsers support the Java Virtual Machine and incorporate Java interpreters.

A Java applet is a short computer program that is compiled into a file of Java bytecode and can be delivered across the network to a browser, usually by means of an HTTP command. The browser recognizes the file as an applet and invokes the Java interpreter to execute it.

Since Java is a full-feature programming language, almost any computing procedure can be incorporated in a web application. The Java system also provides programmers with a set of tools that can be incorporated into programs. These include basic programming constructs, such as strings, numbers, input and output, and data structures; the conventions used to build applets; networking services, such as URLs, TCP sockets, and IP addresses; help for writing programs that can be tailored for scripts and languages other than English; security, including electronic signatures, public/private key management, access control, and certificates; software components, known as JavaBeans, which can plug into other software architectures; and connections to databases. Since the basic functions are permanent parts of the web browser, they do not have to be delivered across the Internet with every applet; in addition, they can be written in the machine code of the individual computer, so that they can be executed faster than the interpreted bytecode.

Java has much to offer to the digital library community, but it is not perfect. Some of its defects are conventional technical ones: It is large, complex, and difficult to learn. Interpreted languages always execute slowly, and Java is no exception. Design decisions that prevent Java applets from bringing viruses across the network and infecting the user's computer also constrain legitimate programs. However, Java's biggest drawback is non-technical: Its developer, Sun Microsystems, set out to develop a standard language. Unfortunately, other companies, notably Microsoft, have created incompatible variants.

language, it can be used for complex operations such as sending an authentication form from a web site to a browser so that the user can then type in an ID and a password, which a Java applet encrypts and sends securely back to the server.

Java is not the only means of providing mobile code. An alternative is for an HTML page to include a script of instructions, usually written in the JavaScript language. JavaScript is simpler to write than Java and executes quickly. A typical use of JavaScript is to check data that a user provides as input as soon as it has been typed, without the delays of transmitting everything back to the server for validation. (Java and JavaScript are completely different languages. The similarity of the names is purely a marketing device. Java has received most of the publicity; however, JavaScript has some advantages, and both are widely used.)

Recent Advances in Design of User Interfaces

Designing user interfaces for digital libraries is part art and part science. Ultimately the success of an interface depends on the designers' instincts and their understanding of users. New concepts are steadily being introduced to the design repertoire. This section looks at some of the new ideas and topics of research. Chapter 12 describes recent research into structural metadata. This is a topic of great importance to user interfaces, since the manner in which digital objects are modeled and the structural metadata associated with them provide the raw material on which user interfaces act.

Conceptual Models

Several research groups have been looking for conceptual models that will help users navigate through the vast collections now available online. There are few landmarks on the Internet, few maps, few signposts. Hyperlinks, which are the heart of the web, lead to unexpected places, and a user can easily get lost. Users of digital libraries often have little formal training and work alone. This argues for interfaces based on conceptual models that will guide users along well-established paths, although with a little ingenuity some people are remarkably adept at finding information on the Internet. Observations suggest that experienced people meet a high percentage of their library needs with networked information, but less experienced users often get lost and have difficulty evaluating the information that they find. Panel 8.3 describes two research projects that have explored novel conceptual models for digital libraries.

Interface Design

Interface design is partly an art, but a number of general principles have emerged from recent research. Consistency in appearance, in controls, and in function is important to users. Users need feedback; they need to understand what the computer system is doing and why they see certain results. They should be able to interrupt or reverse actions. Error handling should be simple and easy to comprehend. Skilled users should be offered shortcuts, while beginners should have simple, well-defined options. Above all, the user should feel in control.

Control—particularly over graphic design and the appearance of materials—creates a continuing tension between the designers of digital libraries and the users. Many designers want the user to see materials exactly as they were designed. They want to control graphical quality, typography, window size, location of information within a window, and everything else that is important to good design. Unfortunately for designers, browsers are generic tools. The designer does not know which browser the user has, what type of computer, how fast the network, or whether the display is large or small. Users may wish to reconfigure their computer. They may prefer a large font, or a small window. They may turn off the displaying of images to reduce network delays. Therefore, a good design must be effective in a range of computing environments. The best designers have a knack of

Panel 8.3

New Conceptual Models: DLITE and Pad++

Several interesting experiments have searched for user-interface metaphors other than the desktop. Since these were research projects, the interfaces will probably never be used in a production system; however, they are important for developing concepts that can be used in other interfaces and for illustrating a design process that is based on systematic analysis of users' needs and expectations.

DLITE is an experimental user interface developed by Steve Cousins at Stanford University as a part of the Digital Libraries Initiative. Created as a user interface for the Stanford InfoBus, DLITE uses concepts from object-oriented programming. Each component is implemented as a CORBA object. (The InfoBus and CORBA are described in chapter 13.)

The conceptual model of DLITE is based on an analysis of the tasks that a user of digital libraries carries out. The following requirements were identified:

- Digital libraries consist of heterogeneous collections that must be accessible from anywhere on the Internet.
- Results created by one service may become the input to another.
- The interface must be extensible, so that new resources can be integrated easily with existing ones.
- Resources may be retained over long periods of time.
- Users must be able to collaborate.

The DLITE model describes digital libraries in terms of components, the four major types being documents, queries, collections, and services. These components are represented by icons that can be manipulated directly by a user viewing windows on the screen. For example, dragging a query onto a search service causes the search to be carried out, thus creating a collection of results. DLITE allows users to create task-specific interfaces by assembling interface elements on the screen.

The functional design of DLITE was motivated by two considerations: rapid response and the ability to add new services with minimal effort. For these reasons, the functional design is divided into user-interface clients and a user-interface server. The clients carry out the manipulations of the components. Several can run at the same time. The server provides an interface to the external services and can operate even when the clients are shut down.

Extensibility is provided by the server. When a new service is added, a new interface must be programmed in the server; the clients, however, need not be modified. Support for a variety of computers and operating systems is provided by having separate client programs for each.

Pad++, a user-interface concept conceived by Ken Perlin at New York University, has been developed by researchers at several universities and research centers. Its fundamental concept is the viewing of a large collection

of information at many different scales. Pad++ takes the metaphor of the desktop far beyond the confines of a computer display, as though a collection of documents were spread out on an enormous wall. User interactions are based on the familiar ideas of panning and zooming. A user can zoom out and see the whole collection with little detail, zoom in part way to see sections of the collection, or zoom in to see every detail. This spatial approach makes extensive use of research in human perception. Since people have good spatial memory, Pad++ emphasizes shape and position as clues to help users explore information and to recall later what they found.

When Pad++ zooms out, details do not grow ever smaller; if it did, the program would be forced to render huge numbers of infinitesimal details. Features have thresholds. Below a certain scale, they are merged into other features, or are not displayed at all. In a process known as *semantic zooming*, objects change appearance when they change size. This approach is familiar from mapmaking, in which a map of a large area represents cities but not individual buildings.

Pad++ is not intended as a universal interface for all applications. For some purposes, such as exploring large hierarchical collections in digital libraries, it may serve as a complete user interface; for other purposes, it may be a visualization component, alongside other user-interface concepts, within a conventional windowing system.

Pad++ is an interesting example of user-interface research on the relationship between system performance and usability. Several versions of the Pad++ concept have been developed, some free-standing and some as parts of web browsers. Each contains internal performance monitoring. The rendering operations are timed so that the frame refresh rate remains constant during pans and zooms. When the interface starts to be slow, medium-size features are rendered approximately; the details are added when the system is idle.

building interfaces that are convenient to use and attractive on a variety of computers, but some designers encounter difficulty in making the transition from traditional media (where they control everything) to digital libraries and the web. A common mistake is overelaboration, so that an interface is almost unusable without a fast network and a high-performance computer.

One unnecessary complication is that Netscape and Microsoft, the two leading vendors of browsers, deliver products that they have chosen to make different from one another. If a designer wishes to use some specialized features, the user must be warned that the application cannot be used

on all browsers. A user interface that works beautifully on one browser may be a disaster on the other. The manufacturers' competitive instincts have been given priority over the convenience of the user.

Functional Design

Digital libraries are distributed systems, with many computers working together as a team. Research into functional design provides designers with choices about what functions belong on which of the various computers and about the relationships between them. Between a repository (where collections are stored) and the end user (typically using a web browser) lie an assortment of computer programs which are sometimes called *middleware*. Middleware programs act as intermediaries between the user and the repository. They interpret instructions from users and deliver them to repositories. They receive information from repositories, organize it, and deliver it to the user's computer. Often the middleware provides supporting services, such as authentication or error checking, but its most important task is to match the services that a user requires with those provided by a repository.

Figures 8.2 and 8.3 show two common configurations. In each case, a user at a personal computer is using a standard web browser as a front-end interface to digital library services: a repository and a search system.

In figure 8.2, the middleware is implemented as independent computer programs (usually CGI scripts) that run on server computers somewhere on the network. The browser sends messages to the interface in a standard protocol (probably HTTP). The scripts can run on the remote service, or they can communicate with the service by means of any convenient protocol. The link between the interface scripts and the search system might use the Z39.50 protocol. The technical variations are important to the system's flexibility and performance, but functionally they are equivalent. This configuration has the disadvantage that each of the user's actions must be transmitted across the network to be processed; the user then waits for a response.

Figure 8.3 shows a more advanced configuration in which mobile code is executed on the user's computer by the browser. The code may be Java applets. Until used, the code is stored on the remote server; it is loaded into the user's computer automatically when required. This configuration has

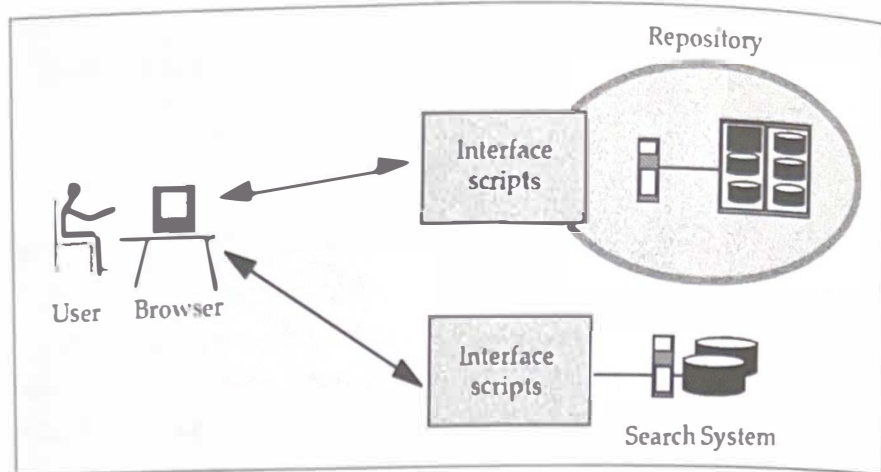


Figure 8.2
User interface with interface scripts.

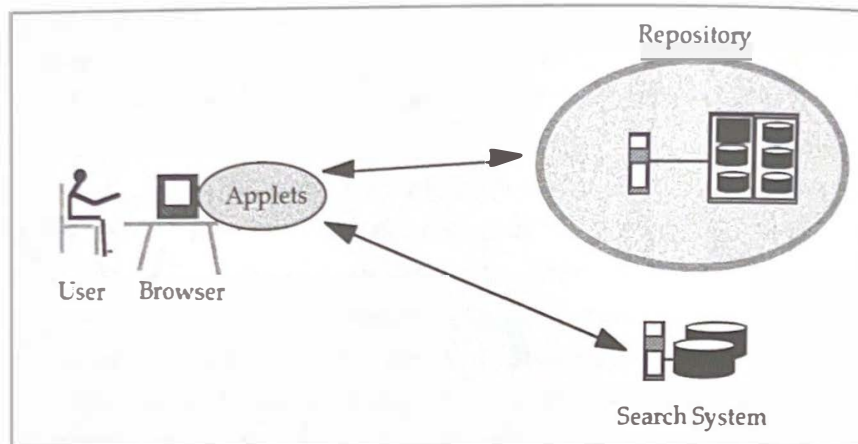


Figure 8.3
User interface with mobile code (applets).

many advantages. Since the user-interface code runs on the user's computer, it can be very responsive. The protocol between the user interface and the services can be any standard protocol or can be tailored specially for the application.

A *presentation profile* is an interesting concept that has recently emerged. Managers of a digital library associate guidelines with stored information. The guidelines suggest how the objects might be presented to the user. For example, the profile might recommend two ways to render an object, one

offering small file size and one offering full detail. The user interface is encouraged to follow the profile in rendering the data but has the option of following different approaches.

Computer Systems and Networks

The performance of computer systems and networks has a considerable impact on usability. The creator of a digital library can make few assumptions about the equipment that a user possesses, beyond the basic knowledge that every user has a personal computer attached to the Internet. However, this simple statement covers a multitude of situations. Some personal computers are more powerful than others. The quality of displays vary greatly. Some people have their own private computer, which they can configure as they wish; others may share a computer, or use more than one computer. Any digital library must assume that users will have a variety of computers, with a variety of operating systems and environments. The environments include the various versions of Microsoft's Windows, Macintosh, and a plethora of types of Unix. Differences among operating systems can be minimized by using a web browser for the user interface, but performance differences are not so easily mediated.

The usability of a computer system depends on how quickly it responds to instructions. The designer of a digital library has little or no knowledge of the quality of network connections between the user and the library. Connections vary from spectacularly fast to frustratingly slow, and even the best connections occasional experience long pauses. A user interface that is a delight to use over a 10 million bits per second local network may be unusable over an erratic dial-up link that rarely reaches its advertised speed of 28,000 bits per second. Thus, digital libraries have to balance effective use of advanced services (which require fast equipment and up-to-date software) with decent service to users who are less well provided for.

Improving the Responsiveness of Browsers

Web browsers incorporate several tricks to improve the responsiveness seen by the user. One is *internal caching*. Information that has been used once is likely to be used again (for example, when the user clicks the "back" button on a web browser), and some graphics (such as logos and buttons) may

be repeated often. Web browsers retain recently used files by storing them temporarily on the disk of a user's personal computer. These files may be HTML pages, images, or files of mobile code (perhaps in JavaScript). When a user requests a file, the browser first checks to see if it can read the file from the local cache rather than having to retrieve it from a distant server.

Another family of methods that improve the responsiveness of browsers carry out many operations in parallel. Browsers display the first part of a long file before the whole file has arrived; images are displayed in outline, and the details are filled in later as more data arrives. Several separate streams of data can be requested in parallel. In aggregate, these techniques do much to mitigate the slow and erratic performance that often plagues the Internet.

Mirroring and Caching

Replication of data is a technique widely used to enhance the performance of systems on the Internet.

If a digital library collection is to be used by people around the world, duplicate copies of the entire collection may be placed at several sites—perhaps two in Europe, two in the United States, one in Australia, and one in Japan. This is called *mirroring*. Generally, users select the nearest mirror site, which is likely to provide the best performance for those users. Mirroring also allows a user to turn to another site if one site breaks down or if a section of the network is giving trouble.

The replication of specific information (rather than an entire library) is referred to as *caching*. A cache is any store that retains recently used information, to avoid delays the next time it is used. Caches are found within the computer hardware used to help computer processors run quickly. They are found in the controllers that read data from computer disks. In the domain-name system, they are used to improve the speed and reliability of the conversion of domain names to Internet addresses.

Digital libraries have caches in many places. For example, an organization may maintain local caches of documents that have been read recently. A digital library that stores large collections on slow but cheap mass storage devices will maintain a cache on faster storage devices.

All methods of replicating data suffer from the danger that the versions may differ and that some users may therefore be receiving obsolete mater-

ial. Elaborate procedures are needed to discard replicated information after a stated time, or to check the source to see if changes have taken place.

Caches are vulnerable to security breaches. A computer system is only as secure as its weakest link, and a cache can easily be the weakest link.

Reliability and User Interfaces

Few computer systems are completely reliable, and digital libraries depend on many subsystems scattered across the Internet. In view of the number of independent components, it is remarkable that any world-wide system ever works.

Well-designed computer systems provide the user with feedback about progress in carrying out tasks. One simple form of feedback is an animation that keeps moving while the user interface is waiting for a response; this at least tells the user that something is happening. More advanced feedback is provided by an indication of the fraction of the task that has been completed and an estimate of the time to completion. In all cases, the user should be allowed to cancel a time-consuming operation.

The term *graceful degradation* describes methods that identify when a task is taking a long time and attempt to provide partial satisfaction to the user. One simple technique is to allow users of web browsers to turn off images in web pages and to see only the text. Several methods of delivering images allow a crude image to be displayed as soon as part of the data has been delivered; full resolution is added later.

User Interfaces for Multimedia Information

The materials in digital libraries are steadily becoming more varied in kind. User interfaces designed for collections of textual material may be inadequate when faced with collections of music, maps, computer software, images, statistical data, and even video games. Nonetheless, some general principles apply:

Summarization When the user has many items to choose from, it is convenient to represent each by some small summary information that conveys its essence. A book is usually summarized by the author's name and the title (e.g., Arms, *Digital Libraries*). For a picture, a thumbnail-size image is usually sufficient. More esoteric information, such as a computer program or a film, is difficult for a digital library to summarize.

Sorting and categorization Libraries organize information by classification or by ordering, but many types of information (e.g., musical tunes) lack easily understood ways of organization. Fields guides to birds are easy to use because they divide birds into well-known categories, such as ducks, hawks, and finches; guides to wildflowers are more awkward because flowers lack equivalent divisions.

Presentation Many types of information require specialized equipment for the best presentation to the user. High-quality digitized video, for example, requires specialized displays, powerful computers, high-speed network connections, and appropriate software. A digital library should make full use of such capabilities, if they are available; to present such information to users who have lesser equipment, alternative means of presentation must be provided.

There is no single way to design digital libraries for different types of information. User interfaces have to be tailored for the different classes of material and probably also for different categories of users. Panel 8.4 describes one good example of a user interface: the Informedia digital library of digitized segments of video.

User Interfaces and the Effectiveness of Digital Libraries

During the past few years, the quality of user-interface design has improved dramatically. It is now assumed that new users can begin productive work without any training. Most important, there are now numerous examples of fine interfaces on the Internet that others can use for inspiration and as models. Standards of graphical design get better every year. (The materials for *D-Lib Magazine* were redesigned three times in as many years. Each design was considered elegant in its day but needed a face lift a year later.)

Good support for users is more than a cosmetic flourish. Elegant design, appropriate functionality, and responsive systems make a measurable difference in the effectiveness of digital libraries. When a system is hard to use, the users may fail to find important results, may misinterpret what they do find, or may give up in disgust. A digital library is only as good as its interface.

Panel 8.4
Informedia

Informedia, a program at Carnegie Mellon University led by Howard Wactlar, is carrying out research on digital libraries of video materials—specifically, broadcast news and documentary programs. The programs are automatically broken into short segments, such as the individual items in a news broadcast. The emphasis is on developing automatic methods for extracting information from the video objects, so as to minimize human intervention. More than 1000 hours of digitized video material (obtained from the Cable News Network, the British Open University, the Pittsburgh public television station WQED, and other sources) are being used.

One of the questions associated with video or audio material is how to provide users with a quick overview of an item. The reader of a book can look at the contents page or flick through the pages to see the chapter headings, but there is no way to flick through a video. *Video skimming*, a technique developed by the Informedia project, uses automatic methods to extract important words and images. In combination, the selected words and images provide a video abstract that conveys the essence of the full video segment. Informedia's research on video skimming has gone far beyond what was expected at the start of the project.

The Informedia user interface employs the conceptual model of searching an index of the collection, obtaining a set of hits, then browsing through the items found. Queries may be entered by typing or by means of a speech-recognition program. After a search, the interface presents ranked results. Each video clip is represented by an image. When the cursor is moved over the image, a text summary is provided. The image is selected automatically as representative of the video segment as it relates to the current query; the summary is created through natural-language processing. The user may then click on an image to view the segment. The interface consists of a video viewer with controls (such as "play" and "stop") similar to those of a home video player. The interface is sufficiently intuitive to be used by individuals who have not been trained on the system. It maximizes feedback to the user at each step. Much of the testing has been done with Pittsburgh high school students.

Information Retrieval and Descriptive Metadata

Helping users find information is among the core services provided by digital libraries. This chapter, the first of two on this general subject, begins with a discussion of *descriptive metadata*—the catalogs, indexes, and other summary information used to describe objects in a digital library. This is followed by a discussion of the methods by which bodies of text are searched for specific information. Chapter 11 extends the concepts discussed here to discuss distributed searching—that is, searching for information that is spread across separate collections or scattered over many computer systems.

These two chapters concentrate on methods used to search for specific information. Direct searching is just one of the strategies that people use to discover information, however. Browsing—unstructured exploration of a body of information—is a popular and effective method for discovering the unexpected. Most traditional libraries arrange their collections by subject so as to facilitate browsing. Dewey Decimal classification and Library of Congress classification provide both subject information and a hierarchical structure that can be used to organize collections. The most widely used web information service, Yahoo, is fundamentally a classification of web resources augmented by searching. Digital libraries, with their hyperlinks, lend themselves to strategies that combine searching and browsing.

The range of the needs of users of digital libraries illustrates why information discovery is such a complex topic and why no single approach satisfies all users or suits all materials. The objective of a *comprehensive search* is to find everything on a specified topic. Serious scholarly, scientific, or legal study typically begins with a comprehensive search to discover prior work in the field. In a *known-item search*, the user is looking for a specific

item, such as the opening chapter of *Moby Dick*. Often the reference is not perfect and a search must be done to find the item; however, there is indeed a specific item, and the user will recognize it when it is found. Then there are *searches for facts*. Facts (e.g., the name of the capital of Barbados, or the atomic weight of mercury) are specific items of information that may be found in many sources of information. A search for a fact is complete when the fact is found. The source of the fact must be reliable, but there may be many possible sources. *Overview information on a topic* is yet another common need. A request such as “How does one organize a wedding?” or “What is public-key encryption?” may lead to many suitable sources of information, but usually the user wants a small selection. Once one useful item has been found, a user may wish to know about *related items*. In research, for example, a common question is “Is there any later research that builds on the work reported in this article?”

Descriptive Metadata

Many methods of information discovery do not search the actual objects in the collections; rather, they work from descriptive metadata about the objects. The metadata for an object may be a catalog entry, an indexing record, or an abstract. Descriptive metadata usually is stored separately from the objects that it describes, but sometimes it is embedded in the objects.

Descriptive metadata is usually expressed as text; however, it can be used to describe information that is in formats other than text, such as images, sound recordings, maps, and computer programs. A single catalog can combine records for all genres, media, and formats. This enables users of digital libraries to discover materials in all media by searching textual records.

Descriptive metadata is usually created by professionals. Library catalogs and scientific indexes represent huge investments of work by skilled people, sustained over decades or even centuries. This economic fact is crucial to understanding current trends. On one hand, it is important to build on the investments and the expertise behind them. On the other, there is a great incentive to find cheaper and faster ways to create metadata, either

by automatic indexing or through the use of computer tools that enhance human expertise.

Catalogs

Catalog records are short records that provide summary information about a library object. The word *catalog* is applied to records that have a consistent structure, organized according to systematic rules. An *abstract* is a short summary of a longer document. Other types of indexing records are less formal than a catalog record but have more structure than a simple abstract.

Library catalogs serve many functions, not only information retrieval. Some catalogs provide comprehensive bibliographic information that cannot be derived directly from the objects. This includes information about authors or the provenance of museum artifacts. For managing collections, catalogs contain administrative information, such as where items are stored (whether online or on library shelves). Catalogs are usually much smaller than the collections they represent; in conventional libraries, materials that are stored on miles of shelving are described by records that can be contained in a group of card drawers at one location or in an online database. Indexes to digital libraries can be mirrored for higher performance and greater reliability.

Information in catalog records is divided into *fields* and *subfields*, with tags to identify them. For example, there might be a field for an author's name with a subfield for the author's surname. The Anglo-American Cataloging Rules and the MARC format are used for many types of material, including monographs, serials, and archives. Because of the labor required to create a detailed catalog record, materials are cataloged once (often by a national library, such as the Library of Congress) and the records are then distributed to other libraries through utilities such as the OCLC.

The role of MARC and the related cataloging rules in digital libraries is a subject of debate. How far can traditional methods of cataloging migrate to support new formats, media types, and methods of publishing? MARC cataloging retains its importance for conventional materials. Librarians have extended it to some of the newer types of object found in digital libraries, but it has not been adopted by organizations other than traditional libraries.

Abstracting and Indexing Services

The sciences and other technical fields rely on abstracting and indexing services more than they rely on catalogs. Each scientific discipline has a service to help users find information in journal articles—e.g., Medline for medicine and biology, Chemical Abstracts for chemistry, and Inspec for physics, computing, and related fields. Each service indexes the articles in a large set of journals. The record for an article includes basic bibliographic information (authors, title, date, etc.) and subject information, organized for information retrieval. The details differ, but the services have many similarities. Since abstracting and indexing services emerged at a time when computers were slower and more expensive than they are today, the information is structured to support simple textual searches; however, the records have proved useful in more flexible systems.

Scientific users often want information on a specific subject. Because of the subtleties of language, subject searching is unreliable unless there is indexing information that describes the subject of each object. The subject

Panel 10.1

Medical Subject Headings

The National Library of Medicine has provided information-retrieval services for medicine and related fields since the 1960s. Since in medicine (a complex field with a huge terminology) it is often the case that the same concept may be described by scientific terms or by any of multiple vernacular terms, the NLM has developed a controlled-vocabulary, known as MeSH, that provides a thesaurus of subject headings for each of the 400,000 articles it indexes every year and for every book acquired by the library. These subject headings can be used for information retrieval. In addition to MeSH (which has about 18,000 primary headings), there is a separate thesaurus of about 80,000 chemical terms. The terms in MeSH are organized in a hierarchy. At the top are general terms, such as *anatomy*, *organisms*, and *diseases*. Going down the hierarchy, anatomy (for example) is divided into sixteen topics, beginning with *body regions* and *musculoskeletal system*. *Body regions* is further divided into sections, such as *abdomen*, *axilla*, and *back*. Some of the sections are then subdivided. Thousands of cross-references are provided.

The success of MeSH depends on the professionals who maintain the thesaurus and on those who assign subject terms to documents. Successful use of MeSH also depends on users or reference librarians who understand medicine and are able to formulate queries using the terms and the structures that MeSH employs.

Panel 10.2
The Art and Architecture Thesaurus

The Art and Architecture Thesaurus was developed by the J. Paul Getty Trust as a controlled vocabulary for describing and retrieving information on fine art, architecture, decorative art, and material culture. It has almost 120,000 terms for objects, textual materials, images, architecture and culture from all periods and all cultures, with an emphasis on Western civilization. The thesaurus can be used by archives, museums, and libraries to describe items in their collections. It also be used to search for materials.

Serious work on this thesaurus began in the early 1980s, when the Internet was in its infancy; however, the data was created in a flexible format that allowed the creation of many versions, including an open-access version on the web, a printed book, and various computer formats. The Getty Trust explicitly organized the thesaurus so that it could be used by computer programs for information retrieval and for natural-language processing.

The Art and Architecture Thesaurus is arranged in seven categories, each containing a hierarchy of terms. The categories are *associated concepts*, *physical attributes*, *styles and periods*, *agents*, *activities*, *materials*, and *objects*. A single concept is represented by a cluster of terms, one of which is established as the *descriptor* (i.e., the preferred term). The thesaurus provides not only the terminology for objects but also the vocabulary used to describe them (e.g., style, period, shape, color, construction, use) and scholarly concepts such as theories and criticism.

The costs of developing and maintaining a large, specialized thesaurus are huge. Even in a mature field, such as art and architecture, terminology is changing continually, and the technical staff has to support new technology. Even though the Getty Trust is extremely rich, developing the thesaurus took many years.

information can be an abstract, keywords, subject terms, or other information. Some services ask authors to provide keywords or an abstract, but this leads to gross inconsistencies. It is far better to have professional indexers assign subject information to each item.

An effective but expensive approach is to use a *controlled vocabulary*. That is, where several terms could be used to describe a concept, one is used exclusively. The indexer is given a list of approved subject terms and rules for applying them. No other terms are permitted. This is the approach used for the Library of Congress subject headings and the National Library of Medicine's MeSH headings.

The controlled-vocabulary approach requires trained indexers. It also requires skilled users, because the terms used in a search query must be consistent with the terms assigned by the indexer. Medicine in the United States is especially fortunate in having a cadre of reference librarians. In digital libraries, the trend is to provide users with tools that permit them to find information directly, without the help of a reference librarian. A thesaurus, such as MeSH or the Art and Architecture Thesaurus, can be used to relate the terminology that a user provides to the controlled terms used in indexing.

The Dublin Core

Since 1995, an international group led by Stuart Weibel of OCLC has been working to devise a set of simple metadata elements that can be applied to a wide variety of digital library materials. The set of elements developed by this group is known as the Dublin Core, after Dublin, Ohio, the home of OCLC, where the first meeting was held. Several hundred people have participated in Dublin Core workshops and have discussed the design by electronic mail. Their spirit of cooperation is a model of how individuals with diverse interests can work together. The fifteen elements they have selected are summarized in panel 10.3.

Simplicity is both the strength and the weakness of the Dublin Core. Whereas traditional cataloging rules are long and complicated, requiring professional training to apply effectively, the Dublin Core can be described simply. However, simplicity conflicts with precision. The team has struggled with this tension. The initial aim was to create a single set of metadata elements for untrained people who publish electronic materials to use in describing their work. Some people continue to hold this minimalist view; they would like to see a simple set of rules that anyone can apply. Others prefer the benefits that come from more tightly controlled cataloging rules and would accept the additional labor and cost. They point out that extra structure in the elements results in extra precision in the metadata records. For example, if entries in a subject field are drawn from the Dewey Decimal Classification, it is helpful to record that fact in the metadata. To further enhance the effectiveness of the metadata for information retrieval, several of the elements are expected to have recommended lists of values. Thus, there might be a specified set of types, and indexers would be advised to select from the list.

Panel 10.3
The Dublin Core

The following fifteen elements constitute the metadata set of the Dublin Core. All elements are optional, and all can be repeated. (These descriptions were condensed from the official Dublin Core definitions with the permission of the design team.)

Title The name given to the resource by the creator or publisher.

Creator The person or organization primarily responsible for the intellectual content of the resource (authors in the case of written documents; artists, photographers, or illustrators in the case of visual resources).

Subject The topic of the resource. Typically, subject will be expressed as a keyword or a phrase that describes the subject or the content of the resource. The use of controlled vocabularies and formal classification schemes is encouraged.

Description A textual description of the content of the resource, including abstracts in the case of document-like objects and content descriptions in the case of visual resources.

Publisher The entity responsible for making the resource available in its present form—e.g., a publishing house, a university department, or a corporate entity.

Contributor A person or organization not specified in a creator element that has made significant intellectual contributions to the resource but whose contribution is secondary to any person or organization specified in a creator element—e.g., an editor, a transcriber, an illustrator.

Date A date associated with the creation or availability of the resource.

Type The category of the resource—e.g., home page, novel, poem, working paper, preprint, technical report, essay, dictionary.

Format The data format of the resource (used to identify software and possibly hardware that might be needed to display or operate the resource).

Identifier A string or number used to uniquely identify the resource. Examples for networked resources include URLs and URNs.

Source Information about a second resource from which the present resource is derived.

Language The language of the intellectual content of the resource.

Relation An identifier of a second resource and its relationship to the present resource. This element permits links between related resources and resource descriptions to be indicated. Two examples are an edition of a work and a chapter of a book.

Coverage Spatial locations and temporal durations characteristic of the resource.

Rights A rights-management statement, an identifier linked to such a statement, or an identifier linked to a service providing information about rights management for the resource.

Object Models, Identifiers, and Structural Metadata

Information comes in many forms and formats, each of which must be captured, stored, retrieved, and manipulated. Much of the early development of digital libraries concentrated on material that has a direct analog to some physical format. These materials can usually be represented as simply structured computer files. Digital libraries can go far beyond such simple digital objects; they include anything that can be represented in a digital format. The digital medium allows for new types of library objects, such as software, simulations, animations, movies, slide shows, and soundtracks, and for new ways to structure material. Computing has introduced its own types of objects, including spreadsheets, databases, symbolic mathematics, and hypertext. Increasingly, computers and networks support continuous streams of digital information—notably speech, music, and video. Even the simplest digital objects may come in many versions and be replicated many times.

Methods for managing this complexity fall into several categories: *identifiers* for digital objects; *data types*, which specify what the data represents; and *structural metadata*, which represent the relationship between digital objects and their component parts. In combination, these techniques create an *object model*, a description of some category of information that enables computer systems to store and provide access to complex information. Interoperability and long-term persistence are constant themes in studies of these topics. Information in today's digital libraries must be usable many years from now by people using computer systems that have not yet been imagined.

A single item may be stored in several alternate digital formats. When existing material is converted to digital form, the same physical item may be converted several times. For example, a scanned photograph may have a high-resolution archival version, a medium-quality version, and a thumbnail. In some cases these formats are exactly equivalent (e.g., an uncompressed image and the same image stored with a lossless compression), and it is possible to convert from one to the other. In other cases, the different formats contain different information (e.g., differing representations of a page of text in SGML and PostScript formats).

Structural Types

To the user, an item appears as a single entity, and the internal representation is unimportant. A bibliography or an index will normally refer to such an item as a single object in a digital library. Yet the internal representation as stored within the digital library may be complex. Structural metadata is used to represent the various components and the relationships among them. The choice of structural metadata for a specific category of material creates an object model.

Different categories of objects require different object models—for example, text with SGML markup, web objects, computer programs, digitized sound recordings. Within each category, rules and conventions describe how to organize the information as sets of digital objects. For example, specific rules describe how to represent a digitized sound recording. For each category, the rules describe how to represent materials in the library, how the components are grouped as a set of digital objects, the internal structure of each component, the associated metadata, and the conventions for naming the digital objects. The categories are distinguished by *structural type*.

Object models for computer programs have been a standard part of computing for many years. A large computer program consists of many files of programs and data, with complex structure and interrelations. This relationship is described in a separate data structure that is used to compile and build the program. For a Unix program, this is called a *make file*.

Structural types must be distinguished from *genres*. For information retrieval, it is convenient to provide descriptive metadata that describes the genre. (For example, some of the genres of popular music are jazz, blues,

rap, and rock.) Genre is a natural and useful way to describe materials for searching and other bibliographic purposes, but another categorization is required for managing distributed digital libraries. Though feature films, documentaries, and training videos are clearly different genres, their digitized equivalents may be encoded in precisely the same manner and processed identically; they are the same structural type. Conversely, two texts may be of the same genre, but if one is represented with SGML markup and the other in PDF format they have different structural types and object models.

Panel 12.8 describes an object model for a scanned image. This model was developed to represent digitized photographs, but the same structural type can be used for any bit-mapped image. For example, a map, a poster, a playbill, a technical diagram, and a baseball card represent different content, but they are stored and manipulated in a computer with the same

Panel 12.8
An Object Model for Scanned Images

In joint work with the Library of Congress, members of the Corporation for National Research Initiatives have developed object models for some types of material digitized by the National Digital Library Program. The first model was for digitized images, such as scanned photographs. When each image is converted, several digital versions are made. Typically, there is a high-resolution archival version, a medium resolution version for access via the web, and one or more thumbnails. Each of these versions has its own metadata in addition to the metadata shared by all the versions.

The first decision made in developing the object model was to store the bibliographic metadata that describes the content in separate catalog records. Structural metadata (which relates the various versions) and administrative metadata (used for access management) is stored with the scanned images and is part of the object model.

Since the individual digital versions may be used independently on some occasions, each version is stored as a separate digital object. Each of these is identified by a handle and has two elements: a digitized image and metadata. Another digital object, known as a *meta-object*, is used to bring the versions together. The main purpose of the meta-object is to provide a list of the versions. It also has a data-type field that describes the function of each image (e.g., for reference, for access, or as a thumbnail). The meta-object, which has its own handle, contains metadata that is shared by all the versions of the image.

structure. Current thinking suggests that even complicated digital library collections can be represented by a small number of structural types. It has been suggested that fewer than ten structural types would be adequate for all the categories of material being converted by the Library of Congress. Among these are digitized images, sets of page images, sets of page images with associated SGML text, digitized sound recordings, and digitized video recordings.

Object Models for Interoperability

Object models are evolving slowly. How the various parts should be grouped together can rarely be specified in a few dogmatic rules. The decision depends on the context, the specific objects, the type of content, and sometimes the actual content. After the introduction of printing, the structure of a book took decades to develop to the form that we know today, with its frontmatter, chapters, figures, and index. Not surprisingly, few conventions for object models and structural metadata have yet emerged in digital libraries. Structural metadata is still at the stage where every digital library is experimenting with its own specifications, making frequent modifications as experience suggests improvements.

This might appear to pose a problem for interoperability, since a client program will not know the structural metadata used to store a digital object in an independent repository. However, clients do not need to know the internal details of how repositories store objects; they need to know the functions that the repository can provide.

Consider the storage of printed materials that have been converted to digital formats. A single item in a digital library collection consists of a set of page images. If the material is also converted to SGML, there will be the SGML markup, the Document Type Definition, style sheets, and related information. Within the repository, structural metadata defines the relationship among the components, but a client program need not know these details. In chapter 8 we looked at a user-interface program that manipulates a sequence of page images. The functions included displaying a specified page and going to a page with a specific page number. A user interface that is aware of the functions supported for this structural type of information is able to present the material to the user without knowing how it is stored on the repository.

Dissemination

In a digital library, the form in which information is stored is rarely the form in which it is delivered to the user. In the web model of access, information is copied from the server to the user's computer, where it is then rendered for use by the user. The rendering typically takes the form of converting the data in the file into a screen image, using suitable fonts and colors, and embedding the image in a display with windows, menus, and icons.

Getting stored information and presenting it to the user of a digital library can be much more complicated than it is in the web model. At the very least, a general architecture must allow the stored information to be processed by computer programs on the server before being sent to the client, or on the client before being presented to the user. The data that is rendered need not have been stored explicitly on the server. Many servers run a computer program on a collection of stored data, extract certain information, and provide it to the user. This information may be fully formatted by the server, but often it is transmitted as a file that has been formatted in HTML or in some other intermediate format that can be recognized by the user's computer.

Important types of dissemination include direct interaction between the client and the stored digital objects and continuous streams of data. When a user interacts with the information, access by the user is not a single, discrete event. Video games are a well-known example of this; the same is true of any interactive material, such as a simulation of a physical situation. Access to such information consists of a series of interactions, which are guided by the user's control and by the structural metadata of the individual objects.

Often a client program may have a choice of disseminations. The selection may be determined by the equipment a user has—for example, users on a slow network sometimes choose to turn off images and receive only the text of web pages. It may be determined by the software on the user's computer—for example, if a text is available in both HTML and PostScript versions, a user whose computer does not have software capable of viewing PostScript will want the HTML dissemination. It may be determined on non-technical grounds, such as the user's wishes or convenience—for

example, a user may prefer to see a short version of a video program rather than the full-length version.

One of the purposes of object models is to provide the user with a variety of dissemination options. Several research projects aim to enable a client computer to discover automatically the range of disseminations that are available and select the one that is most useful at any given time. Typically, one option will be a default dissemination—what the user receives unless a special request is made. Another option is a short summary; this is often a single line of text (perhaps author and title), but it can be something fancier, such as a thumbnail image or a short segment of video. Whatever method is chosen, the aim of the summary is to provide the user with a small amount of information that helps to identify and describe the object. This is still an area of research; there are few good examples of such systems in practical use.

This chapter looks at methods for storing digital materials in repositories and archiving them for the long term. It also examines the protocols that provide access to materials stored in repositories. It may seem strange that such important topics should be addressed so late in the book, since long-term storage is central to digital libraries, but there is a reason. Throughout the book, the emphasis has been on what actually exists today. Research topics have been introduced where appropriate, but most of the discussion has been of systems that are used in libraries today. The topics in this chapter are less well established. Beyond the ubiquitous web server, there is little consensus about repositories for digital libraries and the field of digital archiving is new. The needs are beginning to be understood, but the methods are still embryonic—particularly in the field of archiving.

Repositories

A *repository* is any computer system whose primary function is to store digital material for use in a library. Repositories are the bookshelves of digital libraries. They can be huge or tiny, storing millions of digital objects or just a single object. In some contexts a mobile agent that contains a few digital objects can be considered a repository, but most repositories are straightforward computer systems that store information in a file system or database and present it to the world through a well-defined interface.

Web Servers

Currently, the most common form of repository is the web server. Several companies provide excellent web servers. The main differences between

Panel 13.1 Web Servers

A *web server* is a computer program whose tasks are to store files and to respond to requests in HTTP and associated protocols. A web server runs on a computer connected to the Internet. This computer can be a dedicated one, a shared one that also runs other applications, or a personal computer that provides a small web site.

At the heart of a web server is a process called `httpd`. The letter *d* stands for *daemon*. A daemon is a program that runs continuously but spends most of its time idling until a message arrives for it to process. The HTTP protocol runs on top of TCP, the Internet transport protocol. TCP provides several *ports* (addresses) for every computer. The web server is associated with one of these ports—usually port 80, although others can be specified. When a message arrives at this port, it is passed to the daemon. The daemon starts up a process to handle this particular message, and continues to listen for more messages to arrive. In this way, several messages can be processed at the same time without tying up the daemon with the details of their processing.

The actual processing that a web server carries out is tightly controlled by the HTTP protocol. Early web servers did little more than implement the *get* command. This command receives a message containing a URL from a client. The URL specifies a file stored on the server. The server retrieves the file and returns it to the client. The HTTP connection for the specific message then terminates.

As HTTP has added features and as web sites have grown larger, web servers have become more complex. In addition to the full set of HTTP commands, they have to support CGI scripts and other extensions. One of the requirements of web servers (and also of web browsers) is to continue to support older versions of the HTTP protocol. They have to be prepared to receive messages in any version of the protocol and to handle them appropriately. Furthermore, web servers have steadily added security features. Version 1.1 of the protocol also includes persistent connections, which permit several HTTP commands to be processed over a single TCP connection.

The biggest web sites are so busy that they need more than one computer. Several methods are used to share the load. One straightforward method is simply to replicate the data on several identical servers. This is feasible and convenient when the number of requests is high but the volume of data is moderate. A technique called “DNS round robin” is used to balance the load by means of an extension of the domain-name system that allows a domain name to refer to a group of computers with different IP addresses. For example, the domain name `www.cnn.com` refers to a set of computers each of which has a copy of the Cable News Network’s web site. When a user accesses this site, the domain-name system chooses one of the computers to service the request. But replication of a web site is inconvenient if the volume of data is huge or if it is changing rapidly. Web search services provide an example. Some web search systems use separate computers to carry out the search, to assemble the

their products are in the associated programs that are linked to them, such as electronic mail, indexing programs, security systems, electronic payment mechanisms, and other network services.

For digital libraries, web servers provide moderate functionality with low costs. These attributes have led to broad acceptance and a basic level of interoperability. The web owes much of its success to its simplicity, and web servers are part of that success, but some of their simplifying assumptions are awkward for the implementers of digital libraries. Web servers support only one object model: a hierarchical file system in which information is organized into separate files. Their processing is inherently stateless; each message is received, processed, and forgotten.

Advanced Repositories

Although web servers are widely used, other types of storage systems are used as repositories in digital libraries. In business data processing, relational databases are the standard way to manage large volumes of data. Relational databases are based on an object model that consists of data tables and relations among them. These relations allow data from different tables to be joined or viewed in various ways. The tables and the data fields within a relational database are defined by a schema and a data dictionary. Relational databases are excellent at managing large amounts of data with a well-defined structure. Many of the large publishers mount collections on relational databases, with a web server providing the interface between the collections and the user.

Catalogs and indexes for digital libraries are usually mounted on commercial search systems. These systems have a set of indexes that refer to the digital objects. Typically, they have a flexible and sophisticated model for indexing information but a primitive model for the actual content. Many began as full-text systems, and their greatest strength lies in providing information retrieval for large bodies of text. Some systems have added relevance feedback, fielded searching, and other features that they hope will increase their functionality and hence their sales.

Relational databases and commercial search systems provide good tools for loading data, validating it, manipulating it, and protecting it. Access control is precise, and services that are important in business applications (such as audit trails) are provided. There is an industry-wide trend for database systems to add full-text searching, and for search systems to provide

some parts of the relational database model. These extra features can be useful, but no company has yet created a system that combines the best of both approaches.

Although some digital libraries have used relational databases successfully, the relational model of data is not flexible enough for the rich object models that are emerging. The consensus among the leading digital libraries appears to be that more advanced repositories are needed. A possible set of requirements for such a repository is as follows.

Information hiding The internal organization of the repository should be hidden from client computers. It should be possible to reorganize a collection, change its internal representation, or move it to a different computer without any external effect.

Object models Repositories should support a flexible range of object models, with few restrictions on data, metadata, external links, and internal relationships. New categories of information should not require fundamental changes to other aspects of the digital library.

Open protocols and formats Clients should communicate with the repository through well-defined protocols, data types, and formats. The repository architecture must allow incremental changes of protocols as they are enhanced over time. This applies, in particular, to access management. The repository must allow a broad set of policies to be implemented at all levels of granularity.

Reliability and performance The repository should be able to store very large volumes of data, should be absolutely reliable, and should perform well.

Metadata in Repositories

Repositories store both data and metadata. The metadata can be considered as falling into the general classes of descriptive, structural, and administrative. Identifiers may need to distinguish elements of digital objects as well as the objects themselves. Storage of metadata in a repository requires flexibility, since there is a range of storage possibilities:

- Descriptive metadata often is stored in catalogs and indexes that are managed outside the repository. These catalogs and indexes may be held in separate repositories and may cover material in many independent digital libraries. Identifiers are used to associate the metadata with the corresponding data.